**Homework 2**

**#Part 1:**

***A.*** *(1 mark) Create a collection called ‘games’. We’re going to put some games in it.*

db.createCollection("games")

***B.*** *(2 marks) Add 5 games to the database. Give each document the following properties:*

***name, genre, rating (out of 100)***

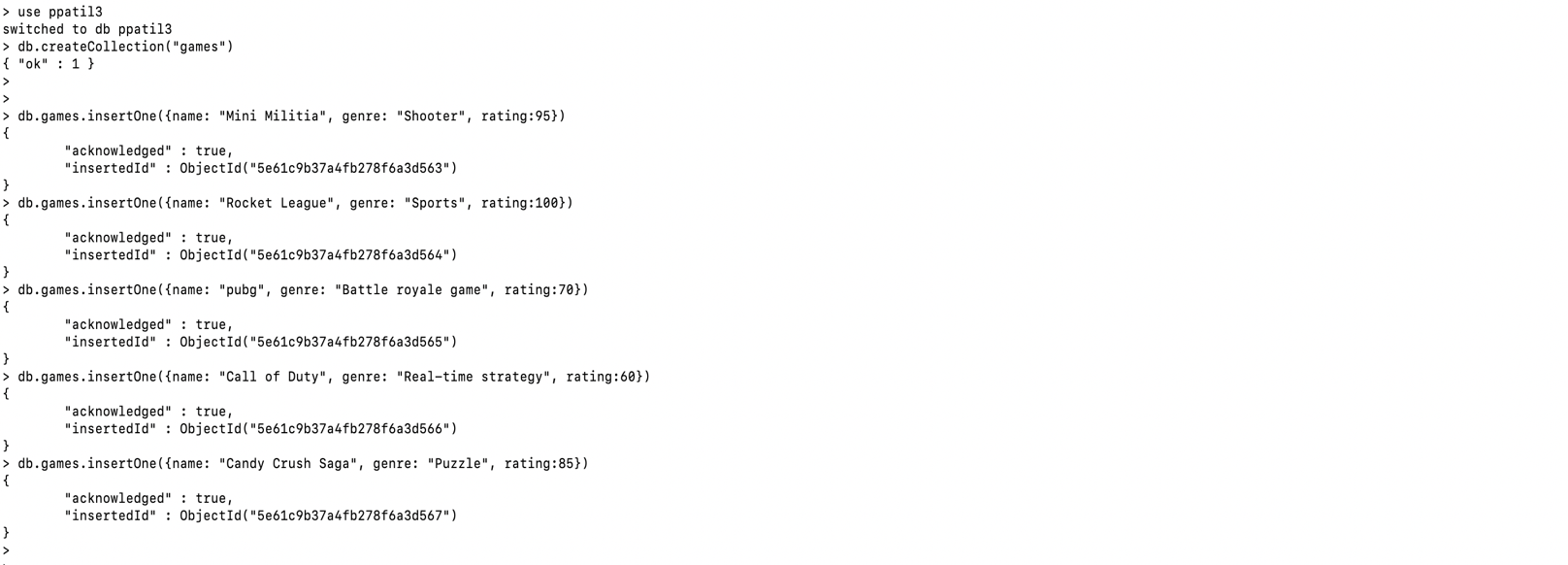
db.games.insertOne({name: "Mini Militia", genre: "Shooter", rating:95})

db.games.insertOne({name: "Rocket League", genre: "Sports", rating:100})

db.games.insertOne({name: "pubg", genre: "Battle royale game", rating:70})

db.games.insertOne({name: "Call of Duty", genre: "Real-time strategy", rating:60})

db.games.insertOne({name: "Candy Crush Saga", genre: "Puzzle", rating:85})



***C.*** *(1 mark) Write a query that returns all the games*

db.games.find().pretty()

***D.*** *(2 marks) Write a query to find one of your games by name without using limit().*

db.games.findOne({"name":"pubg"})



***E.*** *(3 marks) Write a query that returns the 3 highest rated games.*

db.games.find().sort({"rating":-1}).limit(3)

