**Assignment-1**

**Use mongo**\_practice

d.createCollection(‘movies’)

**Insert Documents**

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. Tokein',

'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.'

},

{

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

},

{

'title':'Avtar'

}])

**Query/Find Documents**

1. db.movies.find()
2. db.movies.find({ writer: 'Quentin Taranito'})
3. db.movies.find({ actors: ‘Brad Pitt’})
4. db.movies.find({ franchise: “The Hobbit”})
5. db.movies.find({year:{$lt: 1999, $gt:1990}})
6. db.movies.find({year:{$lt: 2010,$gt: 2000}})

**Update Documents**

1. db.movies.update({title:'The Hobbit: An Unexpected Journey'},{$set: {synopsis: 'A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug.'} })
2. db.movies.update({title:'The Hobbit: The desolation of Smaug'},{$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 magical ring.'} })
3. db.movies.update({title:'Pulp Fiction'},{$push:{actors: 'Samuel L. Jackson'}})

**Text Search**

db.movies.createIndex({synopsis:'text'})

1. db.movies.find({$text:{$search:'Bilbo'}})
2. db.movies.find({$text:{$search:'Gandalf'}})
3. db.movies.find({$text:{$search:'Bilbo -Gandalf'}})
4. db.movies.find({$text:{$search:'dwarves hobbit'}})
5. db.movies.find({$text:{$search:"'gold' 'dragon'"}})

**Delete Documents**

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

**Relationships**

**Insert documents of users collections**

db.createCollection('users')

db.users.insertMany([

{ username: ‘GoodGuyGreg’,

‘first\_name’:"Good Guy",

‘last\_name’:"Greg",

},

{ ‘username’: ‘ScumbagSteve’,

‘full\_name’:

{‘first\_name’:"Scumbag",

‘last\_name’:"Steve",

} } ])

**Insert documents of posts collections**

db.createCollection(‘posts’)

db.posts.insertMany([{

'username':'GoodGuyGreg',

'title':'Passes out at party',

'body':'Wakes up early and cleans house',

},

{ 'username':'GoodGuyGreg',

'title':'Steals your identity',

'body':'Raises your credit score',

},

{ 'ussername':'GoodGuyGreg',

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

'body':'Sends you a Pull Request',

},

{ 'username':'ScumbagSteve',

'title':'Borrows something',

'body':'Sells it',

},

{ 'username':'ScumbagSteve',

'title':'Borrows everything',

'body':'The end',

},

{ 'username':'ScumbagSteve',

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

'body':'Sets to private',

}])

**Insert documents of comments collections**

db.createCollection(‘comments’)

db.comments.insertMany([{

'username':'GoodGuyGreg',

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

'post':['617aa9cc6b910c064c32c339'],

},

{ 'username':'GoodGuyGreg',

'comment':"What's mine is yours!",

'post':['617aa9cc6b910c064c32c33a'],

},

{ 'username':'GoodGuyGreg',

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

'post':['617aa9cc6b910c064c32c33b'],

},

{ 'username':'ScumbagSteve',

'comment':"It still isn't clean",

'post':['617aa9cc6b910c064c32c336'],

},

{ 'username':'ScumbagSteve',

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

'post':['617aa9cc6b910c064c32c338'],

}])

**Querying related collections**

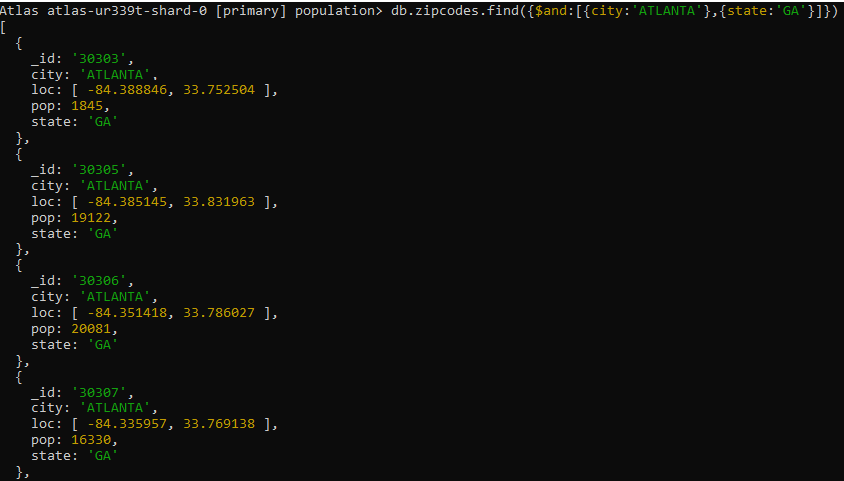
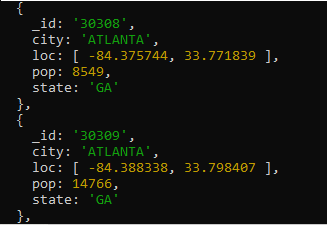
1. db.users.find()
2. db.posts.find()
3. db.posts.find({userusername:'GoodGuyGreg'})
4. db.posts.find({userusername:'ScumbagSteve'})
5. db.comments.find()
6. db.comments.find({uusername:'GoodGuyGreg'})
7. db.comments.find({username:'ScumbagSteve'})
8. db.posts.find({title:'Reports a bug in your code'})

