-:MONGODB:-

Assignment-1

ASHRAF NOMANI

Use mongo_practice

dB.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'

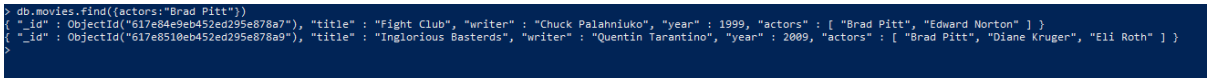
}])

Query/Find Documents

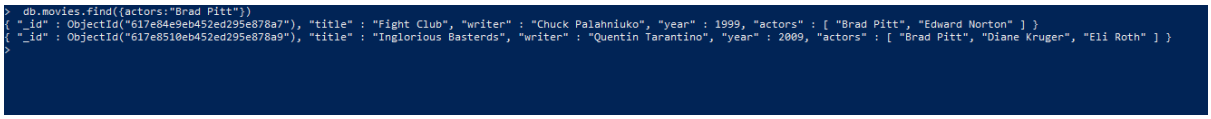
1. db.movies.find()

2. db.movies.find({ writer: 'Quentin Taranito'})

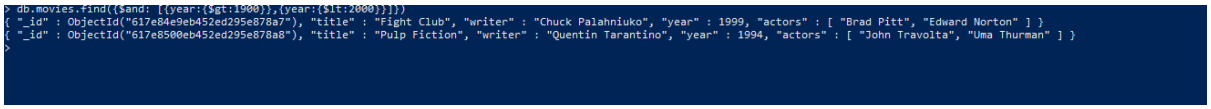
3. db.movies.find({ actors: 'Brad Pitt'})



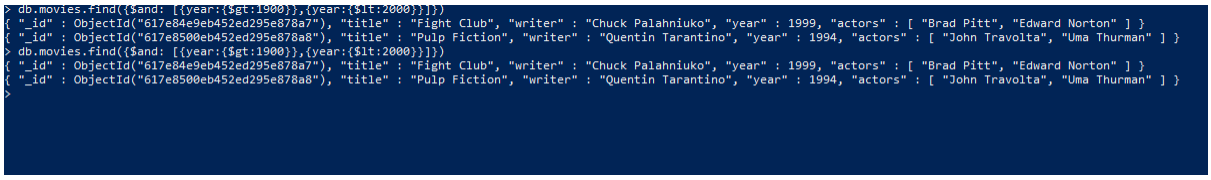
4. db.movies.find({ franchise: “The Hobbit”})



5. db.movies.find({year:{$lt: 1999, $gt:1990}})

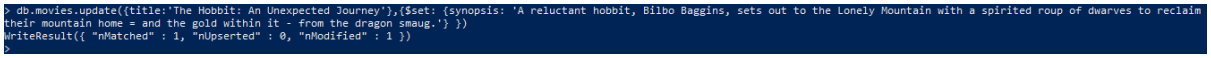


6. db.movies.find({year:{$lt: 2010,$gt: 2000}})

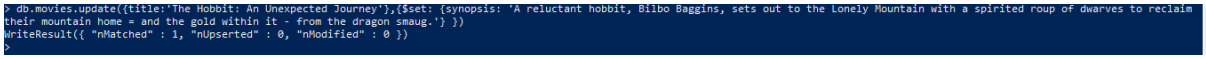


Update Documents

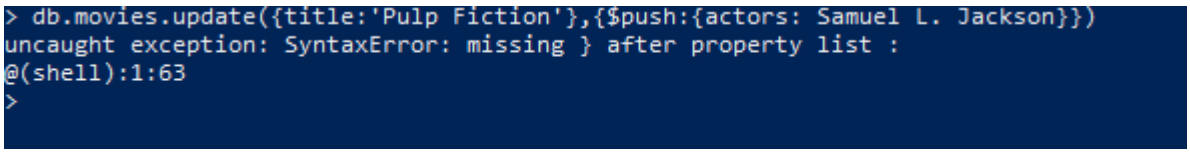
1. db.movies.update({title:'The Hobbit: An Unexpected Journey'},{$set: {synopsis: 'A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug.'} })



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.'} })



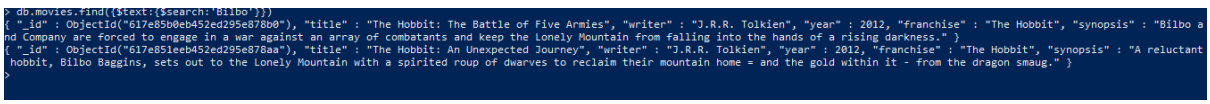
3. db.movies.update({title:'Pulp Fiction'},{$push:{actors: 'Samuel L. Jackson'}})



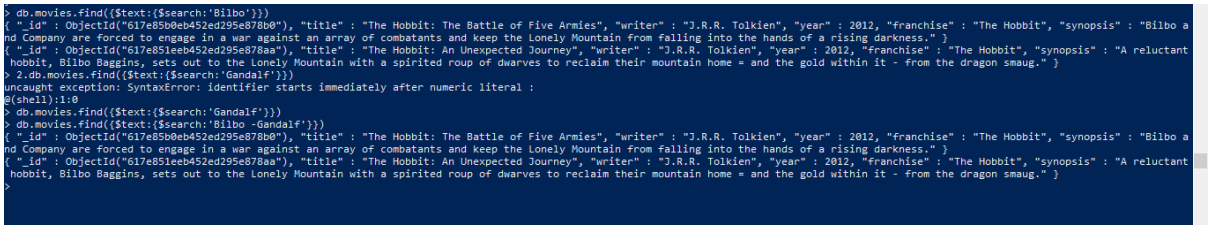
Text Search

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

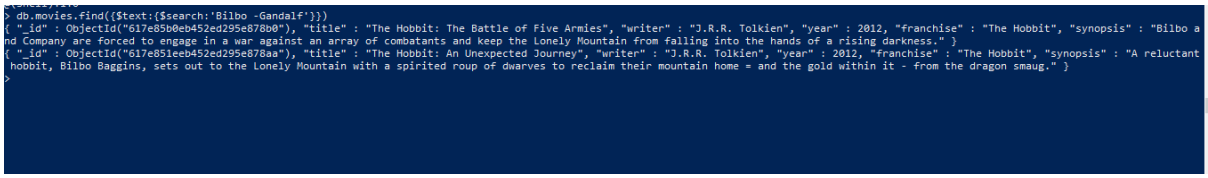
1. db.movies.find({$text:{$search:'Bilbo'}})



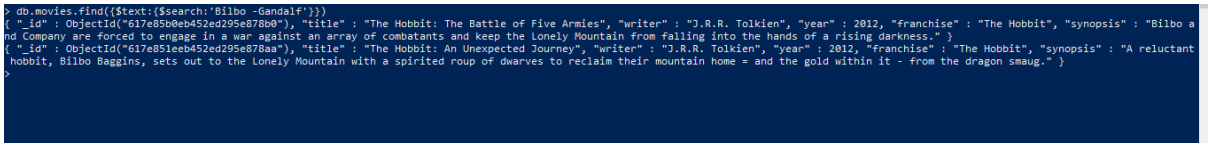
2. db.movies.find({$text:{$search:'Gandalf'}})



