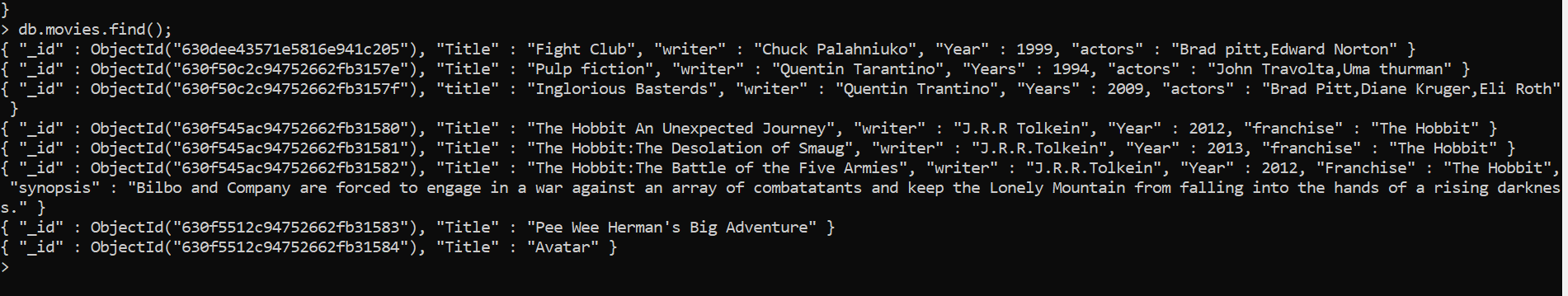
**MongoDB Assignments**

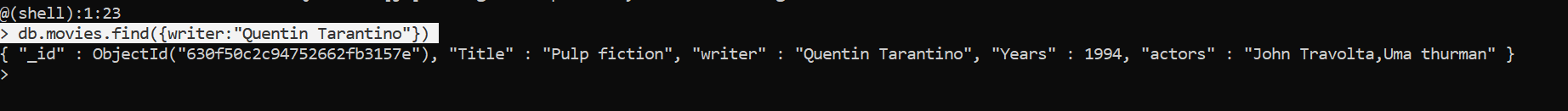
Assignment 1:

1.Insert Documents:

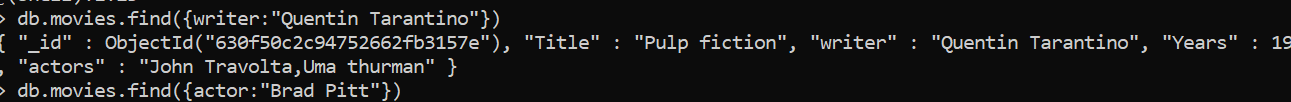
1. get all documents:



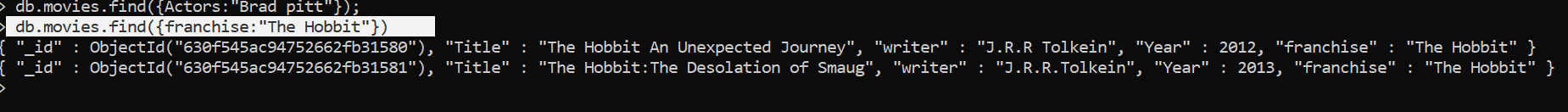
2. get all documents with writer set to "Quentin Tarantino":



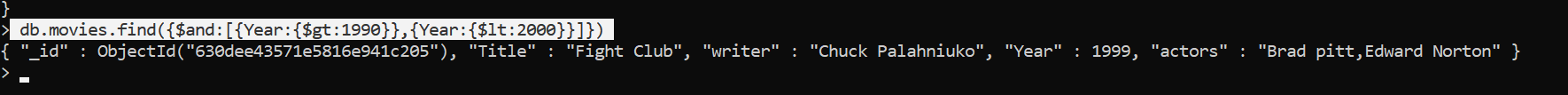
3. get all documents where actors include "Brad Pitt":



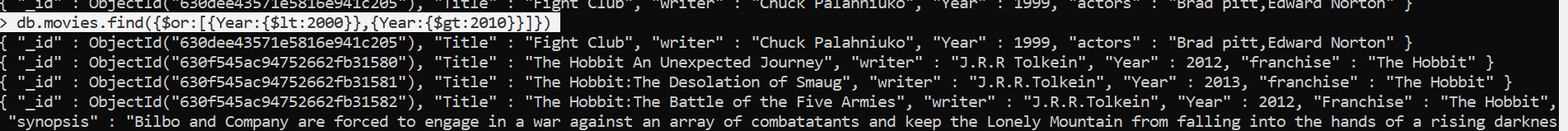
4. get all documents with franchise set to "The Hobbit":