***F.*** *(6 marks) Update your two favourite games to have two achievements called ‘Game Master’ and ‘Speed Demon’, each under a single key. Show two ways to do this. Do the first using* ***update()*** *and do the second using* ***save().*** *Hint: for save, you might want to query the object and store it in a variable first.*

Update:

db.games.update({"name": "Mini Militia"},{$set: {"achievements": ["Game Master", "Speed Demon"]}})

Save:

var n=db.games.findOne({name:"Rocket League"})

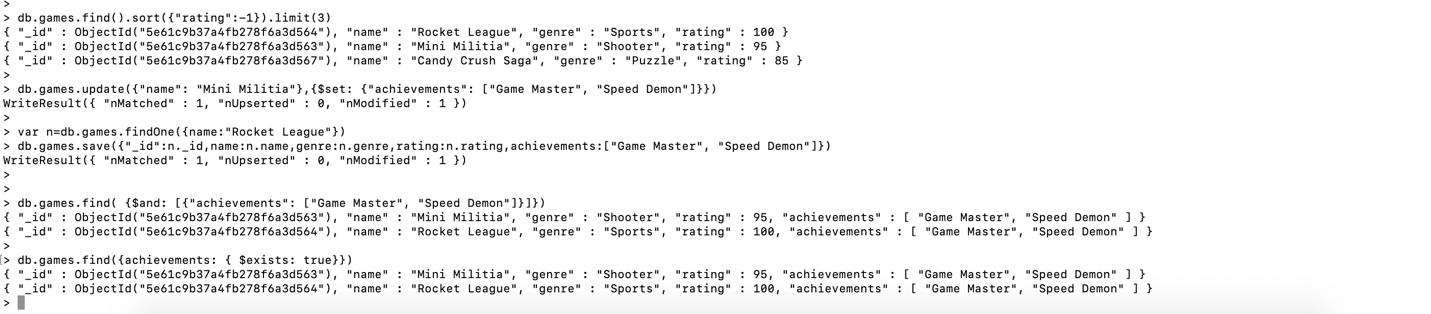
db.games.save({"\_id":n.\_id,name:n.name,genre:n.genre,rating:n.rating,achievements:["Game Master", "Speed Demon"]})

***G.*** *(2 marks) Write a query that returns all the games that have both the ‘  
Game Master’ and the ‘Speed Demon’ achievements.*

db.games.find( {$and: [{"achievements": ["Game Master", "Speed Demon"]}]})

***H.*** *(3 marks) Write a query that returns only games that have achievements. Not all of your games should have achievements, obviously.*

db.games.find({achievements: { $exists: true}})



**#Part 2:**

***Exercise 1:*** *(3 marks) Create 2 collections as #####\_Task1 and #####\_Task2, where ##### is your UMBC Username and upload these 2 json files to the collections on your MongoDB database.*

db.createCollection("ppatil3\_Task1")

db.createCollection("ppatil3\_Task2")

1. *(1 mark) Write a query that returns the total count of documents in #####\_Task2 collection.*

db.ppatil3\_Task2.find().count()

