**MongoDB Lab Assignments- Day 1**

**INSERT DOCUMENTS**

Insert the following documents into a **movies** collection

**title : Fight Club**

**writer : Chuck Palahniuk**

**year : 1999**

**actors : [**

**Brad Pitt**

**Edward Norton**

**]**

* db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})

**title : Pulp Fiction**

**writer : Quentin Tarantino**

**year : 1994**

**actors : [**

**John Travolta**

**Uma Thurman**

**]**

* db.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]})

**title : Inglorious Basterds**

**writer : Quentin Tarantino**

**year : 2009**

**actors : [**

**Brad Pitt**

**Diane Kruger**

**Eli Roth**

**]**

* db.movies.insert({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**

* db.movies.insert({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**

* db.movies.insert({title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"})

**title : The Hobbit: The Battle of the Five Armies**

**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.**

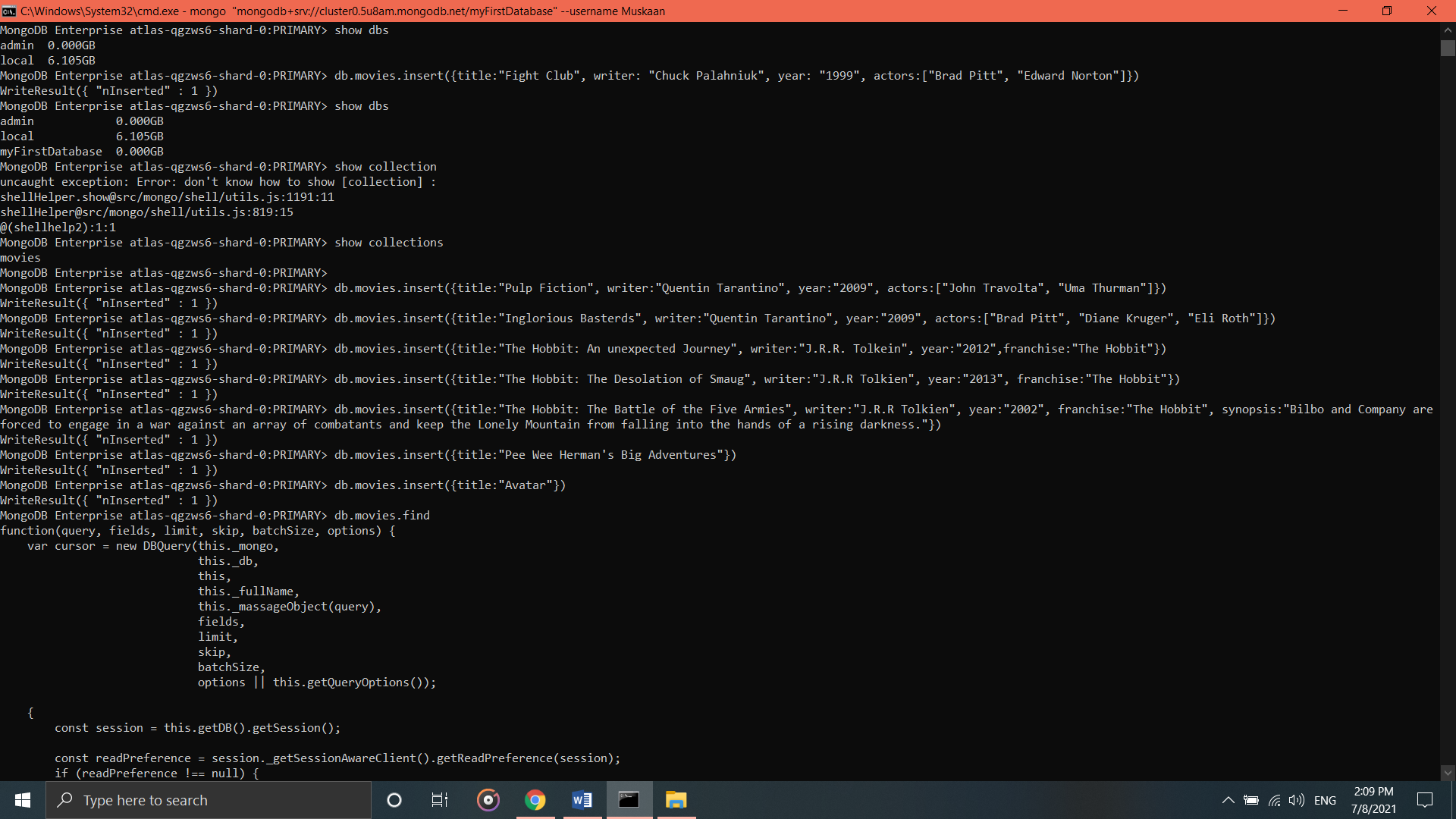
* db.movies.insert({title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", franchise:"The Hobbit", synopsis:"Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."})

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

* db.movies.insert({title:"Pee Wee Herman's Big Adventures"})

**title : Avatar**

* db.movies.insert({title:”Avatar”})



**Query / Find Documents**

Query the **movies** collection to

1. **Get all documents**

* db.movies.find()

1. **Get all documents with write set to “Quentin Tarantino”**

* db.movies.find({writer:"Quentin Tarantino"})

1. **Get all documents where actors include “Brad Pit”**

db.movies.find({actors:"Brad Pitt"})

1. **Get all documents with franchise set to “The hobbit”**

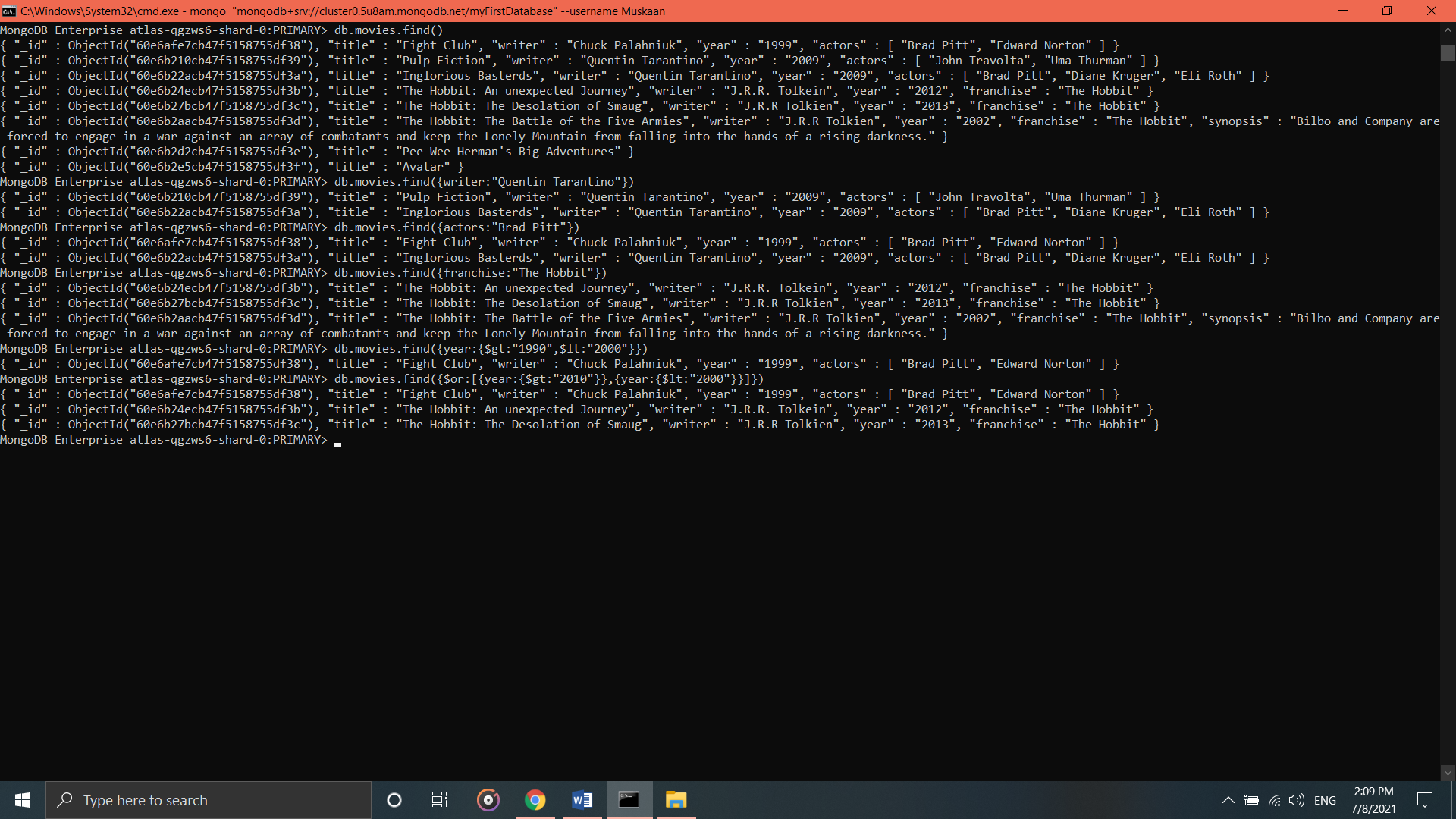
* db.movies.find({franchise:"The Hobbit"})