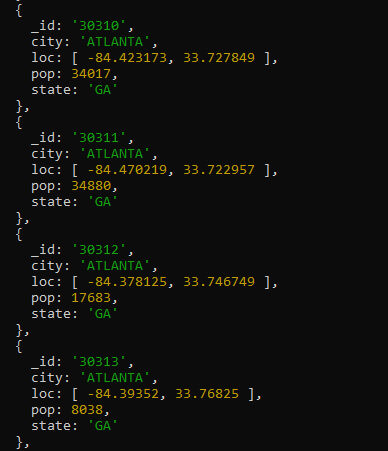
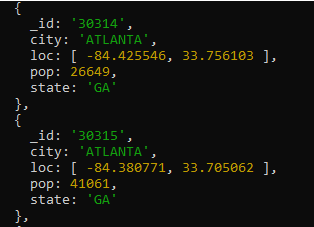
**Assignment-2**

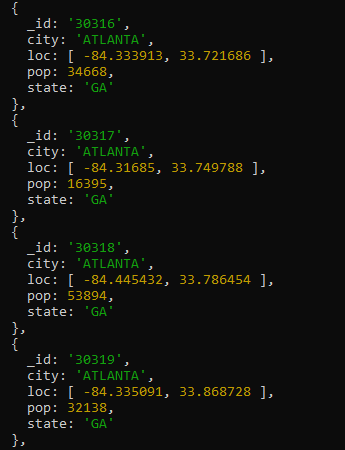
Use population

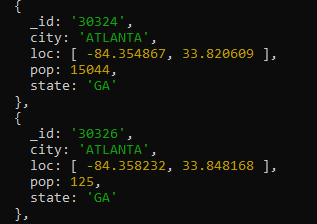
Db.createCollection(‘zipcodes’)

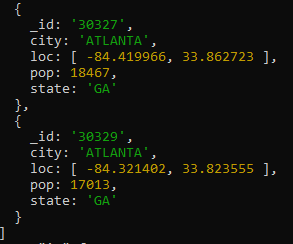
**Atlanta Population**

1. use db.zipcodes.find() to filter results to only the results where city is ATLANTA and state is GA. 