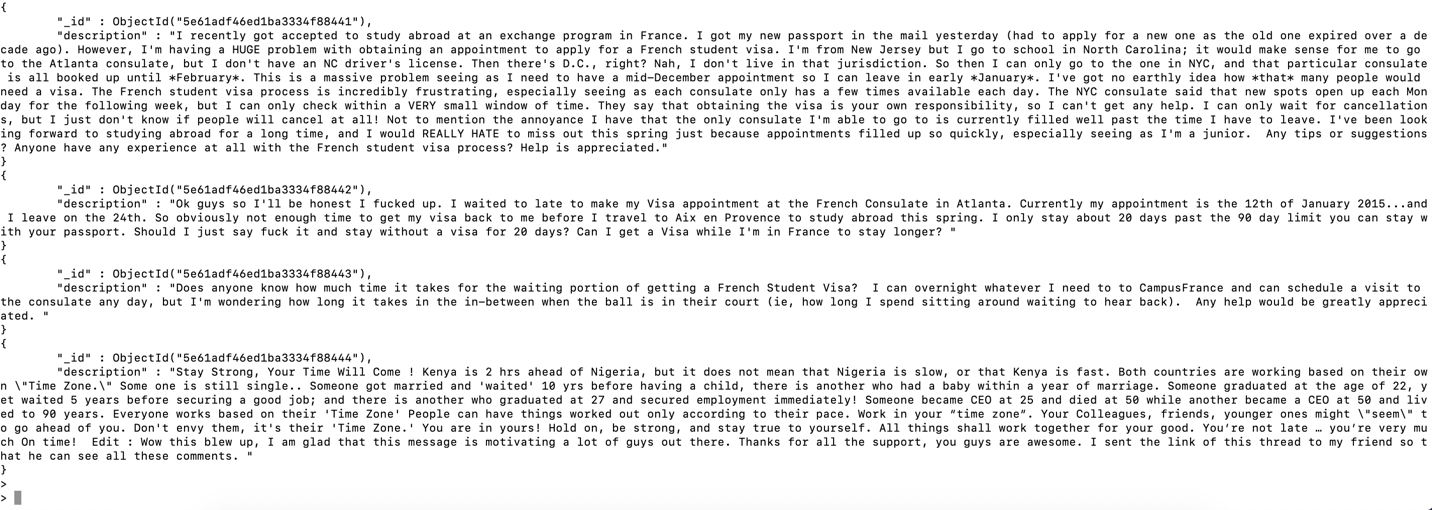
1. *(2 marks) Write a query that returns the count of documents in #####\_Task1 collection with the number of comments greater than 3.*

db.ppatil3\_Task1.find({num\_comments: {$gt: 3}}).count()

1. *(4 marks) Write a query to replace \n with space for the text in the description field for all the documents.*

db.ppatil3\_Task1.find({},{description:/\n/}).forEach(function(f) {f.description=f.description.replace(/\n/g, " "); printjson(f); });





**#Part 2:**

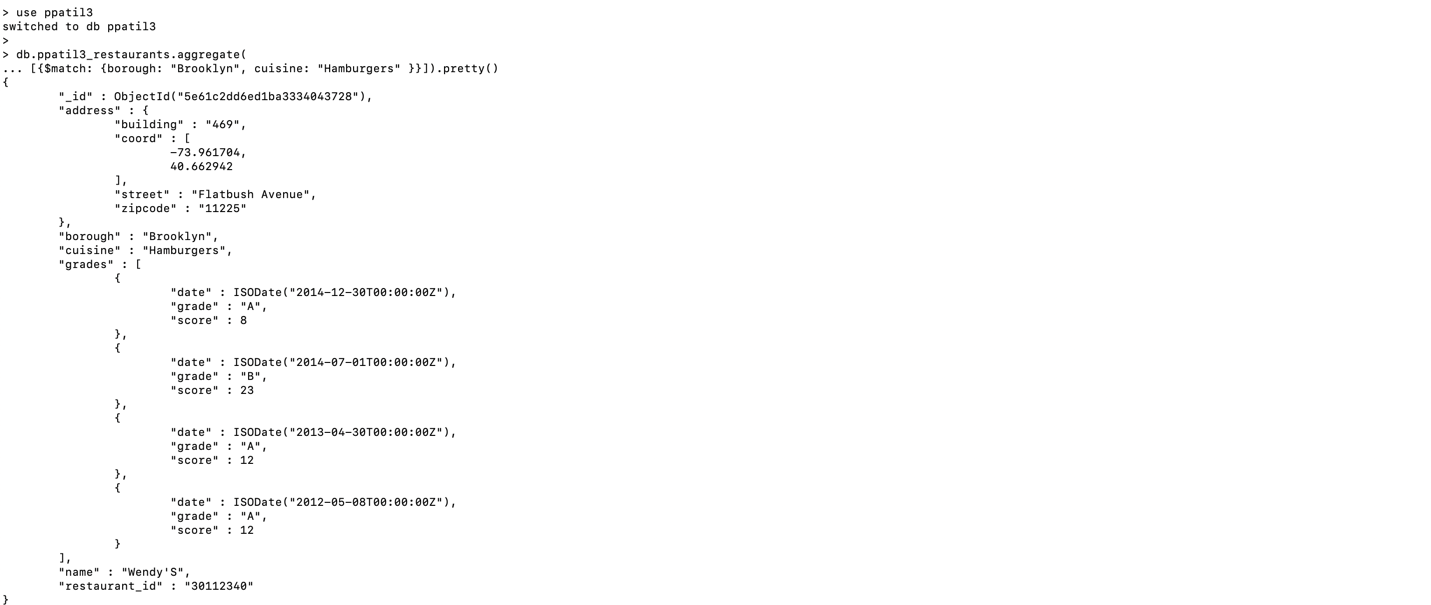
db.createCollection("ppatil3\_restaurants")