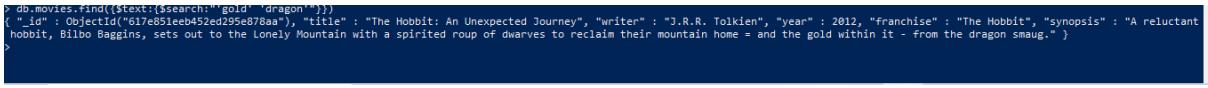
3. db.movies.find({$text:{$search:'Bilbo -Gandalf'}})



4. db.movies.find({$text:{$search:'dwarves hobbit'}})

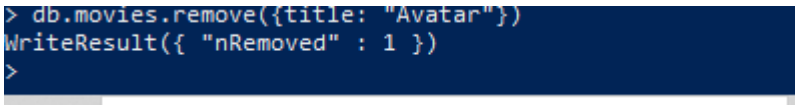


5. db.movies.find({$text:{$search:"'gold' 'dragon'"}})

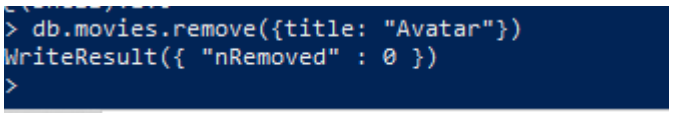


Delete Documents

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.createCollection('users')
{
        "ok" : 0,
        "errmsg" : "Collection already exists. NS: mongo_practice.users",
        "code" : 48,
        "codeName" : "NamespaceExists"
}
>
```

db.users.insertMany([

{ username: ‘GoodGuyGreg’,

‘first_name’:"Good Guy",

‘last_name’:"Greg",

},

{ ‘username’: ‘ScumbagSteve’,

‘full_name’:

{‘first_name’:"Scumbag",

‘last_name’:"Steve",

} }])

```
> db.users.insertMany([
... {  username: ‘GoodGuyGreg’,
... ‘first_name’:"Good Guy",
... ‘last_name’:"Greg",
... },
... {  ‘username’: ‘ScumbagSteve’,
... ‘full_name’:
... {‘first_name’:"Scumbag",
... ‘last_name’:"Steve",
... } } ])
uncaught exception: SyntaxError: illegal character :
@(shell):2:13
> 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"},
...
... {  username: "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"}
...   ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("617e9102eb452ed295e878b2"),
                ObjectId("617e9102eb452ed295e878b3"),
                ObjectId("617e9102eb452ed295e878b4"),
                ObjectId("617e9102eb452ed295e878b5"),
                ObjectId("617e9102eb452ed295e878b6"),
                ObjectId("617e9102eb452ed295e878b7")
        ]
}
>
```

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',
}])
```

```
> db.createCollection(‘posts’)
uncaught exception: SyntaxError: illegal character :
@(shell):1:20
> 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',
... }])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("617e914beb452ed295e878b8"),
                ObjectId("617e914beb452ed295e878b9"),
                ObjectId("617e914beb452ed295e878ba"),
                ObjectId("617e914beb452ed295e878bb"),
                ObjectId("617e914beb452ed295e878bc"),
                ObjectId("617e914beb452ed295e878bd")
        ]
}
>
```

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'],

}])

```
> db.createCollection('comments')
uncaught exception: SyntaxError: illegal character :
@(shell):1:20
> 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'],
... }])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("617e9179eb452ed295e878be"),
                ObjectId("617e9179eb452ed295e878bf"),
                ObjectId("617e9179eb452ed295e878c0"),
                ObjectId("617e9179eb452ed295e878c1"),
                ObjectId("617e9179eb452ed295e878c2")
        ]
}
>
```

Querying related collections

1. db.users.find()

2. db.posts.find()

```
> db.posts.find()
{ "_id" : ObjectId("617e9102eb452ed295e878b2"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }
{ "_id" : ObjectId("617e9102eb452ed295e878b3"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("617e9102eb452ed295e878b4"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("617e9102eb452ed295e878b5"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("617e9102eb452ed295e878b6"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("617e9102eb452ed295e878b7"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
{ "_id" : ObjectId("617e914beb452ed295e878b8"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }
{ "_id" : ObjectId("617e914beb452ed295e878b9"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("617e914beb452ed295e878ba"), "ussername" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }
{ "_id" : ObjectId("617e914beb452ed295e878bb"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("617e914beb452ed295e878bc"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("617e914beb452ed295e878bd"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
>
```

3. db.posts.find({userusername:'GoodGuyGreg'})
4. db.posts.find({userusername:'ScumbagSteve'})
5. db.comments.find()

```
> db.comments.find()
{ "_id" : ObjectId("617e9179eb452ed295e878be"), "username" : "GoodGuyGreg", " comment" : "Hope you got a good deal!", "post" : [ "617aa9cc6b910c064c32c339" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878bf"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : [ "617aa9cc6b910c064c32c33a" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c0"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement", "post" : [ "617aa9cc6b910c064c32c33b" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c1"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : [ "617aa9cc6b910c064c32c336" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : [ "617aa9cc6b910c064c32c338" ] }
>
```

6. db.comments.find({uusername:'GoodGuyGreg'})
7. db.comments.find({username:'ScumbagSteve'})

```
> db.comments.find({username:'ScumbagSteve'})
{ "_id" : ObjectId("617e9179eb452ed295e878c1"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : [ "617aa9cc6b910c064c32c336" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : [ "617aa9cc6b910c064c32c338" ] }
>
```

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

```
> db.posts.find({title:'Reports a bug in your code'})
{ "_id" : ObjectId("617e9102eb452ed295e878b4"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("617e914beb452ed295e878ba"), "ussername" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }
>
```

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({city:'ATLANTA',state:'GA'})

```
> show dbs
admin             0.000GB
config            0.000GB
local             0.000GB
mongoDB_practice  0.000GB
mongo_practice    0.002GB
population        0.002GB
studentmng        0.000GB
> use population
switched to db population
> db.zipcodes.find({city:'ATLANTA',state:'GA'})
{ "_id" : "30303", "city" : "ATLANTA", "loc" : [ -84.388846, 33.752504 ], "pop" : 1845, "state" : "GA" }
{ "_id" : "30305", "city" : "ATLANTA", "loc" : [ -84.385145, 33.831963 ], "pop" : 19122, "state" : "GA" }
{ "_id" : "30306", "city" : "ATLANTA", "loc" : [ -84.351418, 33.786027 ], "pop" : 20081, "state" : "GA" }
{ "_id" : "30307", "city" : "ATLANTA", "loc" : [ -84.335957, 33.769138 ], "pop" : 16330, "state" : "GA" }
{ "_id" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "state" : "GA" }
{ "_id" : "30309", "city" : "ATLANTA", "loc" : [ -84.388338, 33.798407 ], "pop" : 14766, "state" : "GA" }
{ "_id" : "30310", "city" : "ATLANTA", "loc" : [ -84.423173, 33.727849 ], "pop" : 34017, "state" : "GA" }
{ "_id" : "30311", "city" : "ATLANTA", "loc" : [ -84.470219, 33.722957 ], "pop" : 34880, "state" : "GA" }
{ "_id" : "30312", "city" : "ATLANTA", "loc" : [ -84.378125, 33.746749 ], "pop" : 17683, "state" : "GA" }
{ "_id" : "30313", "city" : "ATLANTA", "loc" : [ -84.39352, 33.76825 ], "pop" : 8038, "state" : "GA" }
{ "_id" : "30314", "city" : "ATLANTA", "loc" : [ -84.425546, 33.756103 ], "pop" : 26649, "state" : "GA" }
{ "_id" : "30315", "city" : "ATLANTA", "loc" : [ -84.380771, 33.705062 ], "pop" : 41061, "state" : "GA" }
{ "_id" : "30316", "city" : "ATLANTA", "loc" : [ -84.333913, 33.721686 ], "pop" : 34668, "state" : "GA" }
{ "_id" : "30317", "city" : "ATLANTA", "loc" : [ -84.31685, 33.749788 ], "pop" : 16395, "state" : "GA" }
{ "_id" : "30318", "city" : "ATLANTA", "loc" : [ -84.445432, 33.786454 ], "pop" : 53894, "state" : "GA" }
{ "_id" : "30319", "city" : "ATLANTA", "loc" : [ -84.335091, 33.868728 ], "pop" : 32138, "state" : "GA" }
{ "_id" : "30324", "city" : "ATLANTA", "loc" : [ -84.354867, 33.820609 ], "pop" : 15044, "state" : "GA" }
{ "_id" : "30326", "city" : "ATLANTA", "loc" : [ -84.358232, 33.848168 ], "pop" : 125, "state" : "GA" }
{ "_id" : "30327", "city" : "ATLANTA", "loc" : [ -84.419966, 33.862723 ], "pop" : 18467, "state" : "GA" }
{ "_id" : "30329", "city" : "ATLANTA", "loc" : [ -84.321402, 33.823555 ], "pop" : 17013, "state" : "GA" }
Type "it" for more
>
```