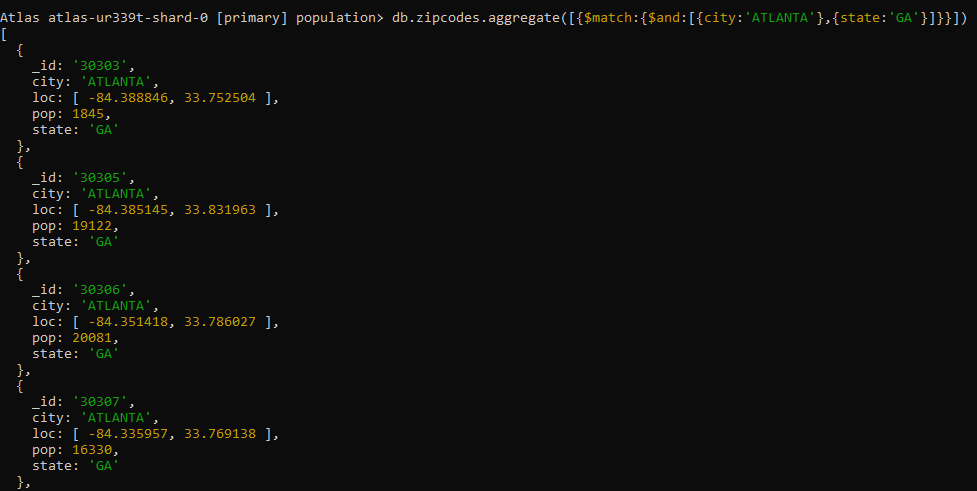
****

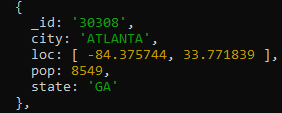
****

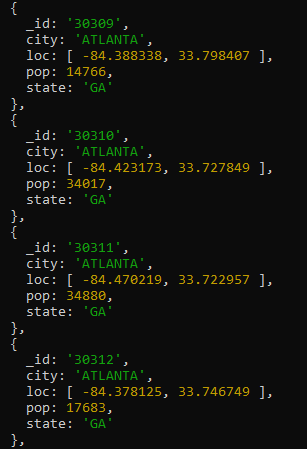
****

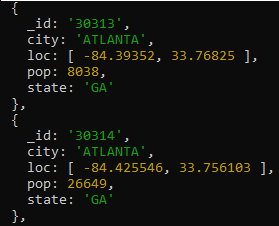
****

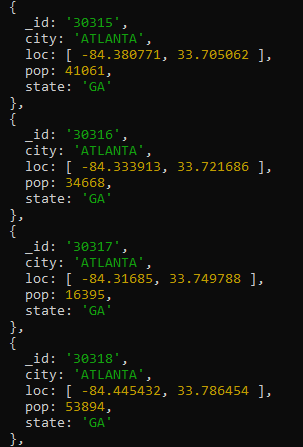
1. use db.zipcodes.aggregate with $match to do the same as above.

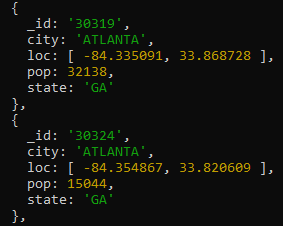
****

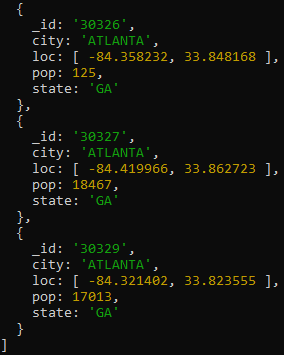
****

****

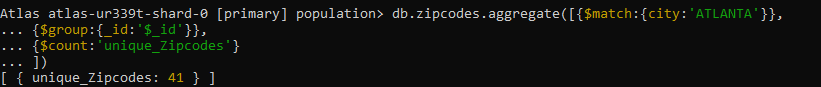
****

****

****

****

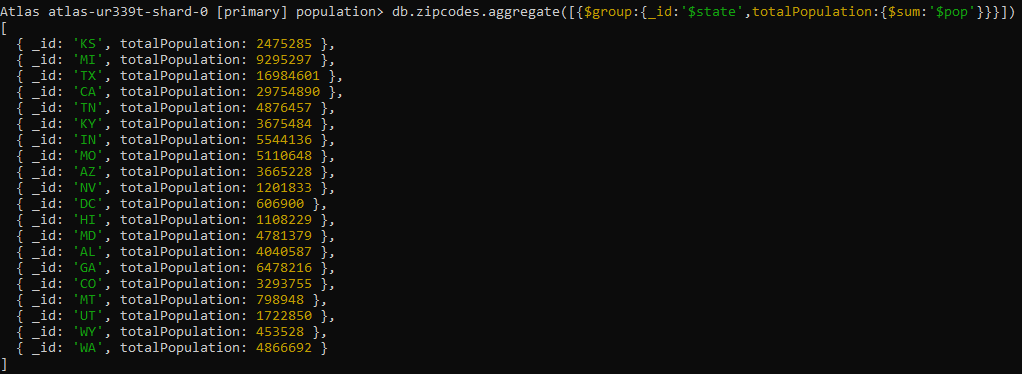
1. use $group to count the number of zip codes in Atlanta.



1. use $group to find the total population in Atlanta.