mongoimport --db ppatil3 --collection ppatil3\_restaurants --file /home/local/AD/ppatil3/primer-dataset.json

*1. (1 mark) All restaurants in borough (a town) "Brooklyn" and cuisine (a style of cooking) "Hamburgers".*

db.ppatil3\_restaurants.aggregate(

[{$match: {borough: "Brooklyn", cuisine: "Hamburgers" }}]).pretty()



*2. (1 mark) The number of restaurants in the borough "Brooklyn" and cuisine "Hamburgers".*

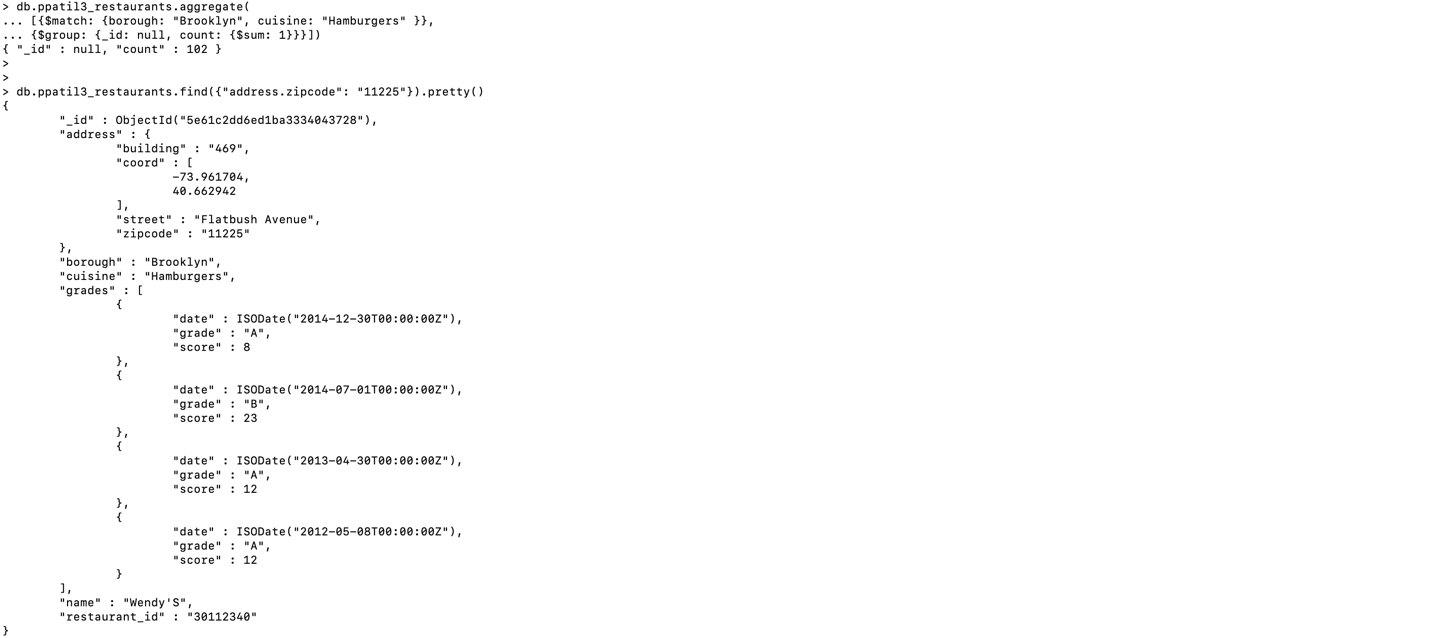
db.ppatil3\_restaurants.aggregate(

[{$match: {borough: "Brooklyn", cuisine: "Hamburgers" }},

{$group: {\_id: null, count: {$sum: 1}}}])

*3. (1 mark) All restaurants with zipcode 11225.*

db.ppatil3\_restaurants.find({"address.zipcode": "11225"}).pretty()