1. **Get all movies released in the 90s**

* db.movies.find({year:{$gt:"1990", $lt:"2000"}})

1. **Get all movies released before the year 2000 or after 2010**

* db.movies.find({$or:[{year:{$gt:"2010"}},{year: {$lt:"2000"}}]})



**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."**

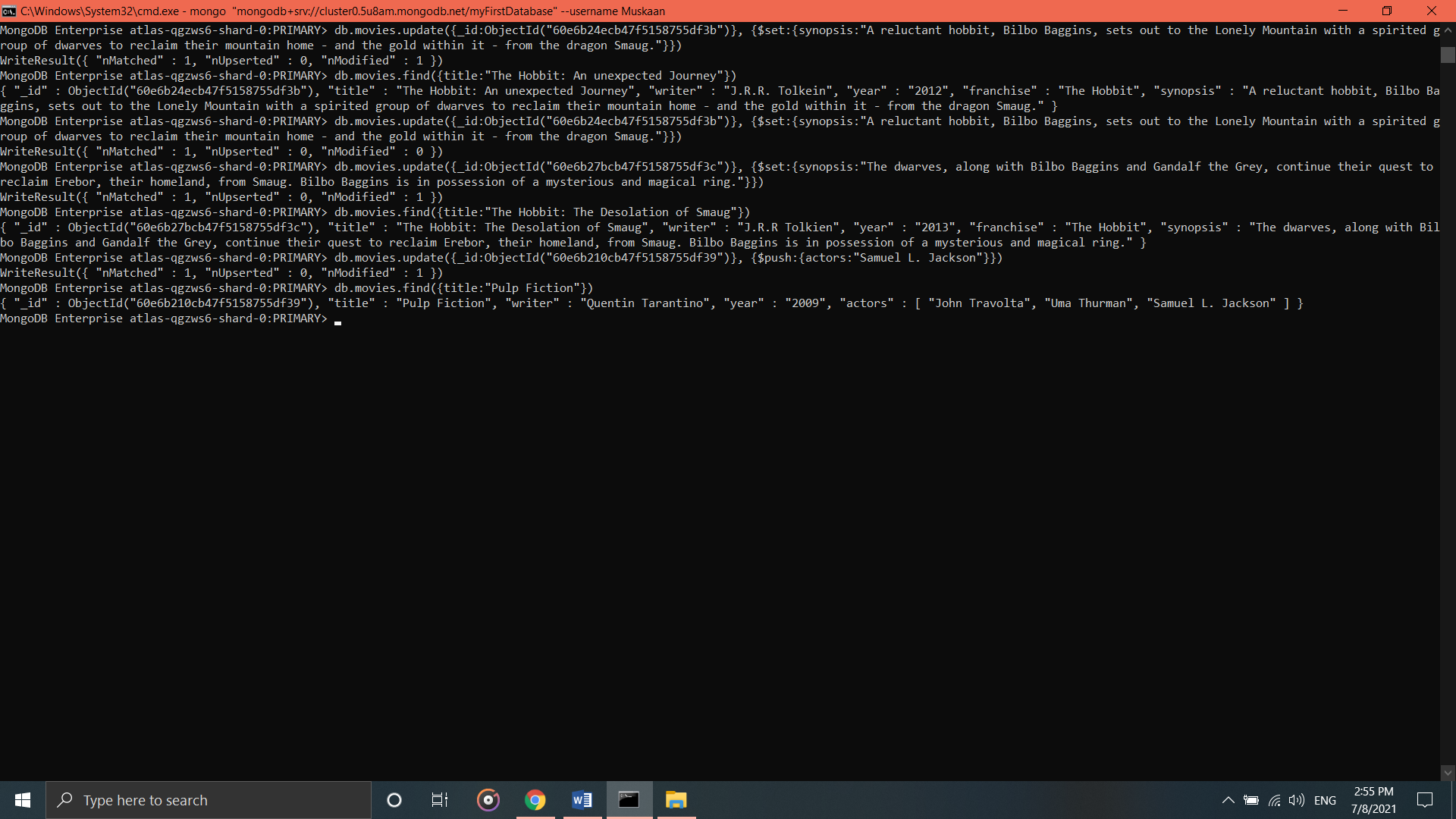
* db.movies.update({\_id:ObjectId("60e6b24ecb47f5158755df3b")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})

1. **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."**

* db.movies.update({\_id:ObjectId("60e6b27bcb47f5158755df3c")}, {$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."}})

1. **add an actor named “Samuel L. Jackson” to the movie “Pulp Fiction”**

* db.movies.update({\_id:ObjectId("60e6b210cb47f5158755df39")}, {$push:{actors:"Samuel L. Jackson"}})



**TEXT SEARCH**

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

* db.movies.find({synopsis:{$regex:"Bilbo"}})

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

* db.movies.find({synopsis:{$regex:"Gandalf"}})

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

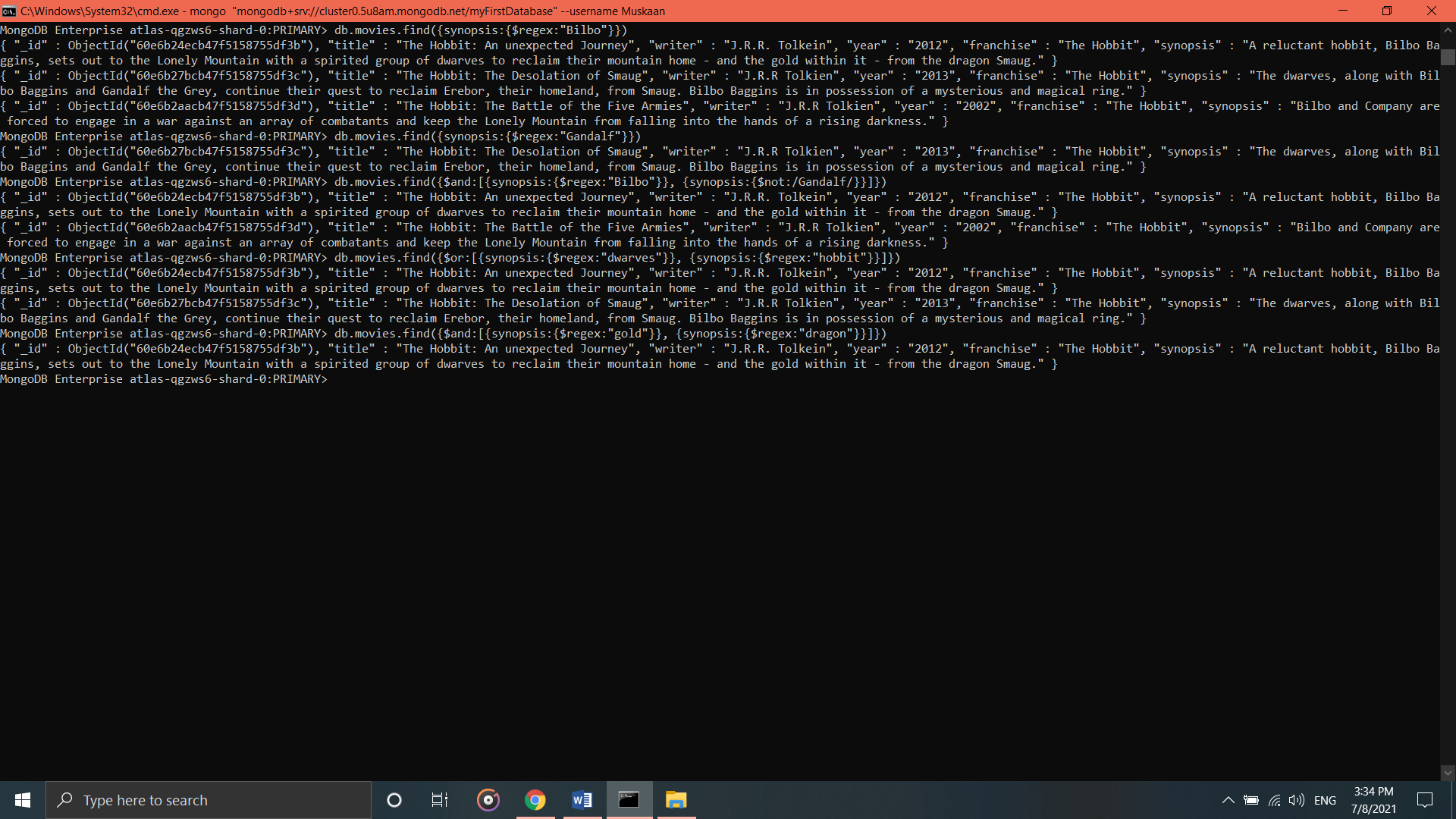
* db.movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]})