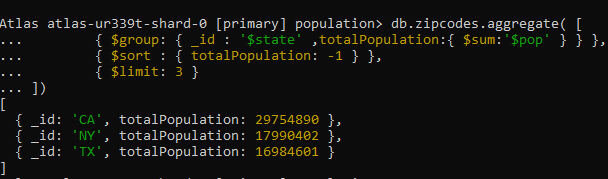


**Populations By State**

1. Use aggregate to calculate the total population for each state
2. sort the results by population, highest first

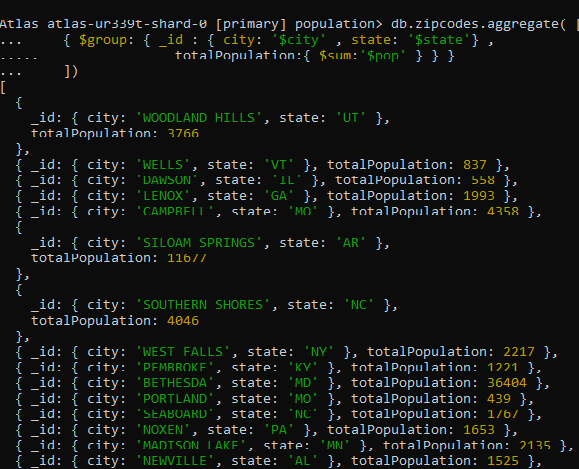


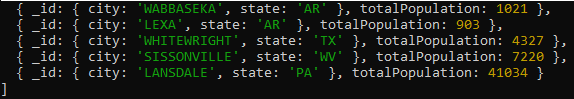
1. limit the results to just the first 3 results. What are the top 3 states in population?



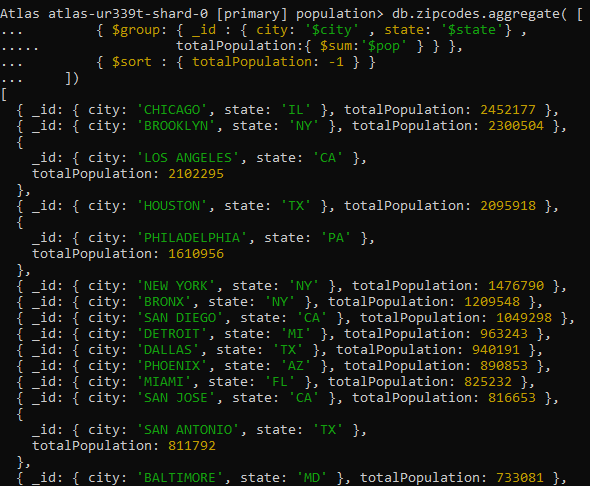
**Populations by City**

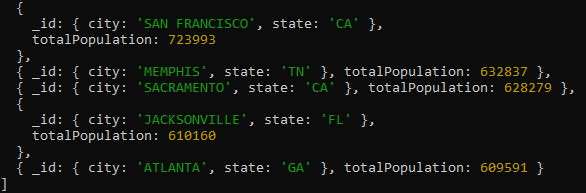
1. use aggregate to calculate the total population for each city (you have to use city/state combination). You can use a combination for the \_id of the $group: { city: '$city', state: '$state' }



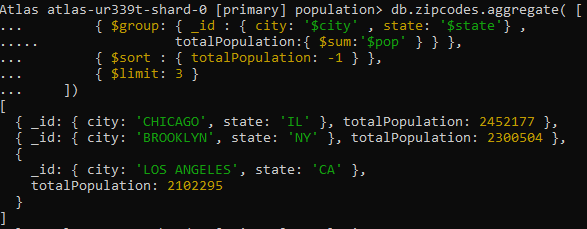


1. sort the results by population, highest first

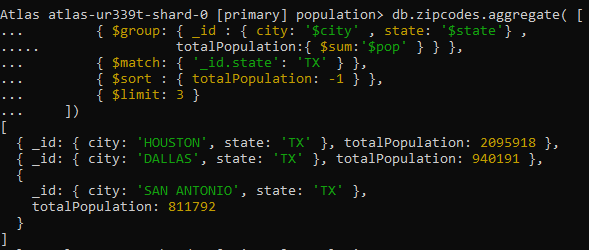




1. limit the results to just the first 3 results. What are the top 3 cities in population?