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

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

```
> db.zipcodes.aggregate([{$match:{city:'ATLANTA',state:'GA'}}])
{ "_id" : "30303", "city" : "ATLANTA", "loc" : [ -84.388846, 33.752504 ], "pop" : 1845, "state" : "GA" }
{ "_id" : "30305", "city" : "ATLANTA", "loc" : [ -84.385145, 33.831963 ], "pop" : 19122, "state" : "GA" }
{ "_id" : "30306", "city" : "ATLANTA", "loc" : [ -84.351418, 33.786027 ], "pop" : 20081, "state" : "GA" }
{ "_id" : "30307", "city" : "ATLANTA", "loc" : [ -84.335957, 33.769138 ], "pop" : 16330, "state" : "GA" }
{ "_id" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "state" : "GA" }
{ "_id" : "30309", "city" : "ATLANTA", "loc" : [ -84.388338, 33.798407 ], "pop" : 14766, "state" : "GA" }
{ "_id" : "30310", "city" : "ATLANTA", "loc" : [ -84.423173, 33.727849 ], "pop" : 34017, "state" : "GA" }
{ "_id" : "30311", "city" : "ATLANTA", "loc" : [ -84.470219, 33.722957 ], "pop" : 34880, "state" : "GA" }
{ "_id" : "30312", "city" : "ATLANTA", "loc" : [ -84.378125, 33.746749 ], "pop" : 17683, "state" : "GA" }
{ "_id" : "30313", "city" : "ATLANTA", "loc" : [ -84.39352, 33.76825 ], "pop" : 8038, "state" : "GA" }
{ "_id" : "30314", "city" : "ATLANTA", "loc" : [ -84.425546, 33.756103 ], "pop" : 26649, "state" : "GA" }
{ "_id" : "30315", "city" : "ATLANTA", "loc" : [ -84.380771, 33.705062 ], "pop" : 41061, "state" : "GA" }
{ "_id" : "30316", "city" : "ATLANTA", "loc" : [ -84.333913, 33.721686 ], "pop" : 34668, "state" : "GA" }
{ "_id" : "30317", "city" : "ATLANTA", "loc" : [ -84.31685, 33.749788 ], "pop" : 16395, "state" : "GA" }
{ "_id" : "30318", "city" : "ATLANTA", "loc" : [ -84.445432, 33.786454 ], "pop" : 53894, "state" : "GA" }
{ "_id" : "30319", "city" : "ATLANTA", "loc" : [ -84.335091, 33.868728 ], "pop" : 32138, "state" : "GA" }
{ "_id" : "30324", "city" : "ATLANTA", "loc" : [ -84.354867, 33.820609 ], "pop" : 15044, "state" : "GA" }
{ "_id" : "30326", "city" : "ATLANTA", "loc" : [ -84.358232, 33.848168 ], "pop" : 125, "state" : "GA" }
{ "_id" : "30327", "city" : "ATLANTA", "loc" : [ -84.419966, 33.862723 ], "pop" : 18467, "state" : "GA" }
{ "_id" : "30329", "city" : "ATLANTA", "loc" : [ -84.321402, 33.823555 ], "pop" : 17013, "state" : "GA" }
Type "it" for more
```

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

db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{$group:{_id:'$_id'}},{$count:'uni_zipcode'}])

```
> db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{$group:{_id:'$_id'}},{$count:'uni_zipcode'}])
{ "uni_zipcode" : 41 }
>
```

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

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

```
> db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{$group:{_id:'city',totalpop:{$sum:'$pop'}}}])
{ "_id" : "city", "totalpop" : 630046 }
>
```

Populations By State

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

db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}}])

```
> db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{$group:{_id:'city',totalpop:{$sum:'$pop'}}}])
{ "_id" : "city", "totalpop" : 630046 }
> db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}}])
{ "_id" : "NC", "totalpop" : 6628637 }
{ "_id" : "OH", "totalpop" : 10846517 }
{ "_id" : "MN", "totalpop" : 4372982 }
{ "_id" : "ME", "totalpop" : 1226648 }
{ "_id" : "IN", "totalpop" : 5544136 }
{ "_id" : "NY", "totalpop" : 17990402 }
{ "_id" : "CT", "totalpop" : 3287116 }
{ "_id" : "ND", "totalpop" : 638272 }
{ "_id" : "MT", "totalpop" : 798948 }
{ "_id" : "MD", "totalpop" : 4781379 }
{ "_id" : "AL", "totalpop" : 4040587 }
{ "_id" : "NJ", "totalpop" : 7730188 }
{ "_id" : "MO", "totalpop" : 5110648 }
{ "_id" : "KS", "totalpop" : 2475285 }
{ "_id" : "GA", "totalpop" : 6478216 }
{ "_id" : "NE", "totalpop" : 1578139 }
{ "_id" : "LA", "totalpop" : 4217595 }
{ "_id" : "AR", "totalpop" : 2350725 }
{ "_id" : "CO", "totalpop" : 3293755 }
{ "_id" : "WY", "totalpop" : 453528 }
Type "it" for more
>
```