*4. (2 marks) Names of restaurants with zipcode 11225 that have at least one grade "C".*

db.ppatil3\_restaurants.find({$and: [{"address.zipcode": "11225"},{"grades.grade": { $in: ["C"]}}]}, {name: 1, \_id:0})

*5. (3 marks) Names of restaurants with zipcode 11225 that have as first grade "C" and as second grade "A".*

db.ppatil3\_restaurants.find({$and: [{"address.zipcode": "11225"},

{"grades.0.grade": "C"},{"grades.1.grade": "A"}]},{name: 1, \_id:0}).pretty()

*6. (2 marks) Names and streets of restaurants that don't have an "A" grade.*

db.ppatil3\_restaurants.find({"grades.grade": {$ne: "A"}}, {name: 1,

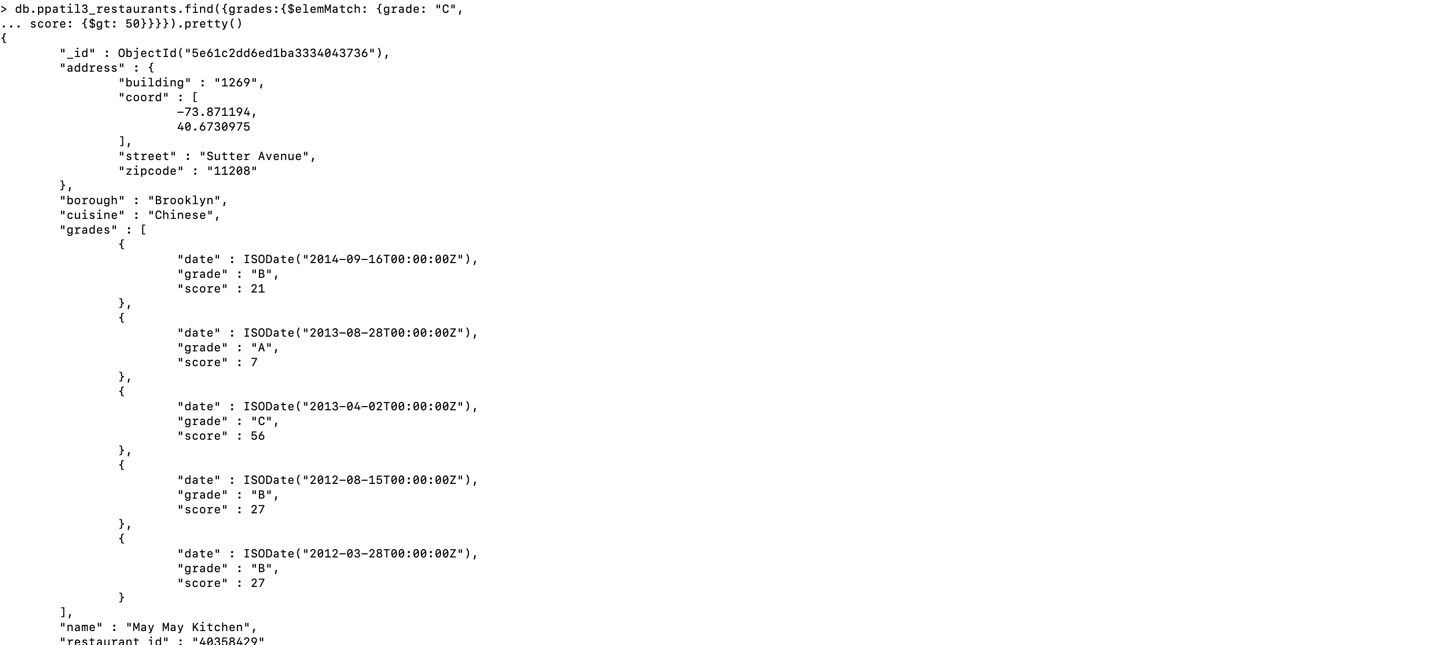
"address.street": 1, \_id: 0},{name: 1, "address.street": 1, \_id:0}).pretty()