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

* db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})

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

* db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]})

****

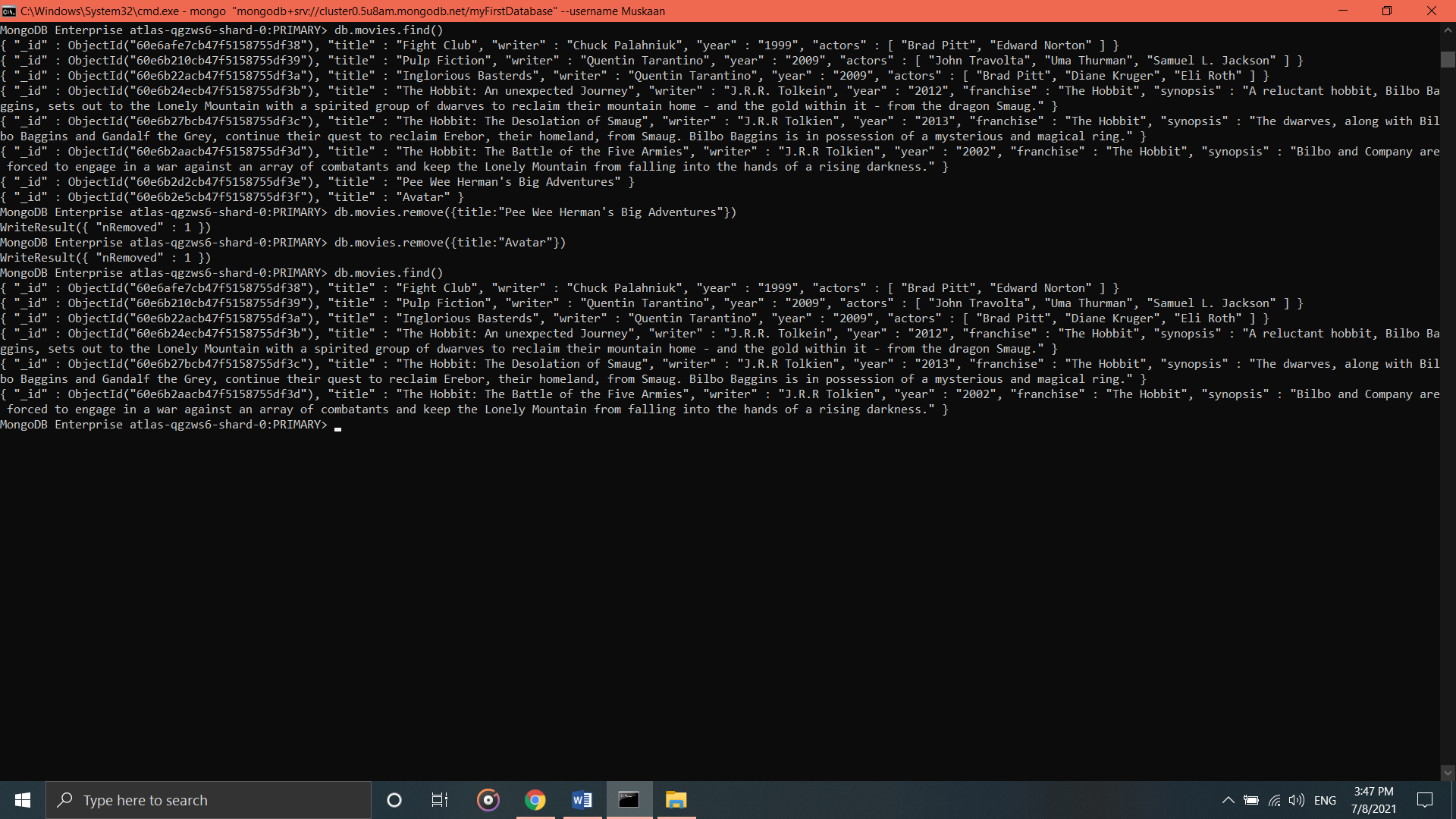
**DELETE DOCUMENTS**

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

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

1. **delete the movie "Avatar"**

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



**RELATIONSHIPS**

Insert the following documents into a **users** collection

**username : GoodGuyGreg**

**first\_name : "Good Guy”**

**last\_name : "Greg"**

* db.users.insert({username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"})

**username : ScumbagSteve**

**full\_name :**

**first : "Scumbag"**

**last : "Steve"**

* db.users.insert({username:"GoodGuyGreg",fullname:{first\_name:"Good Guy",last\_name:"Greg"}})

Insert the following documents into a **posts** collection

**username : GoodGuyGreg**

**title : Passes out at party**

**body : Wakes up early and cleans house**

* db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})

**username : GoodGuyGreg**

**title : Steals your identity**

**body : Raises your credit score**

* db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

**username : GoodGuyGreg**

**title : Reports a bug in your code**

**body : Sends you a Pull Request**

* db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

**username : ScumbagSteve**

**title : Borrows something**

**body : Sells it**

* db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

**username : ScumbagSteve**

**title : Borrows everything**

**body : The end**

* db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

**username : ScumbagSteve**

**title : Forks your repo on github**

**body : Sets to private**

* db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})

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

**username : GoodGuyGreg**

**comment : Hope you got a good deal!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"**

* db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("60e6d777cb47f5158755df45")})

**username : GoodGuyGreg**

**comment : What's mine is yours!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"**

* db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("60e6d7f3cb47f5158755df46")})

**username : GoodGuyGreg**

**comment : Don't violate the licensing agreement!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github**

* db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("60e6d809cb47f5158755df47")})

**username : ScumbagSteve**

**comment : It still isn't clean**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"**

* db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("60e6d732cb47f5158755df42")})