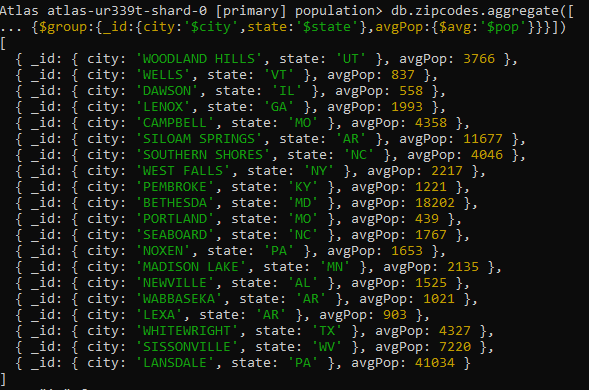


1. What are the top 3 cities in population in Texas?

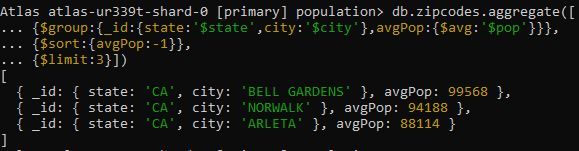


**Bonus**

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

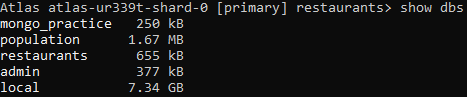


1. What are the top 3 states in terms of average city population?



**Assignment-3**

1. show databases



1. use restaurants



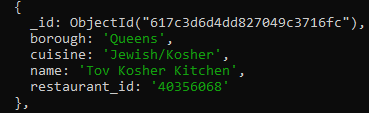
**Exercise Questions**

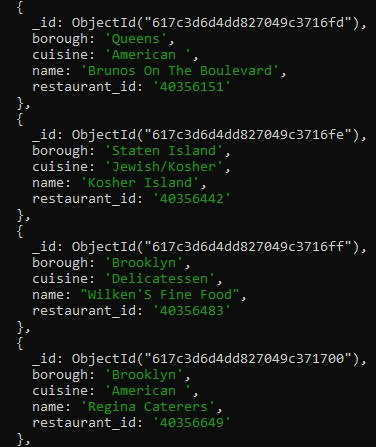
1. Write a MongoDB query to display all the documents in the collection restaurants.

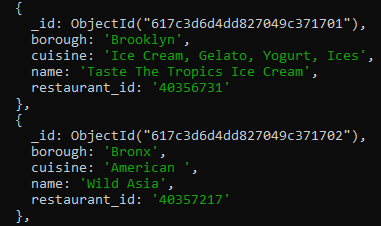
db.addresses.find()

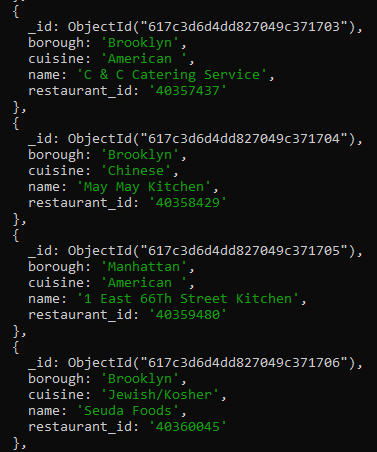
1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.