5. get all movies released in the 90s:

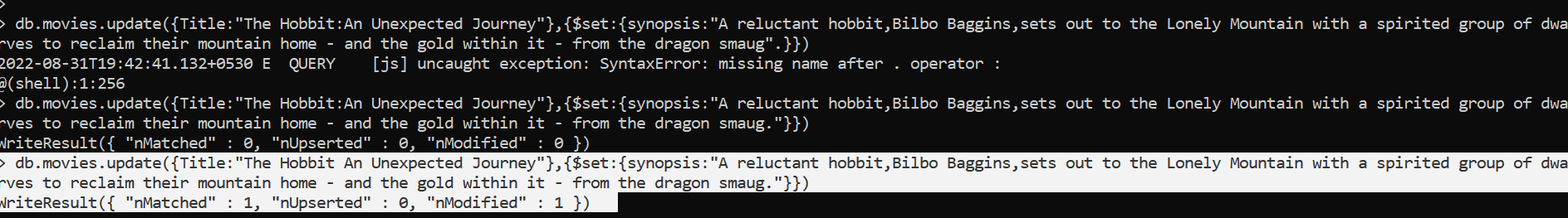


6.get all movies released before the year 2000 or after 2010:

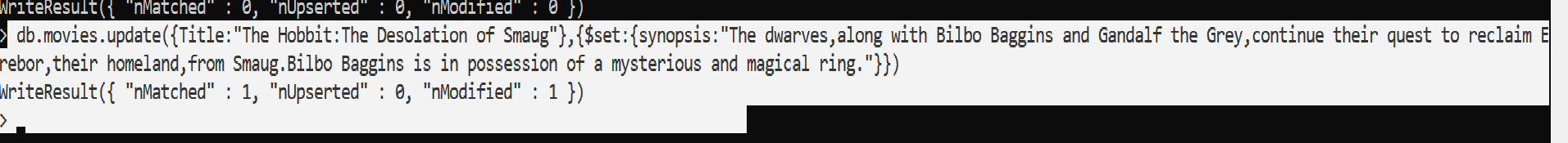


Update Documents:

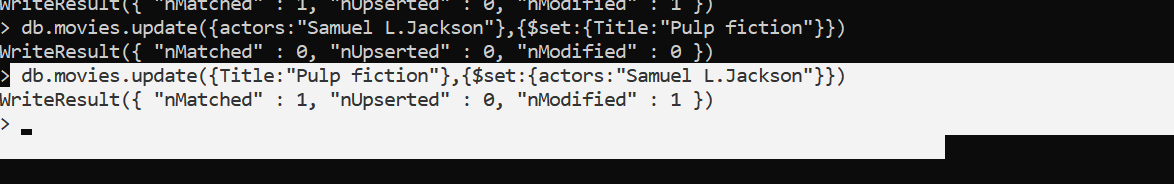
7. Add synopsis to The Hobbit: An Unexpected Journey : ”A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})



8. Add synopsis

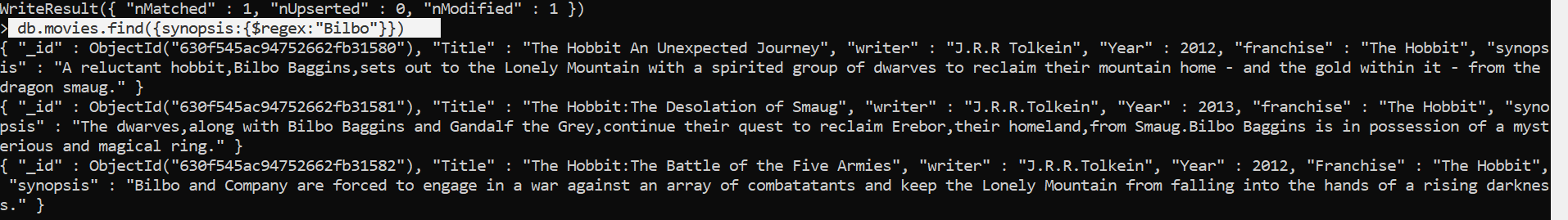


9.Add actor name:

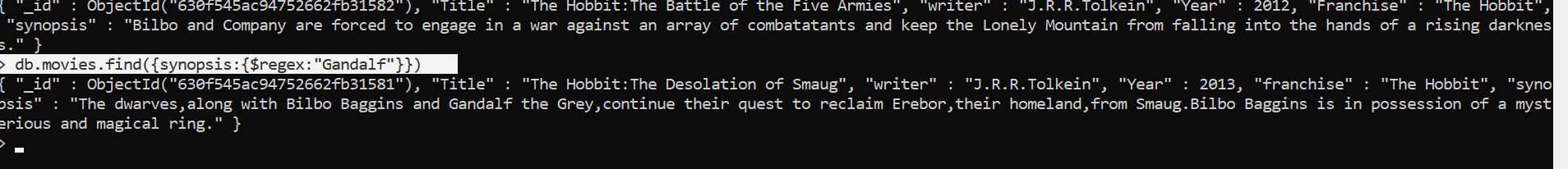


Text Search:

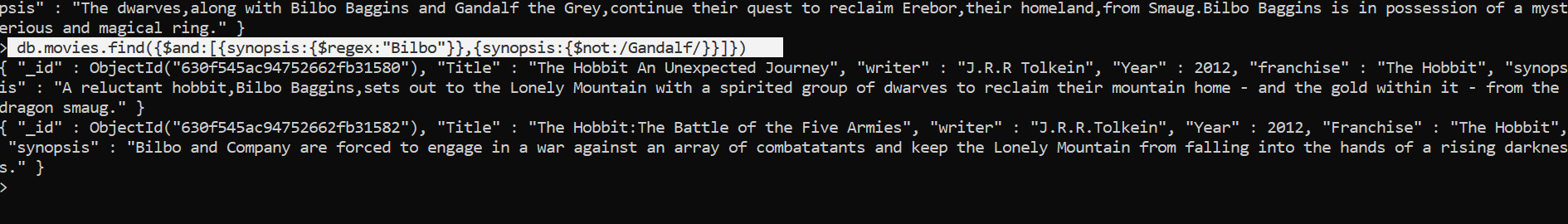
1.find all movies contains word “Bilbo”



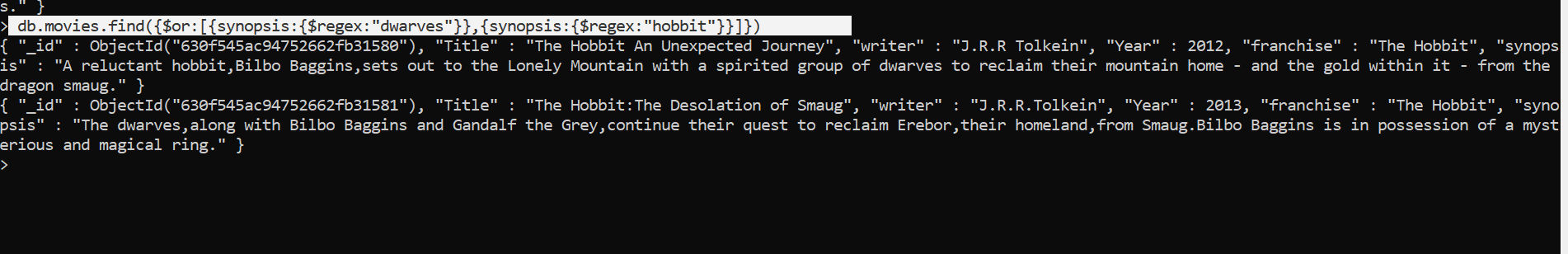
2.Find all movies contains word “Gandalf”