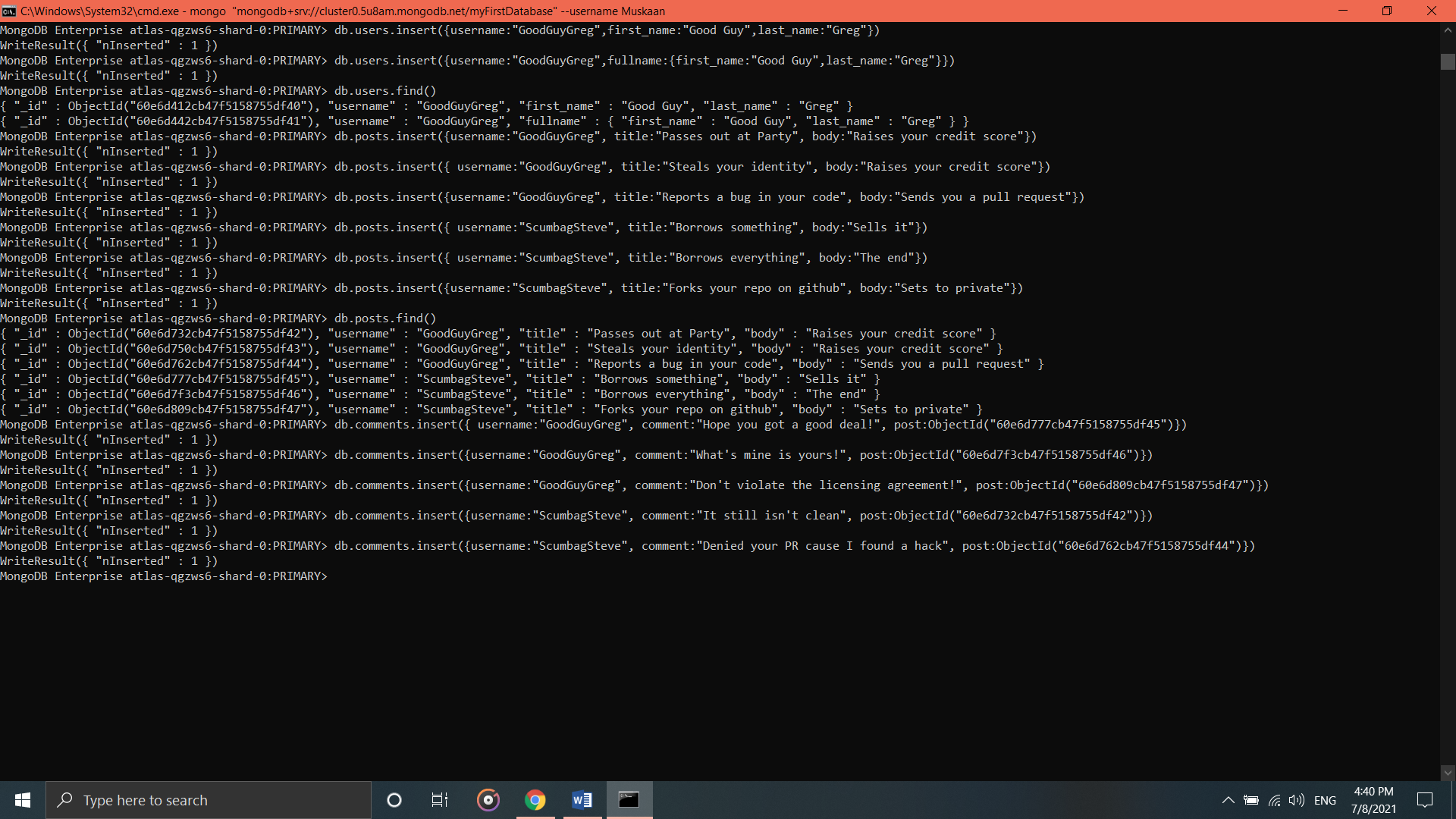
**username : ScumbagSteve**

**comment : Denied your PR cause I found a hack**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"**

* db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("60e6d762cb47f5158755df44")}



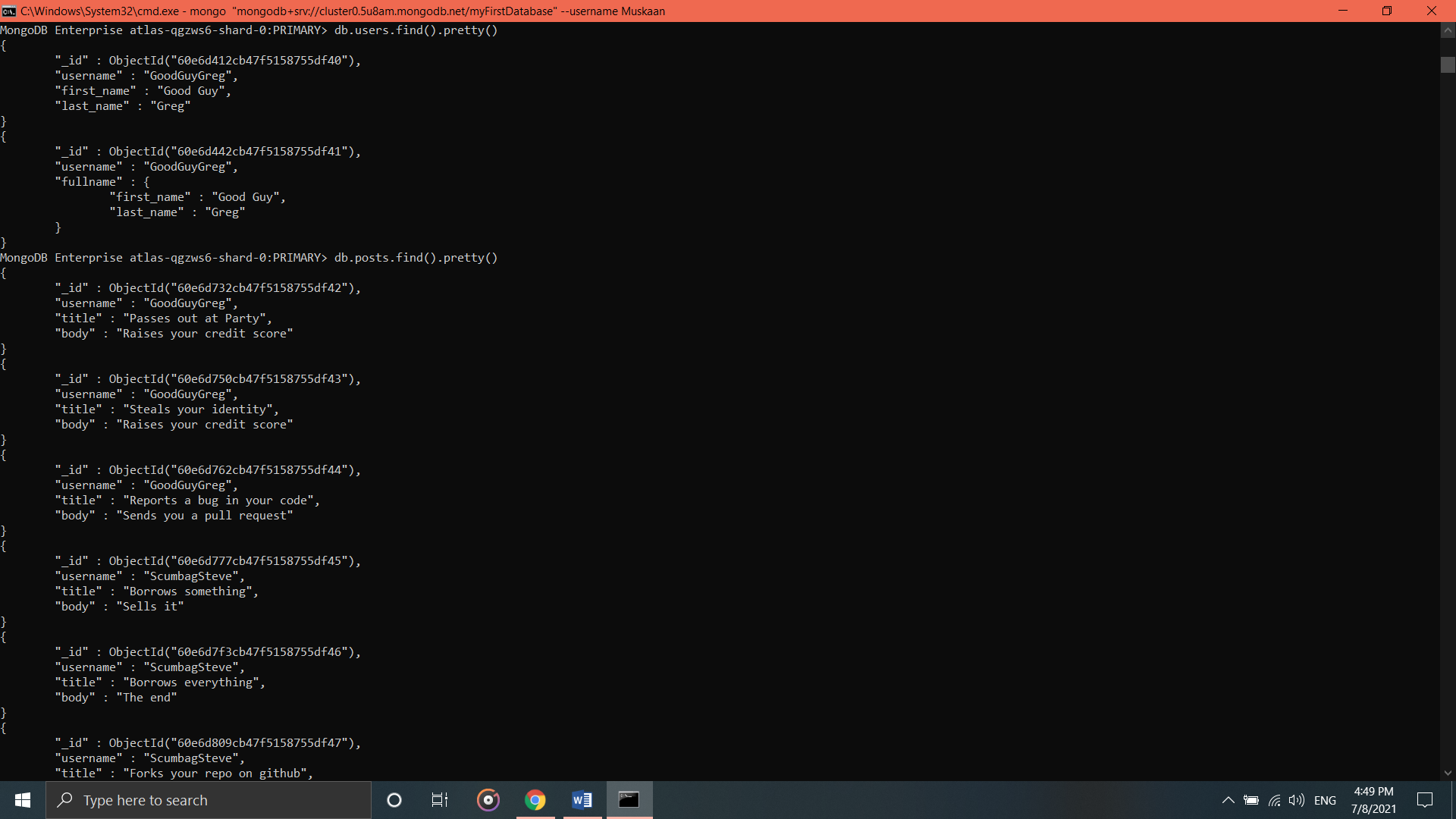
**Querying related collections**

1. find all users

* db.users.find().pretty()

1. find all posts

* db.posts.find().pretty()



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

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

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

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

1. find all comments

* db.comments.find().pretty()