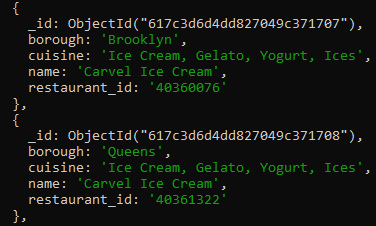






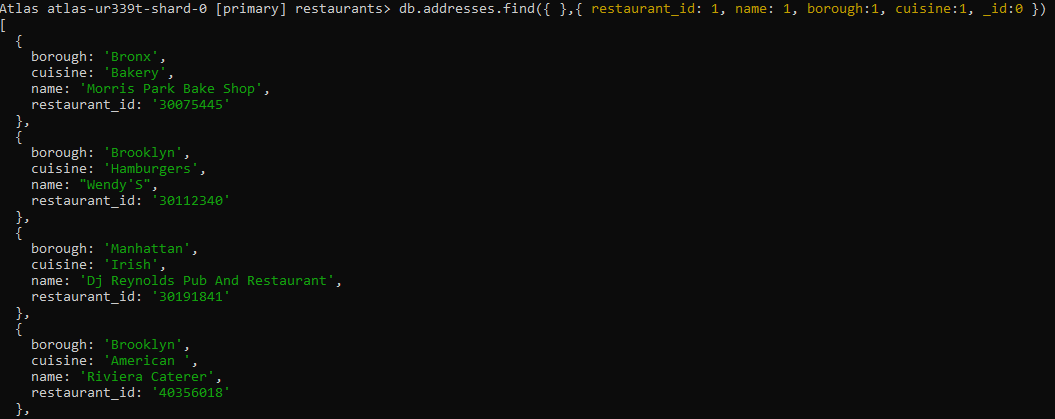


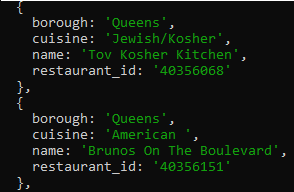


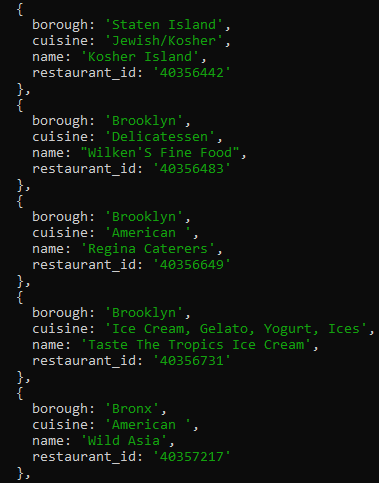


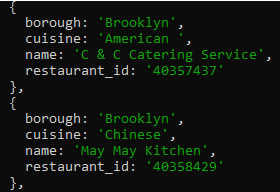


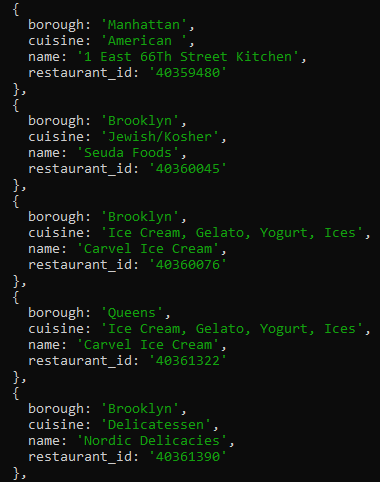
1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