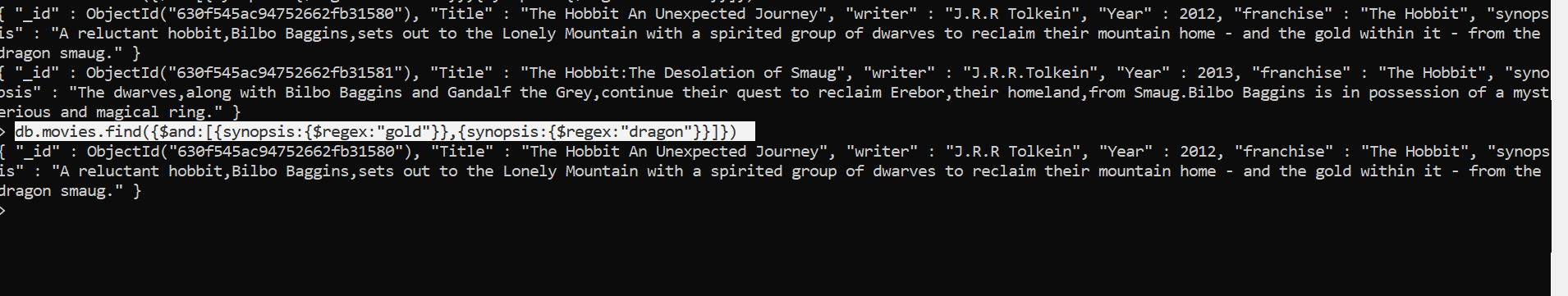
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"



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

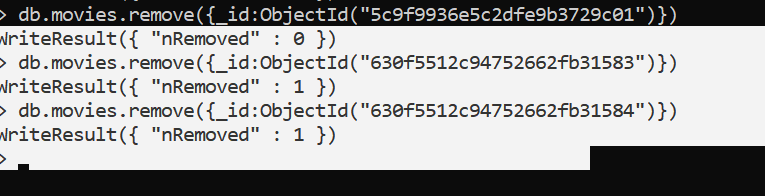


5.find all movies that have a synopsis that contains the word "gold" and "dragon":

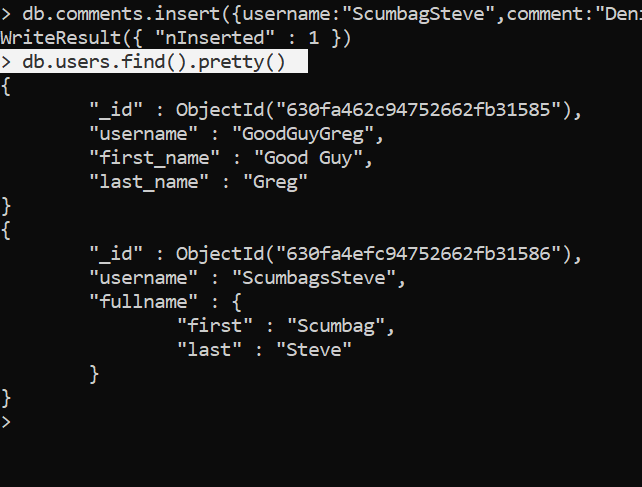


**Delete Documents:**

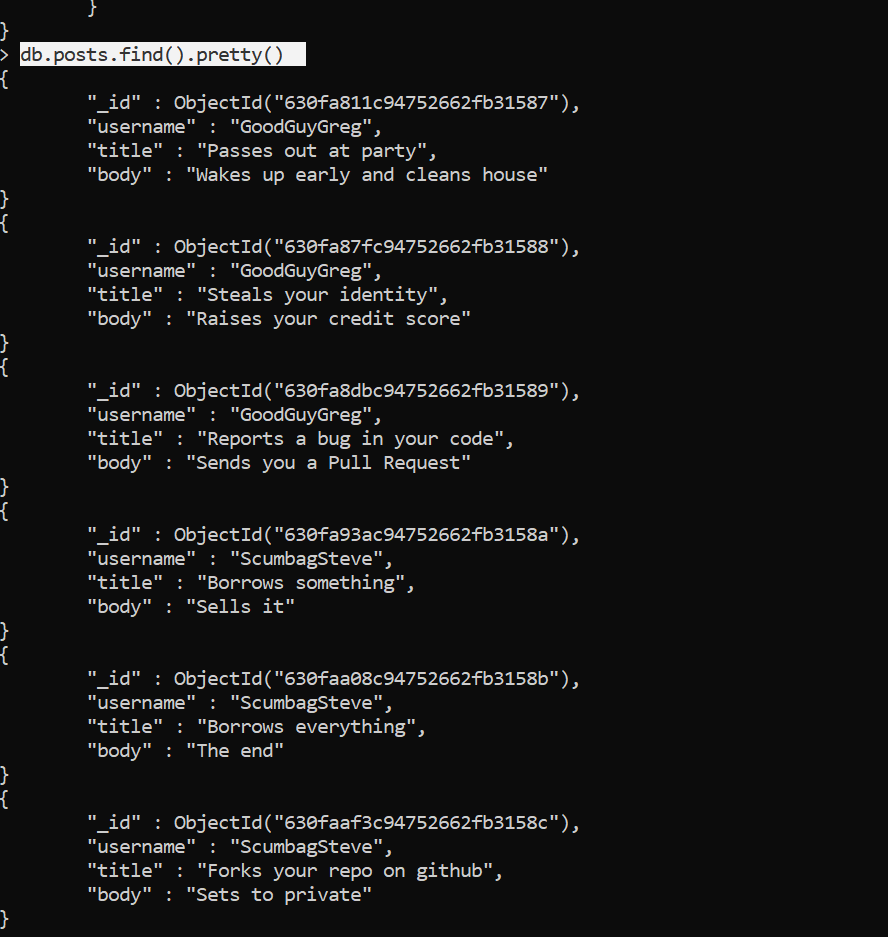
1 &2: delete the movie "Pee Wee Herman's Big Adventure" and “Avatar"