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

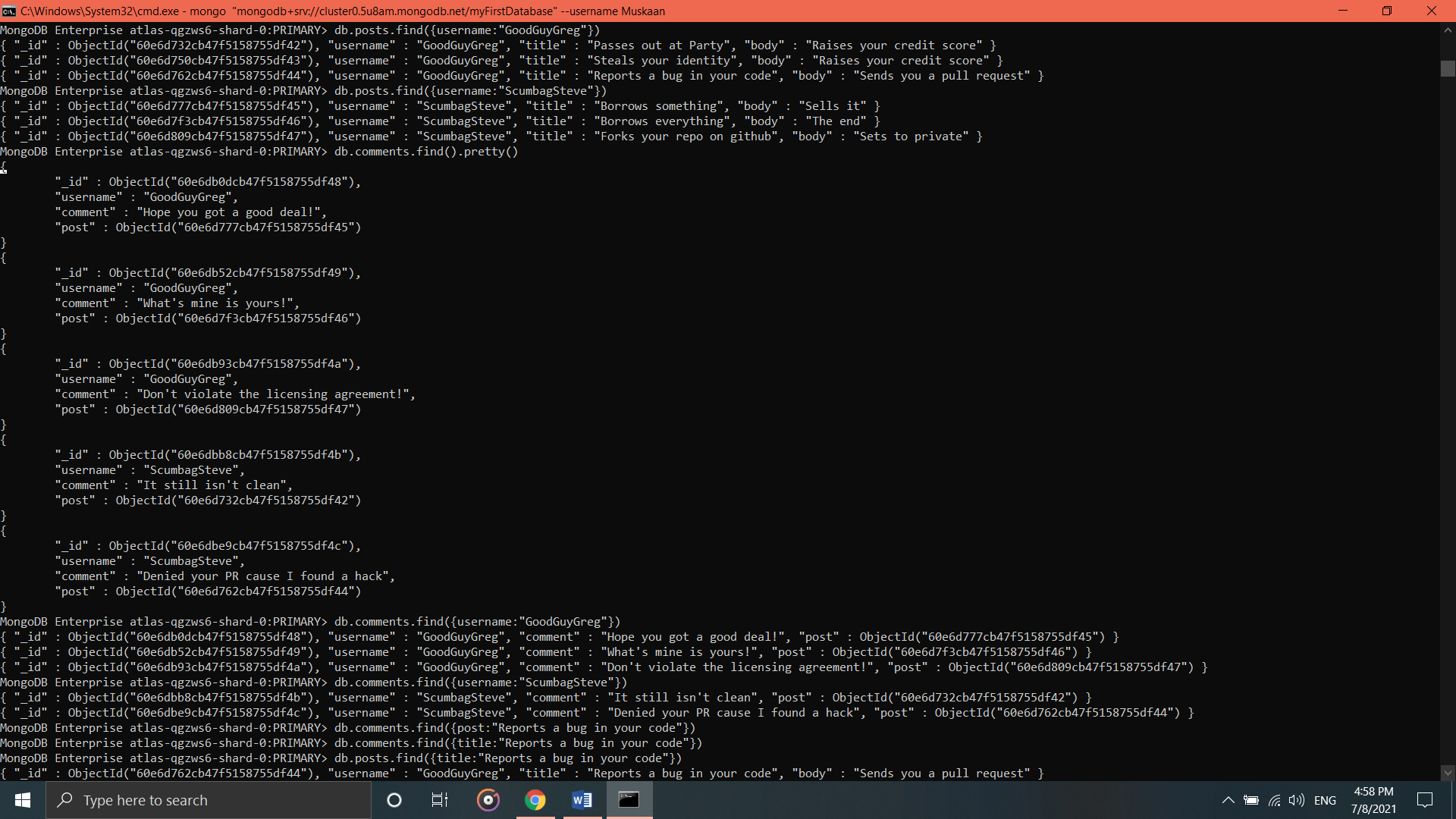
* db.comments.find({username:"GoodGuyGreg"})

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

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

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

* db.posts.find({title:"Reports a bug in your code"})



**MongoDB Aggregate Assignments**

**ATLANTA POPULATION**

1. **Use db.zipcodes.find() to filter results to only the results where city is ATLANTA and State is GA.**

* db.zipcodes.find({city:"ATLANTA"})

1. **Use db.zipcodes.aggregate with $match to do the same as above.**

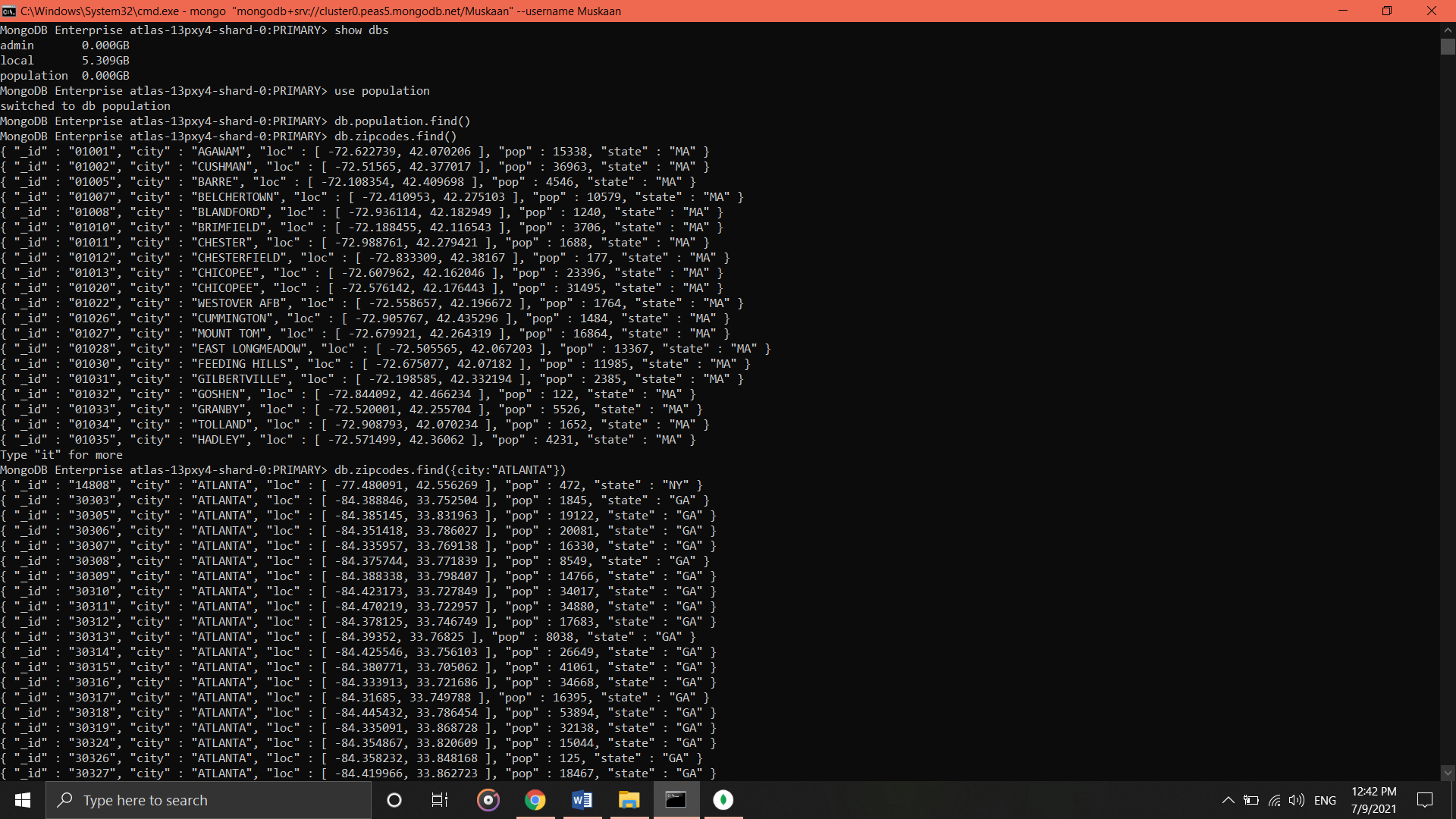
* db.zipcodes.aggregate([{$match:{city:"ATLANTA"}}])