2. sort the results by population, highest first

db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}}])

```
> db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}}])
{ "_id" : "CA", "totalpop" : 29754890 }
{ "_id" : "NY", "totalpop" : 17990402 }
{ "_id" : "TX", "totalpop" : 16984601 }
{ "_id" : "FL", "totalpop" : 12686644 }
{ "_id" : "PA", "totalpop" : 11881643 }
{ "_id" : "IL", "totalpop" : 11427576 }
{ "_id" : "OH", "totalpop" : 10846517 }
{ "_id" : "MI", "totalpop" : 9295297 }
{ "_id" : "NJ", "totalpop" : 7730188 }
{ "_id" : "NC", "totalpop" : 6628637 }
{ "_id" : "GA", "totalpop" : 6478216 }
{ "_id" : "VA", "totalpop" : 6181479 }
{ "_id" : "MA", "totalpop" : 6016425 }
{ "_id" : "IN", "totalpop" : 5544136 }
{ "_id" : "MO", "totalpop" : 5110648 }
{ "_id" : "WI", "totalpop" : 4891769 }
{ "_id" : "TN", "totalpop" : 4876457 }
{ "_id" : "WA", "totalpop" : 4866692 }
{ "_id" : "MD", "totalpop" : 4781379 }
{ "_id" : "MN", "totalpop" : 4372982 }
Type "it" for more
```

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

db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}},{$limit:3}])

```
> db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}},{$limit:3}])
{ "_id" : "CA", "totalpop" : 29754890 }
{ "_id" : "NY", "totalpop" : 17990402 }
{ "_id" : "TX", "totalpop" : 16984601 }
>
```

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: '$state' } ,totalpop:{$sum:'$pop'}}}])

```
> db.zipcodes.aggregate([{$group:{_id: { city: '$city', state: '$state' } ,totalpop:{$sum:'$pop
{ "_id" : { "city" : "DUNCANNON", "state" : "PA" }, "totalpop" : 10021 }
{ "_id" : { "city" : "MILLS", "state" : "PA" }, "totalpop" : 653 }
{ "_id" : { "city" : "OTTO", "state" : "NC" }, "totalpop" : 2297 }
{ "_id" : { "city" : "BOLIVAR", "state" : "TN" }, "totalpop" : 10327 }
{ "_id" : { "city" : "SONDHEIMER", "state" : "LA" }, "totalpop" : 887 }
{ "_id" : { "city" : "DOGWOOD", "state" : "TX" }, "totalpop" : 5942 }
{ "_id" : { "city" : "MC LOUD", "state" : "OK" }, "totalpop" : 3334 }
{ "_id" : { "city" : "KONAWA", "state" : "OK" }, "totalpop" : 2800 }
{ "_id" : { "city" : "WINDYVILLE", "state" : "MO" }, "totalpop" : 769 }
{ "_id" : { "city" : "HURT", "state" : "VA" }, "totalpop" : 5283 }
{ "_id" : { "city" : "LORAINE", "state" : "ND" }, "totalpop" : 1455 }
{ "_id" : { "city" : "NEWPORT CENTER", "state" : "VT" }, "totalpop" : 1367 }
{ "_id" : { "city" : "RANGER", "state" : "WV" }, "totalpop" : 5703 }
{ "_id" : { "city" : "GATES", "state" : "TN" }, "totalpop" : 2508 }
{ "_id" : { "city" : "MALVERN", "state" : "AL" }, "totalpop" : 1686 }
{ "_id" : { "city" : "ROCKY GAP", "state" : "VA" }, "totalpop" : 510 }
{ "_id" : { "city" : "OBERON", "state" : "ND" }, "totalpop" : 556 }
{ "_id" : { "city" : "LODGE GRASS", "state" : "MT" }, "totalpop" : 2938 }
{ "_id" : { "city" : "MILLSAP", "state" : "TX" }, "totalpop" : 9110 }
{ "_id" : { "city" : "VERNAL", "state" : "UT" }, "totalpop" : 17641 }
Type "it" for more
```

2. sort the results by population, highest first

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

```
> db.zipcodes.aggregate([{$group:{_id: { city: '$city', state: '$state' } ,totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}
}])
{ "_id" : { "city" : "CHICAGO", "state" : "IL" }, "totalpop" : 2452177 }
{ "_id" : { "city" : "BROOKLYN", "state" : "NY" }, "totalpop" : 2300504 }
{ "_id" : { "city" : "LOS ANGELES", "state" : "CA" }, "totalpop" : 2102295 }
{ "_id" : { "city" : "HOUSTON", "state" : "TX" }, "totalpop" : 2095918 }
{ "_id" : { "city" : "PHILADELPHIA", "state" : "PA" }, "totalpop" : 1610956 }
{ "_id" : { "city" : "NEW YORK", "state" : "NY" }, "totalpop" : 1476790 }
{ "_id" : { "city" : "BRONX", "state" : "NY" }, "totalpop" : 1209548 }
{ "_id" : { "city" : "SAN DIEGO", "state" : "CA" }, "totalpop" : 1049298 }
{ "_id" : { "city" : "DETROIT", "state" : "MI" }, "totalpop" : 963243 }
{ "_id" : { "city" : "DALLAS", "state" : "TX" }, "totalpop" : 940191 }
{ "_id" : { "city" : "PHOENIX", "state" : "AZ" }, "totalpop" : 890853 }
{ "_id" : { "city" : "MIAMI", "state" : "FL" }, "totalpop" : 825232 }
{ "_id" : { "city" : "SAN JOSE", "state" : "CA" }, "totalpop" : 816653 }
{ "_id" : { "city" : "SAN ANTONIO", "state" : "TX" }, "totalpop" : 811792 }
{ "_id" : { "city" : "BALTIMORE", "state" : "MD" }, "totalpop" : 733081 }
{ "_id" : { "city" : "SAN FRANCISCO", "state" : "CA" }, "totalpop" : 723993 }
{ "_id" : { "city" : "MEMPHIS", "state" : "TN" }, "totalpop" : 632837 }
{ "_id" : { "city" : "SACRAMENTO", "state" : "CA" }, "totalpop" : 628279 }
{ "_id" : { "city" : "JACKSONVILLE", "state" : "FL" }, "totalpop" : 610160 }
{ "_id" : { "city" : "ATLANTA", "state" : "GA" }, "totalpop" : 609591 }
Type "it" for more
>
```

3. 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: '$state' }
,totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}},{$limit:3}])

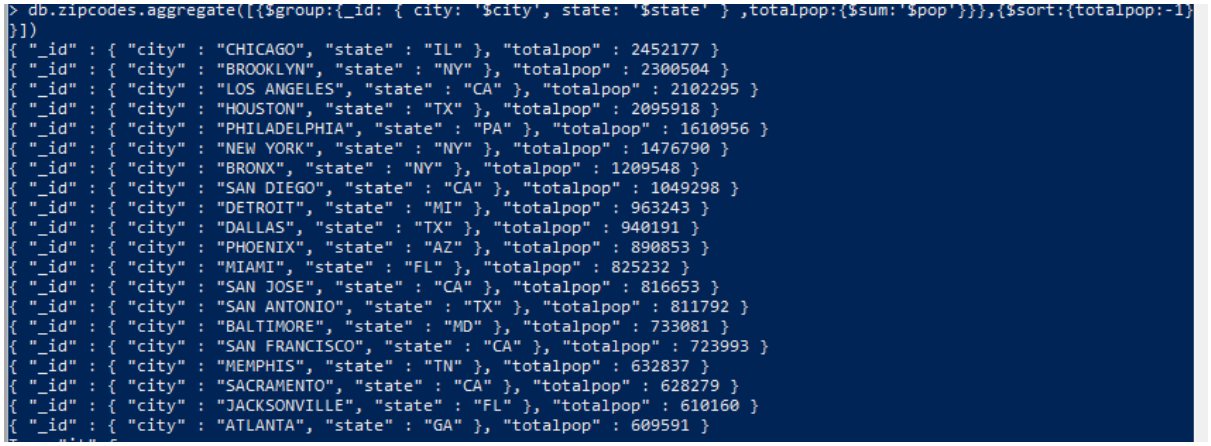
4. What are the top 3 cities in population in Texas?