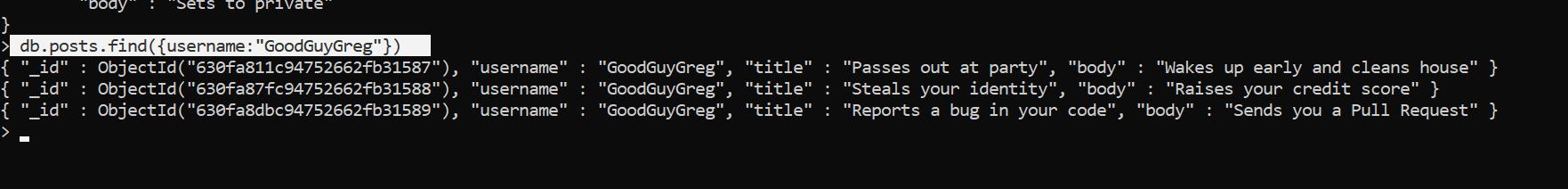
1. find all users



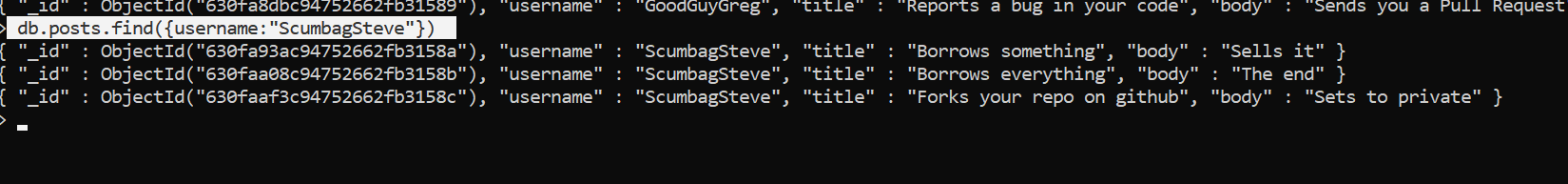
1. find all posts



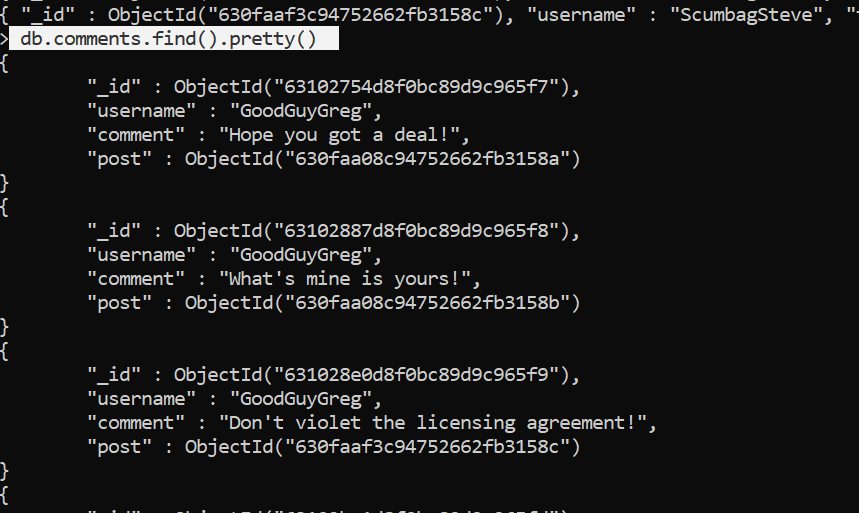
1. find all posts that was authored by "GoodGuyGreg"