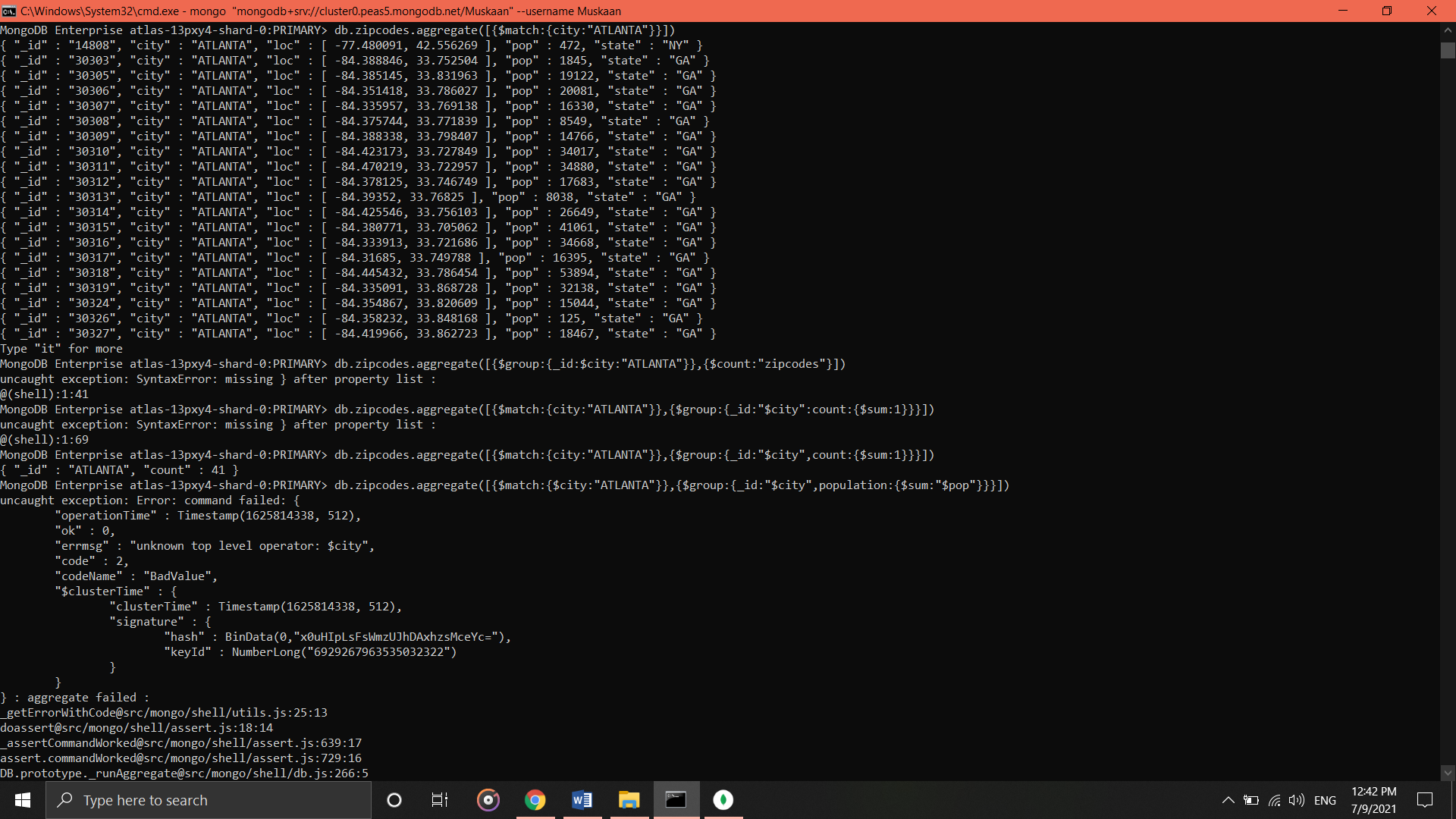
1. **Use $group to count the number of zip codes in Atlanta.**

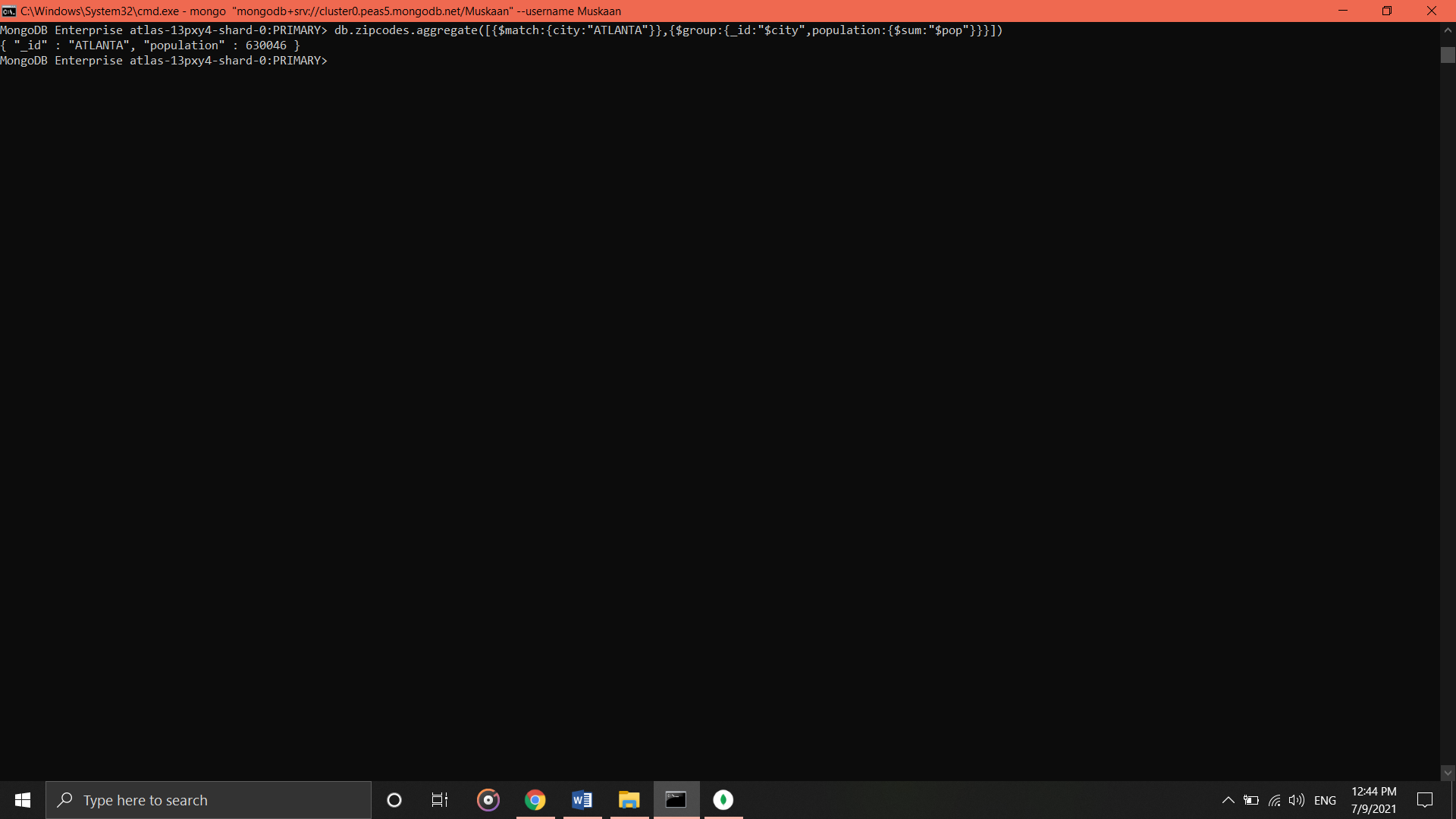
* db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",count:{$sum:1}}}])

1. **Use $group to find the total population in Atlanta.**

* db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",population:{$sum:"$pop"}}}])