db.zipcodes.aggregate([{$match:{state:'TX'}},{$group:{_id:{state:'$state',city:'$city'},totalpop:{$sum:'
$pop'}}},{$sort:{totalpop:-1}},{$limit:3}])

Bonus

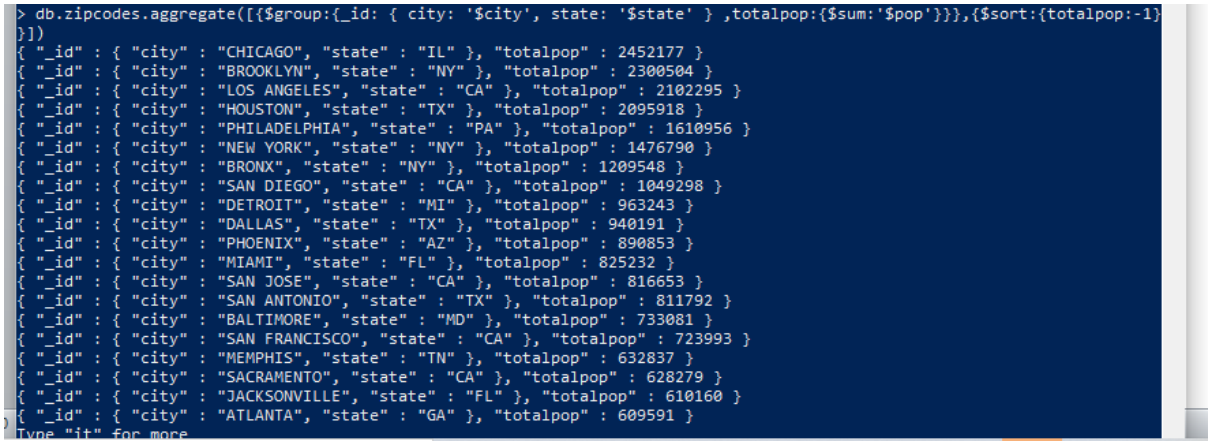
1. Write a query to get the average city population for each state.

db.zipcodes.aggregate([{$group:{_id:{state:'$state',city:'$city'},Avgpop:{$avg:'$pop'}}}])



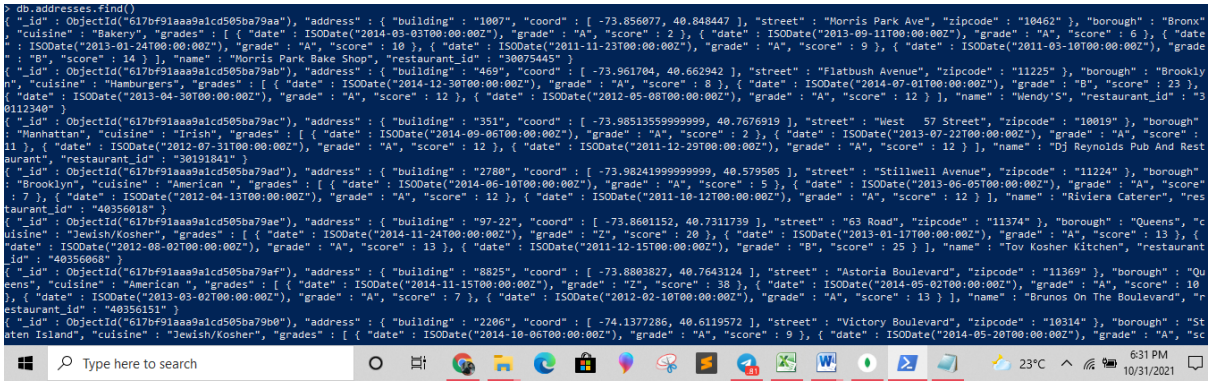
2. What are the top 3 states in terms of average city population?

db.zipcodes.aggregate([{$group:{_id:{state:'$state',city:'$city'},Avgpop:{$avg:'$pop'}}},{$sort:{Avgpop:-1}},{$limit:3}])



ASSIGNMENT 3

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



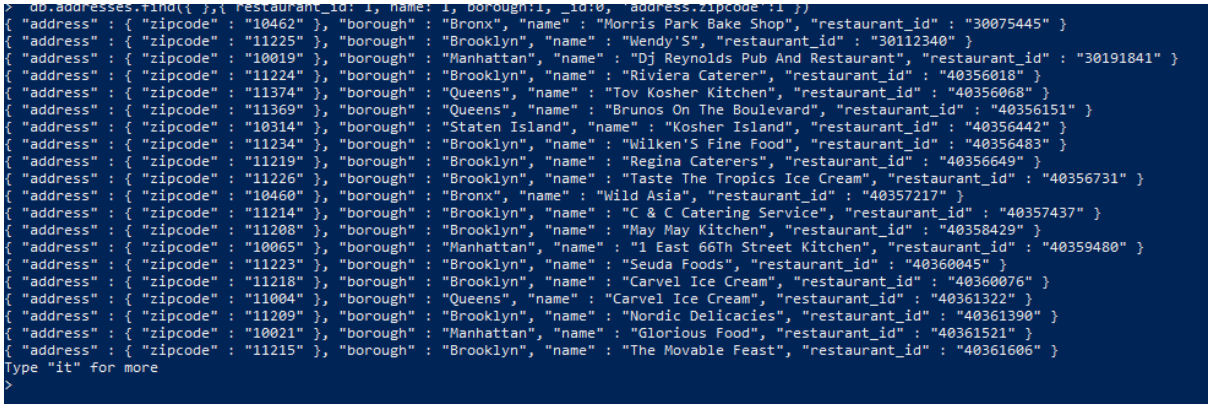
2. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

```
> db.addresses.find({ },{ restaurant_id: 1, name: 1, borough:1, cuisine:1})
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79aa"), "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ab"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ac"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ad"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ae"), "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79af"), "borough" : "Queens", "cuisine" : "American ", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b0"), "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b1"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b2"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b3"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b4"), "borough" : "Bronx", "cuisine" : "American ", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b6"), "borough" : "Brooklyn", "cuisine" : "Chinese", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b8"), "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b9"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ba"), "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bb"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bc"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bd"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
Type "it" for more
>
```

3. 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.