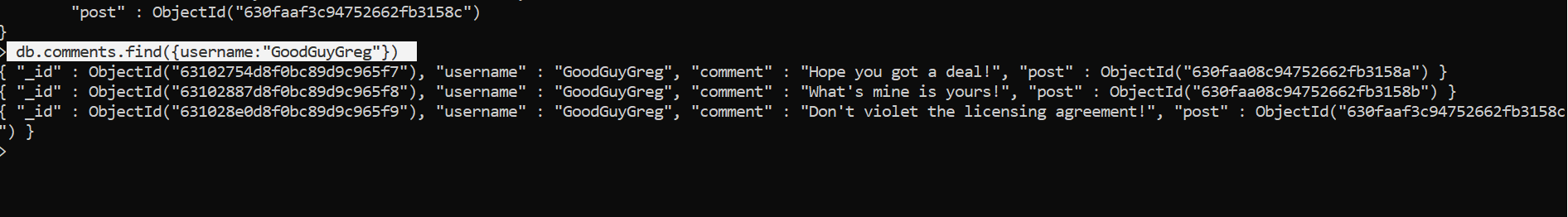
1. find all posts that was authored by "ScumbagSteve"



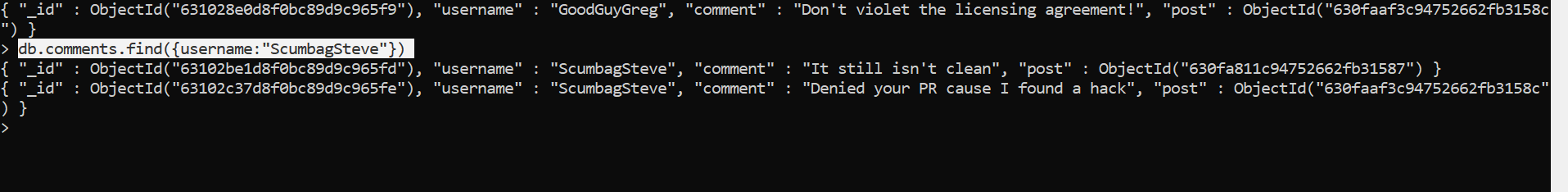
1. find all comments



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



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



Assignment 2:

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({$and:[{city: "ATLANTA"},{state: "GA"}]})**

1. Use db.zipcodes.Aggregate with $Match to do  the same as above

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

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

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

1. Use $group to find the total population in atlanta.

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

Population By State:

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

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

1. Sort the results by population , highest first

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

1. Sort the results by population, highest first

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

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

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

4.What are the top 3 cities in population in Texas?

Bonus:

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

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

1. What are the top 3 states in term of average  city population

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

Assignment 3:

1. Write a MongoDB query to display all the documents in the collection restaurants

**db.restaurants.find()**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant

**db.restaurants.find({},{restaurant\_id:1, name:1,borough:1, cuisine:1})**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

**db.restaurants.find({}, {restaurant\_id:1, \_id:0, name:1,borough:1,cuisine:1 })**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

**db.restaurants.find({}, {restaurant\_id:1, \_id:0, name:1,borough:1, "address.zipcode":1 })**

1. Write a MongoDB query to display all the restaurant which is in the borough Bronx.

**db.restaurants.find({"borough":"Bronx"})**

1. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

**db.restaurants.find({"borough":"Bronx"}).limit(5)**

1. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

db.restaurants.find({"borough":"Bronx"}).limit(5).skip(5)

1. Write a MongoDB query to find the restaurants who achieved a score more than 90

**db.restaurants.find({"grades.score":{$gt:90}})**

1. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

**db.restaurants.find({"grades.score":{$gt:80, $lt:100}})**

1. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

**db.restaurants.find({"address.coord":{$lt : -95.754168}})**

1. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168. :

**db.restaurants.find{$and:[ {"cuisine" : {$ne :"American "}},{"grades.score" : {$gt : 70}},{"address.coord" : {$lt : -65.754168}}]});**

1. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

**db.restaurants.find({"cuisine" : {$ne : "American "},"grades.score" :{$gt: 70},"address.coord" : {$lt : -65.754168}});**