****

****

****

**POPULATIONS BY STATE**

1. **Use aggregate to calculate the total population for each state.**

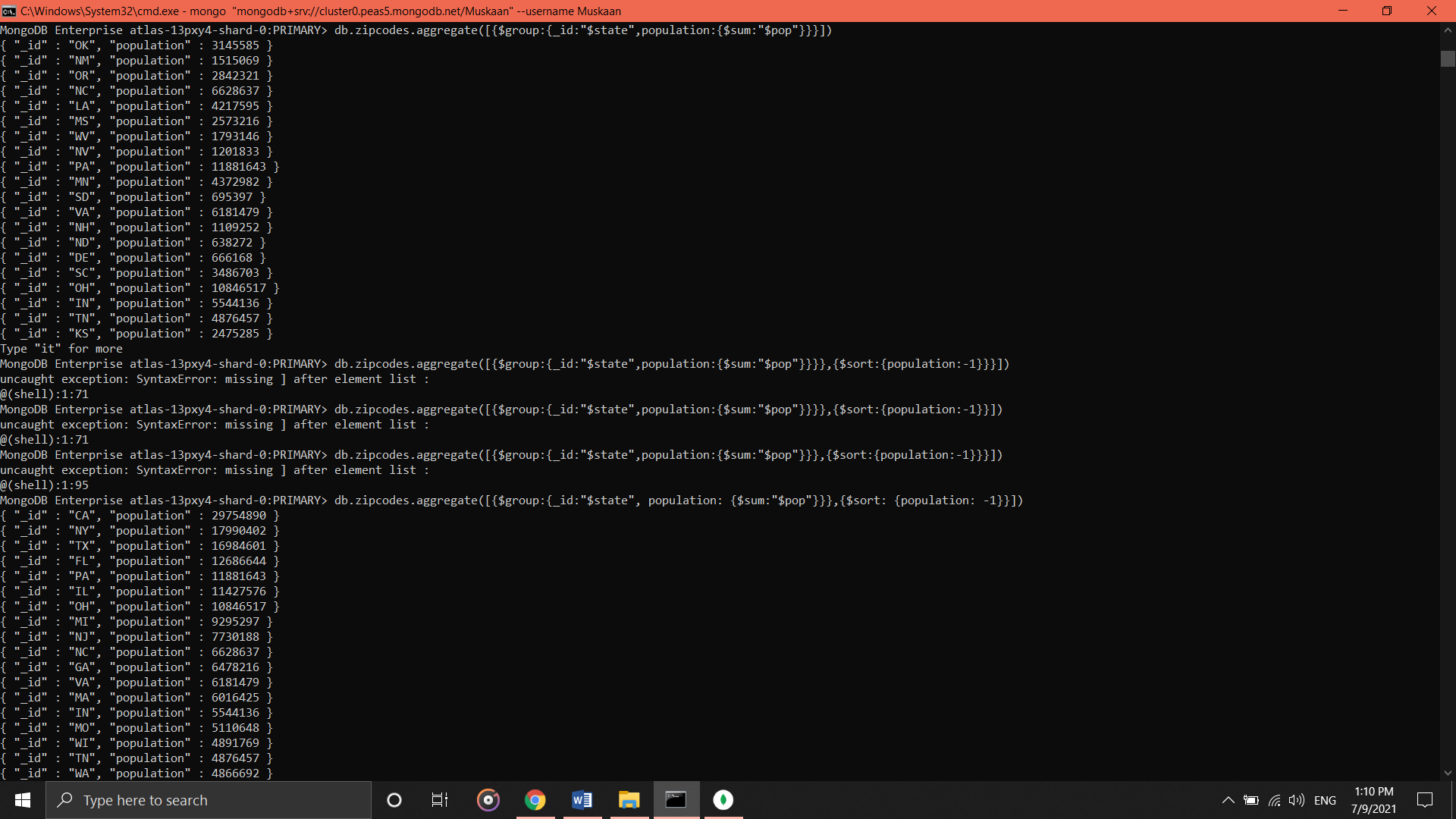
* db.zipcodes.aggregate([{$group:{\_id:”$state”,population:{$sum:”$pop”}}}])

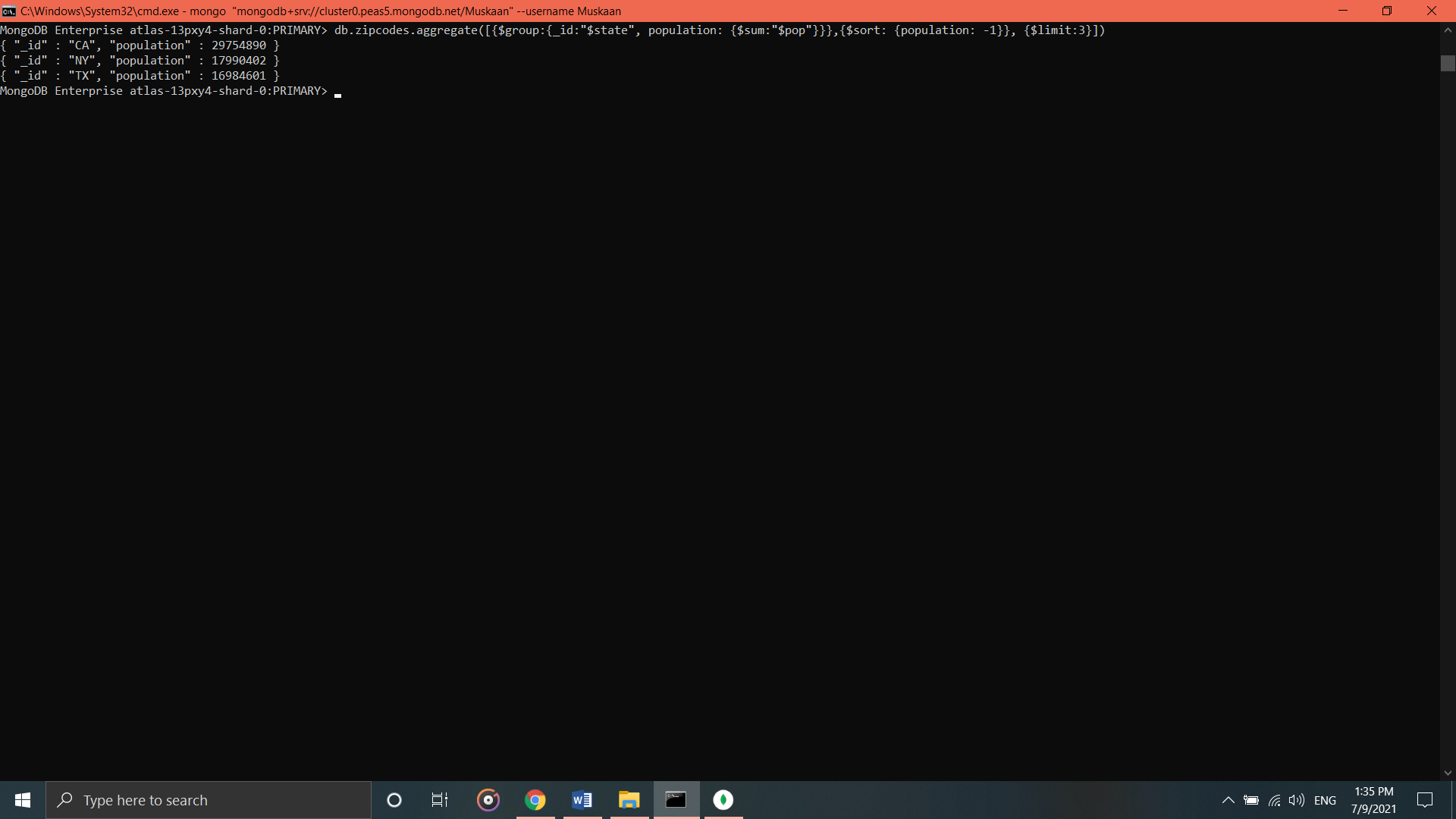
1. **Sort the results by population, highest first**

* db.zipcodes.aggregate([{$group:{\_id:"$state", population: {$sum:"$pop"}}},{$sort: {population: -1}}])

1. **Limit the results to just the first 3 results. What are the top 3 states in population?**

* db.zipcodes.aggregate([{$group:{\_id:"$state", population: {$sum:"$pop"}}},{$sort: {population: -1}}, {$limit:3}])

****

****

**POPULATIONS BY CITY**

1. **Use aggregate to calculate the total population for each city (you have to use city/state combination). You can use a combination for the \_id of the $group: {city:’$city’, state:’$state’}**

* db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}}])