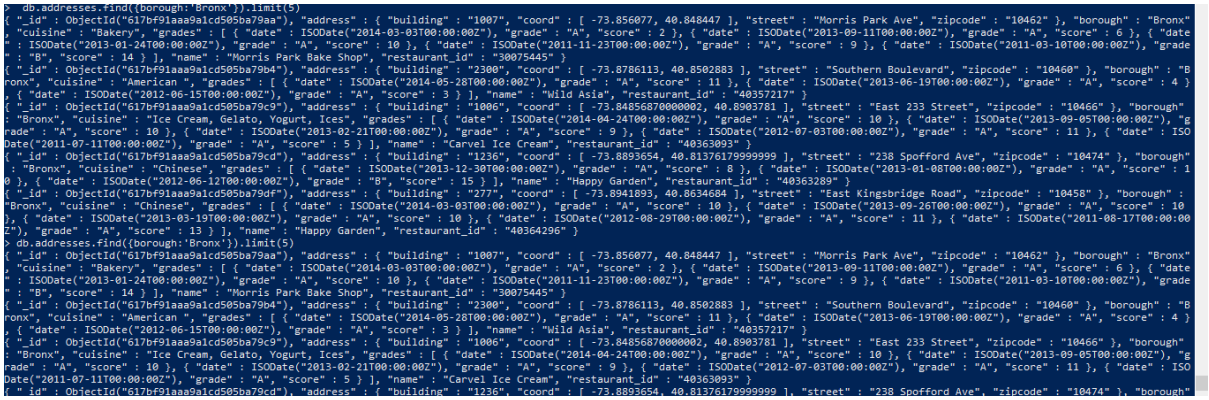
```
> db.addresses.find({ },{ restaurant_id: 1, name: 1, borough:1, cuisine:1, _id:0 })
{ "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "borough" : "Queens", "cuisine" : "American ", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "borough" : "Bronx", "cuisine" : "American ", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "borough" : "Brooklyn", "cuisine" : "Chinese", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "borough" : "Manhattan", "cuisine" : "American ", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
Type "it" for more
>
```

3. 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

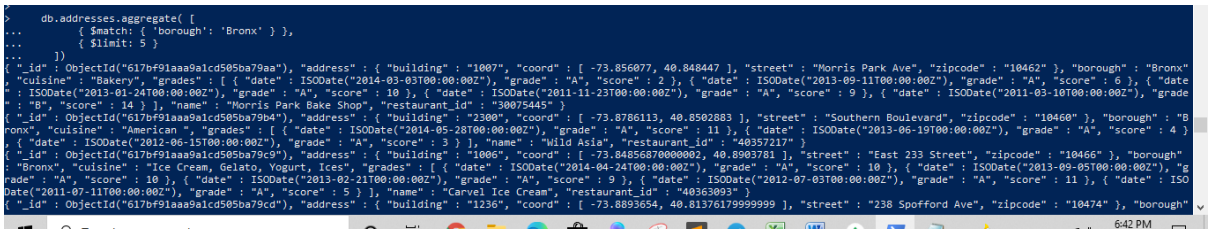


```
> db.addresses.find({ },{ restaurant_id: 1, name: 1, borough:1, _id:0, "address.zipcode":1 })
{ "address" : { "zipcode" : "10462" }, "borough" : "Bronx", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "address" : { "zipcode" : "11225" }, "borough" : "Brooklyn", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "address" : { "zipcode" : "10019" }, "borough" : "Manhattan", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "address" : { "zipcode" : "11224" }, "borough" : "Brooklyn", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "address" : { "zipcode" : "11374" }, "borough" : "Queens", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "address" : { "zipcode" : "11369" }, "borough" : "Queens", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "address" : { "zipcode" : "10314" }, "borough" : "Staten Island", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "address" : { "zipcode" : "11234" }, "borough" : "Brooklyn", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "address" : { "zipcode" : "11219" }, "borough" : "Brooklyn", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "address" : { "zipcode" : "11226" }, "borough" : "Brooklyn", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "address" : { "zipcode" : "10460" }, "borough" : "Bronx", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "address" : { "zipcode" : "11214" }, "borough" : "Brooklyn", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "address" : { "zipcode" : "11208" }, "borough" : "Brooklyn", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "address" : { "zipcode" : "10065" }, "borough" : "Manhattan", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "address" : { "zipcode" : "11223" }, "borough" : "Brooklyn", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "address" : { "zipcode" : "11218" }, "borough" : "Brooklyn", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "address" : { "zipcode" : "11004" }, "borough" : "Queens", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "address" : { "zipcode" : "11209" }, "borough" : "Brooklyn", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "address" : { "zipcode" : "10021" }, "borough" : "Manhattan", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "address" : { "zipcode" : "11215" }, "borough" : "Brooklyn", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
Type "it" for more
>
```

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

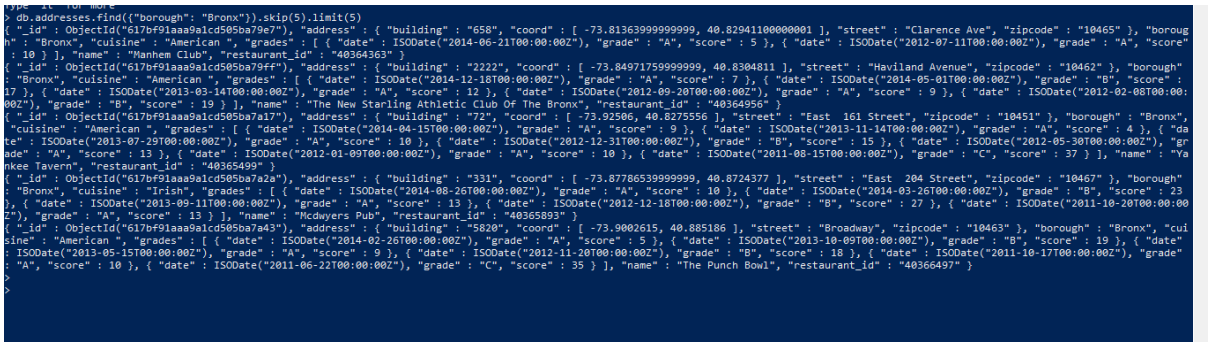


6. Write a MongoDB query to display all the restaurant which is in the borough Bronx.



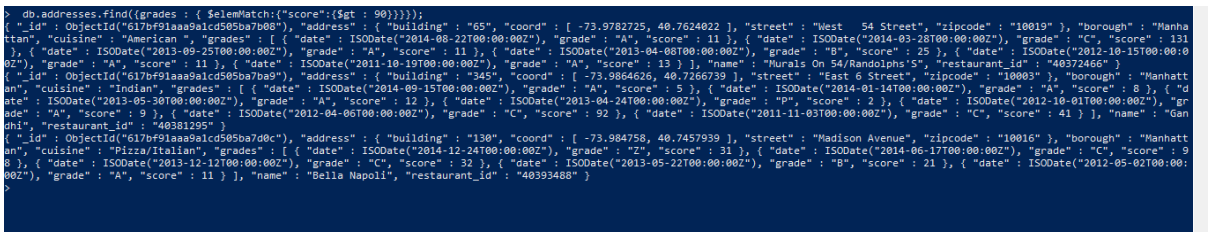
7. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.\