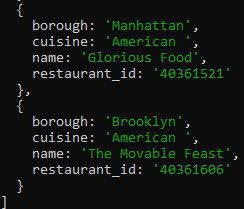




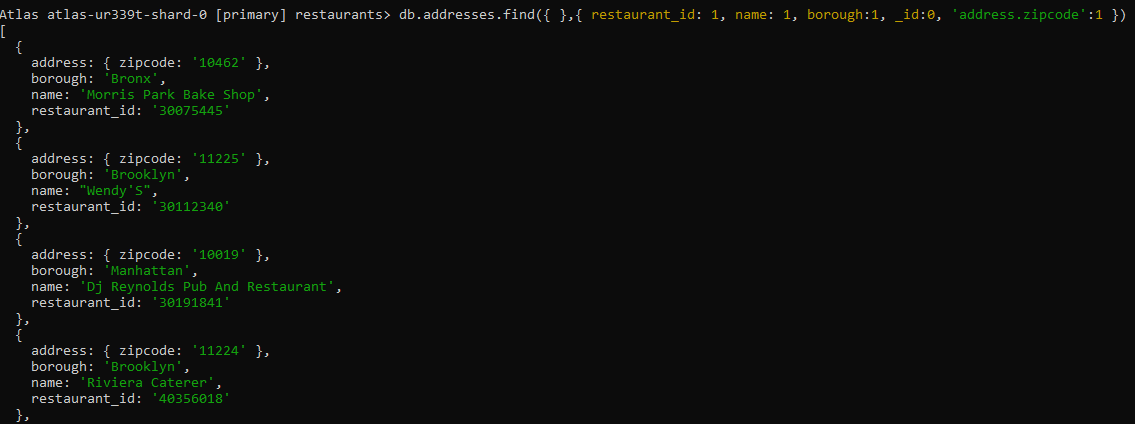


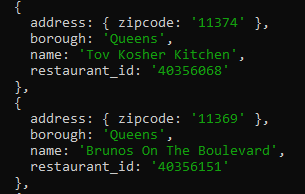


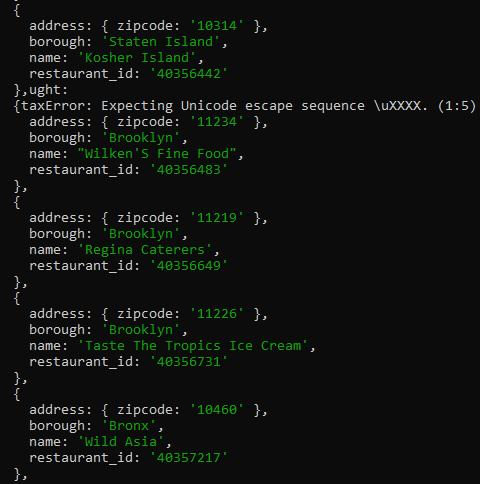


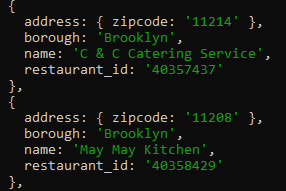


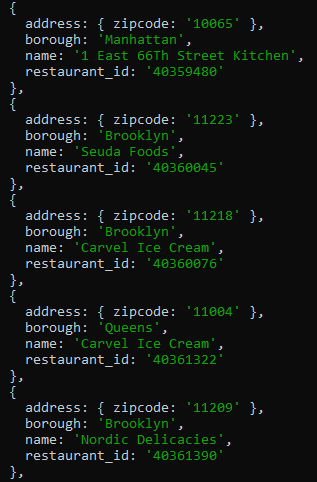
1. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

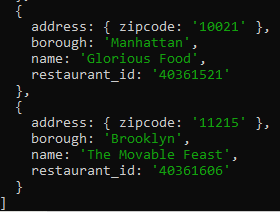












1. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

db.addresses.find({borough:’Bronx’}).limit(5)

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

db.addresses.find({borough:'Bronx'})

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

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

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

db.addresses.find({grades : { $elemMatch:{"score":{$gt : 90}}}})

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

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

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

db.addresses.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.addresses.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.addresses.find({

"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 grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

db.addresses.find( {"cuisine" : {$ne : "American "},

"grades.grade" :"A",

"borough": {$ne : "Brooklyn"}

} ).sort({"cuisine":-1})

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

db.addresses.find(

{name: ‘wil’},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1})

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

db.addresses.find(

{name: /ces$/},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1}

)

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

db.addresses.find(

{"name": /.\*Reg.\*/},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1}

)

1. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

db.addresses.find({ "borough": "Bronx" ,

$or : [{ "cuisine" : "American " },{ "cuisine" : "Chinese" }]

})

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.

db.addresses.find(

{"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1}

)

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.

db.addresses.find(

{"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1}

)

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

db.addresses.find(

{"grades.score" : { $not: {$gt : 10}}},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1})

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

db.addresses.find(

{$or: [{name: /^Wil/},

{"$and": [{"cuisine" : {$ne :"American "}},{"cuisine" : {$ne :"Chinese"}} ]} ]},

{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1})

1. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates.

db.addresses.find(

{

"grades.date": ISODate("2014-08-11T00:00:00Z"),

"grades.grade":"A" ,

"grades.score" : 11},

{"restaurant\_id" : 1,"name":1,"grades":1})

1. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

db.addresses.find(

{ "grades.1.date": ISODate("2014-08-11T00:00:00Z"),

"grades.1.grade":"A" ,

"grades.1.score" : 9},{"restaurant\_id" : 1,"name":1,"grades":1})

1. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52.

db.addresses.find(

{

"address.coord.1": {$gt : 42, $lte : 52}},

{"restaurant\_id" : 1,"name":1,"address":1,"coord":1})

1. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

db.addresses.find().sort({"name":1})

1. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

db.addresses.find().sort({"name": -1})

1. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

db.addresses.find().sort({"cuisine":1,"borough" : -1,})

1. Write a MongoDB query to know whether all the addresses contains the street or not.

db.addresses.find(

{"address.street" :

{ $exists : true }})

1. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

db.addresses.find(

{"address.coord" :

{$type : 1}})

1. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

db.addresses.find(

{"grades.score" :{$mod : [7,0]}},

{"restaurant\_id" : 1,"name":1,"grades":1})

1. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

db.addresses.find(

{ name : { $regex : "mon.\*", $options: "i" }},

{ "name":1,"borough":1,"address.coord":1,"cuisine" :1})

1. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

db.addresses.find(

{ name : { $regex : /^Mad/i, }} ,

{ "name":1, "borough":1,"address.coord":1,"cuisine" :1})