*7. (3 marks) All restaurants with a grade C and a score greater than 50.*

db.ppatil3\_restaurants.find({grades:{$elemMatch: {grade: "C",

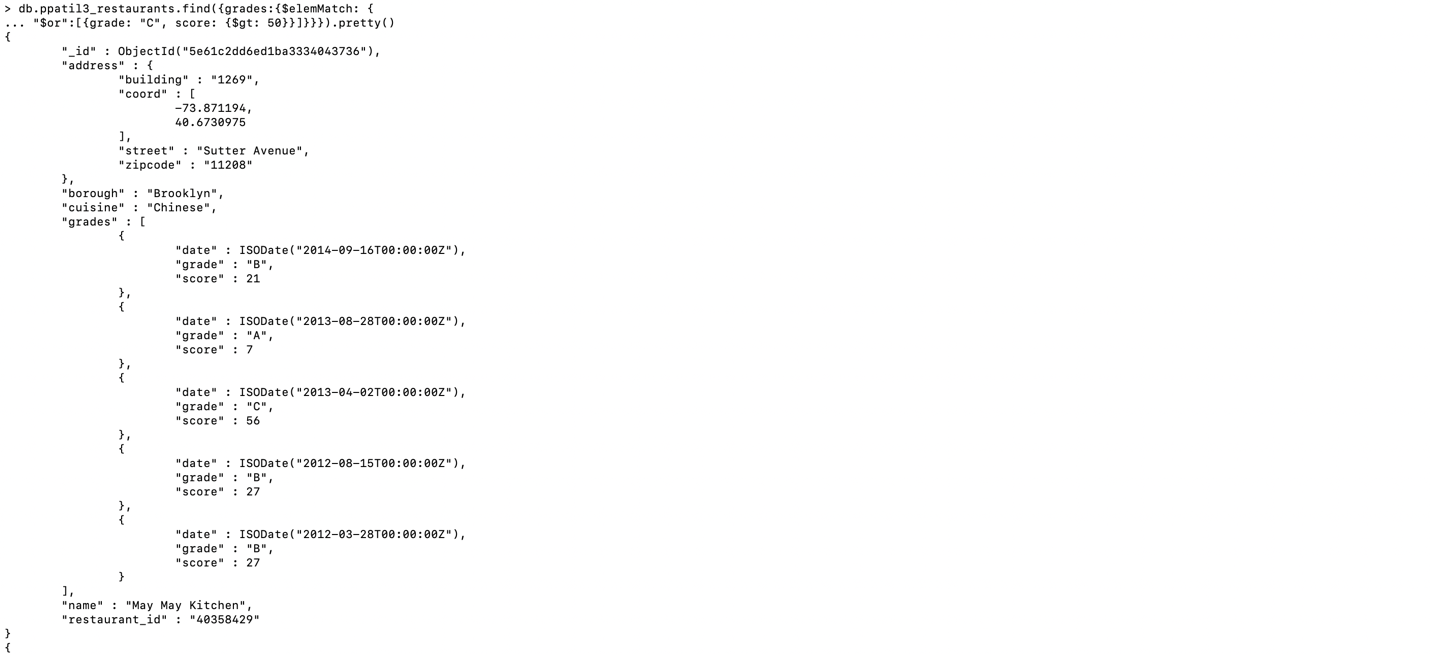
score: {$gt: 50}}}}).pretty()