1. **Sort the results by population, highest first**

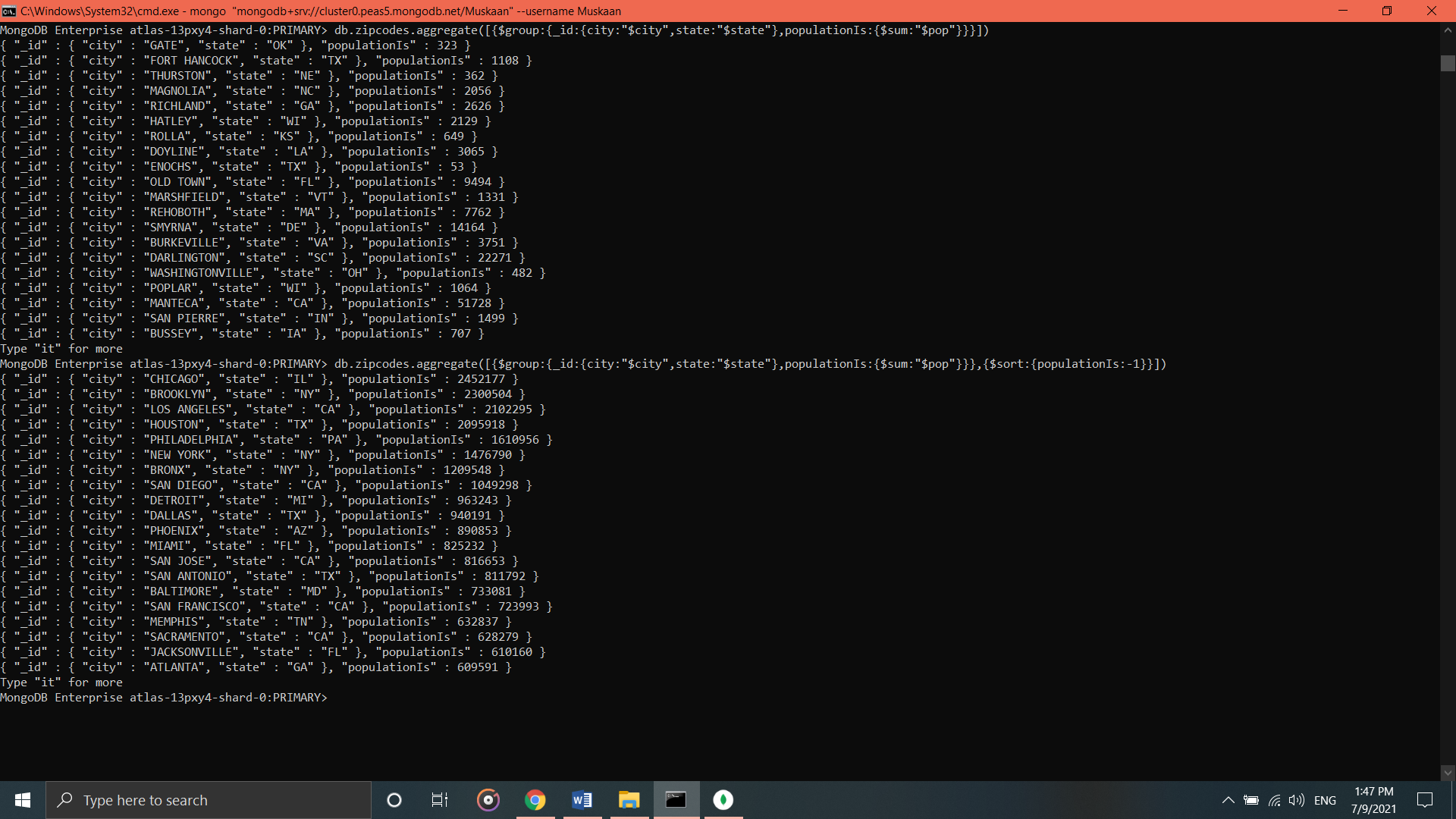
* db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}},{$sort:{populationIs:-1}}])

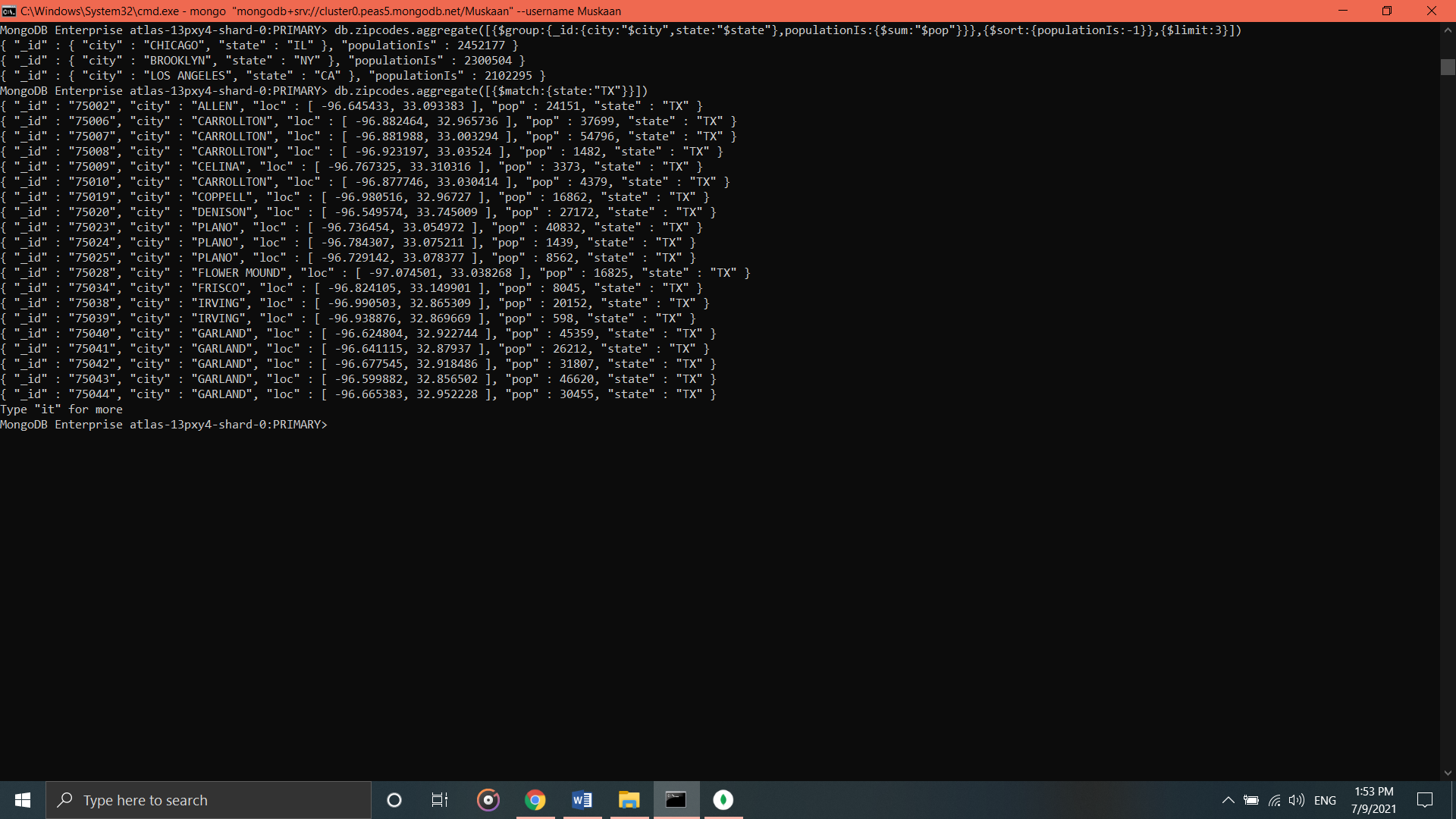
1. **Limit the results to just the first 3 results. What are the top 3 cities in population?**

* db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}},{$sort:{populationIs:-1}},{$limit:3}])

1. **What are the top 3 cities in population in Texas?**

* db.zipcodes.aggregate([{$match:{state:"TX"}}])

****

****

**BONUS**

1. **Write a query to get the average city population for each state.**

* db.zipcodes.aggregate([{$group: {\_id: {state: "$state"}, avgpopulationIs: {$avg: "$pop"}}}])

1. **What are the top 3 states in terms of average city population?**

* db.zipcodes.aggregate([{$group: {\_id: {state: "$state"}, avgpopulationIs: {$avg: "$pop"}}},{$sort:{avgpopulationIs:-1}},{$limit:3}])