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

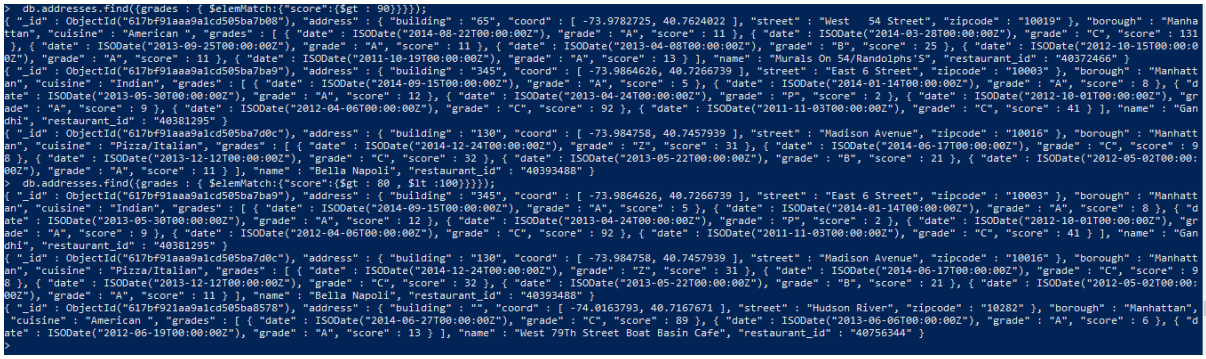


8. Write a MongoDB query to find the restaurants who achieved a score more than 90.

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

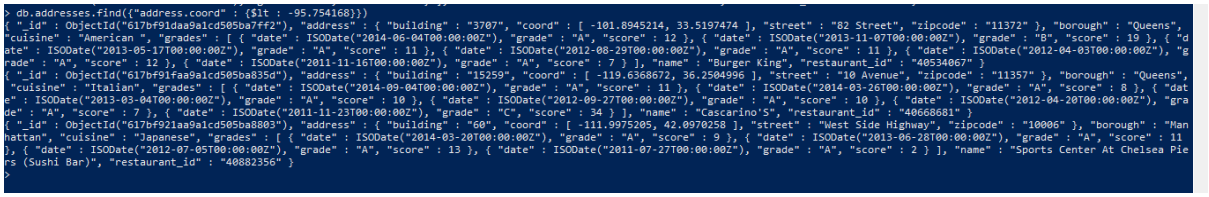


9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.



10. 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}})



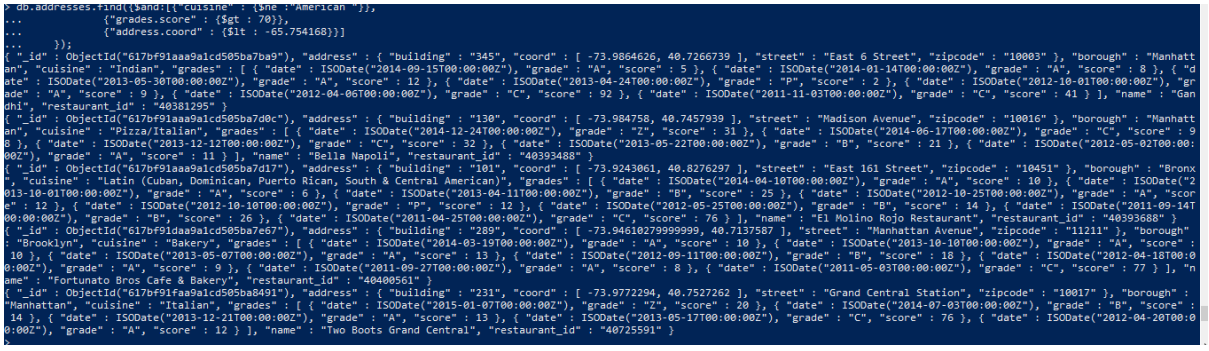
11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

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

{"grades.score" : {$gt : 70}},

{"address.coord" : {$lt : -65.754168}}]

});



12. 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 "},

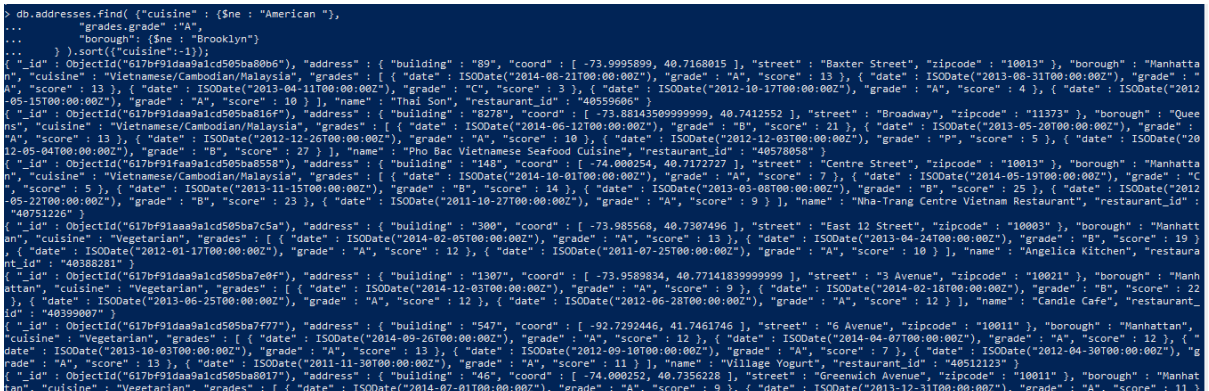
"grades.score" :{$gt: 70},

"address.coord" : {$lt : -65.754168}

});

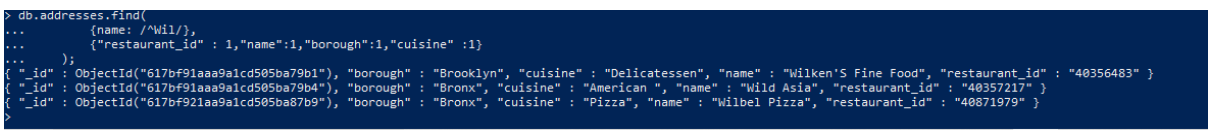
13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

```
db.addresses.find( {"cuisine" : {$ne : "American "},
    "grades.grade" :"A",
    "borough": {$ne : "Brooklyn"}
  } ).sort({"cuisine":-1});
```



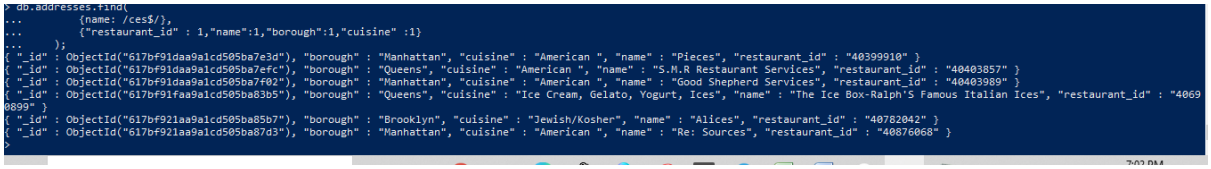
14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
db.addresses.find(
    {name: /^Wil/},
    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
  );
```



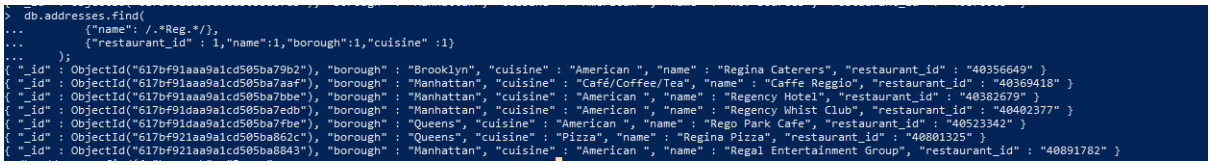
15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