*8. (3 marks) All restaurants with a grade C or a score greater than 50.*

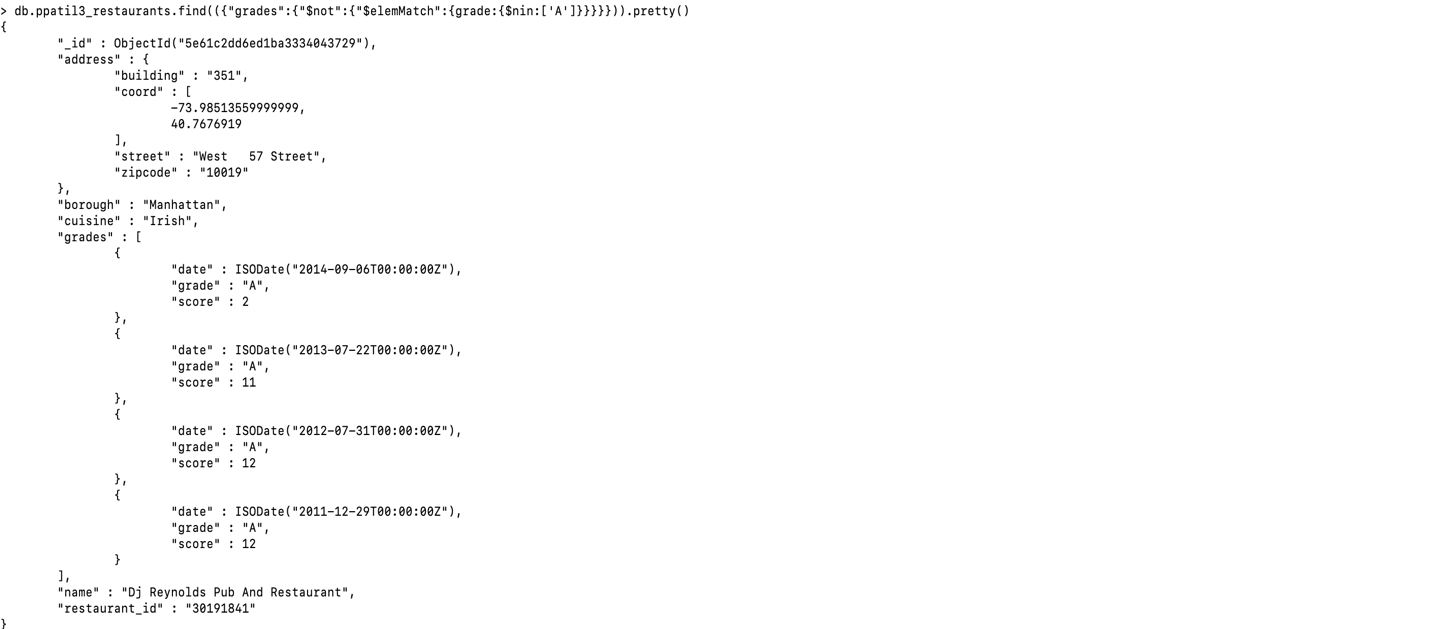
db.ppatil3\_restaurants.find({grades:{$elemMatch: {

"$or":[{grade: "C", score: {$gt: 50}}]}}}).pretty()



*9. (1 mark) All restaurants that have only A grades.*

db.ppatil3\_restaurants.find(({"grades":{"$not":{"$elemMatch":{grade:{$nin:['A']}}}}})).pretty()



*10. (3 marks) A table with zipcode and number of restaurants that are in the borough "Queens" and have "Brazilian" cuisine.*

db.ppatil3\_restaurants.aggregate([

{$match: {borough: "Queens", cuisine: "Brazilian"}},

{ "$group" : {\_id:"$address.zipcode", Number\_of\_restaurants: {$sum:1}}},

{$project: {\_id: 0, zipcode: "$\_id", Number\_of\_restaurants: 1}}]).pretty();