```
db.addresses.find(
    {name: /ces$/},
    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
);
```



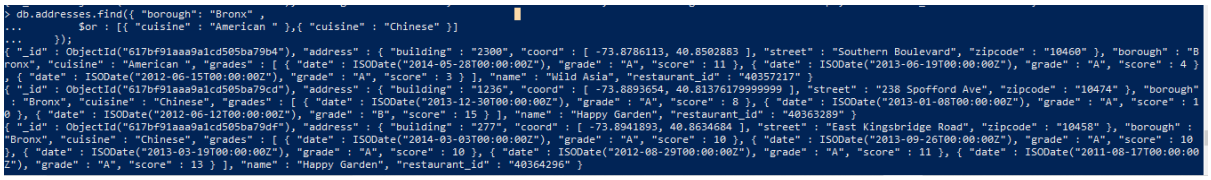
16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

```
db.addresses.find(
    {"name": /.*Reg.*/},
    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
);
```



17. 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" }]
});
```



18. 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}
);
```

```
> db.addresses.find(
...         {"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},
...         {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
...      );
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79aa"), "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ab"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ad"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ae"), "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79af"), "borough" : "Queens", "cuisine" : "American ", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b0"), "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b1"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b2"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b3"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b4"), "borough" : "Bronx", "cuisine" : "American ", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b6"), "borough" : "Brooklyn", "cuisine" : "Chinese", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b8"), "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b9"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ba"), "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bb"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bd"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79be"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Sal'S Deli", "restaurant_id" : "40361618" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c0"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Steve Chu'S Deli & Grocery", "restaurant_id" : "40361998" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c4"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "White Castle", "restaurant_id" : "40362344" }
Type "it" for more
```

19. 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}
  );
```

```
> db.addresses.find(
...         {"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},
...         {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
...      );
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ac"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bc"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bf"), "borough" : "Manhattan", "cuisine" : "Delicatessen", "name" : "Bully'S Deli", "restaurant_id" : "40361708" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c1"), "borough" : "Manhattan", "cuisine" : "Chicken", "name" : "Harriet'S Kitchen", "restaurant_id" : "40362098" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c2"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "P & S Deli Grocery", "restaurant_id" : "40362264" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c3"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Angelika Film Center", "restaurant_id" : "40362274" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c6"), "borough" : "Manhattan", "cuisine" : "Turkish", "name" : "The Country Cafe", "restaurant_id" : "40362715" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c8"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Downtown Deli", "restaurant_id" : "40363021" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79cc"), "borough" : "Manhattan", "cuisine" : "Bakery", "name" : "Olive'S", "restaurant_id" : "40363151" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ce"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Café Metro", "restaurant_id" : "40363298" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79d0"), "borough" : "Manhattan", "cuisine" : "Sandwiches/Salads/Mixed Buffet", "name" : "Lexler Deli", "restaurant_id" : "40363426" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79d4"), "borough" : "Manhattan", "cuisine" : "Continental", "name" : "Lorenzo & Maria'S", "restaurant_id" : "40363630" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79d5"), "borough" : "Manhattan", "cuisine" : "Pizza", "name" : "Domino'S Pizza", "restaurant_id" : "40363644" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79d6"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Berkely", "restaurant_id" : "40363685" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79da"), "borough" : "Manhattan", "cuisine" : "Pizza", "name" : "Domino'S Pizza", "restaurant_id" : "40363945" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79db"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Spoon Bread Catering", "restaurant_id" : "40364179" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79e1"), "borough" : "Manhattan", "cuisine" : "Chicken", "name" : "Texas Rotisserie", "restaurant_id" : "40364304" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79e4"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Metropolitan Club", "restaurant_id" : "40364347" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79e5"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Palm Restaurant", "restaurant_id" : "40364355" }
Type "it" for more
```

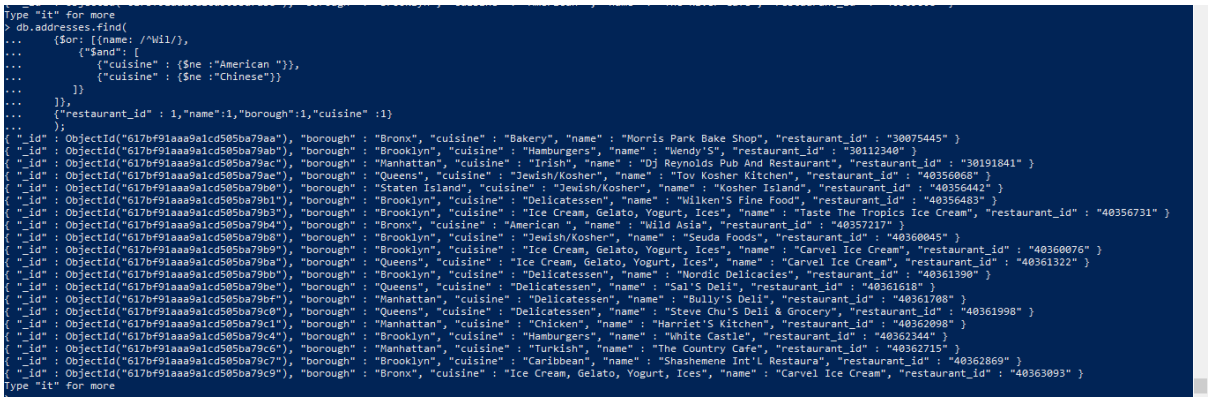
20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

```
db.addresses.find(
    {"grades.score" : { $not: {$gt : 10}}},
    {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
  );
```

```
> db.addresses.find(
...         {"grades.score" : { $not: {$gt : 10}}},
...         {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
...      );
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bb"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79c4"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "White Castle", "restaurant_id" : "40362344" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79d7"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Sonny'S Heros", "restaurant_id" : "40363744" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79e7"), "borough" : "Bronx", "cuisine" : "American ", "name" : "Manhem Club", "restaurant_id" : "40364363" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79f5"), "borough" : "Staten Island", "cuisine" : "American ", "name" : "Great Kills Yacht Club", "restaurant_id" : "40364610" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79fc"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Serendipity 3", "restaurant_id" : "40364863" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a00"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "White Horse Tavern", "restaurant_id" : "40364958" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a09"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dorrian'S Red Hand Restaurant", "restaurant_id" : "40365239" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a5a"), "borough" : "Manhattan", "cuisine" : "Mexican", "name" : "Mexico Lindo Restaurant", "restaurant_id" : "40367038" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a7a"), "borough" : "Brooklyn", "cuisine" : "Greek", "name" : "El Greco Diner", "restaurant_id" : "40367795" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a80"), "borough" : "Bronx", "cuisine" : "Not Listed/Not Applicable", "name" : "The Lark'S Nest", "restaurant_id" : "40367946" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a81"), "borough" : "Bronx", "cuisine" : "Café/Coffee/Tea", "name" : "Terrace Cafe", "restaurant_id" : "40368018" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a82"), "borough" : "Bronx", "cuisine" : "African", "name" : "African Terrace", "restaurant_id" : "40368021" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a85"), "borough" : "Bronx", "cuisine" : "American ", "name" : "African Market (Baboon Cafe)", "restaurant_id" : "40368026" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a86"), "borough" : "Staten Island", "cuisine" : "Italian", "name" : "Roadhouse Restaurant", "restaurant_id" : "40368034" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7aa8"), "borough" : "Manhattan", "cuisine" : "French", "name" : "Pergola Des Artistes", "restaurant_id" : "40369139" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7ab4"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Mcdonald'S", "restaurant_id" : "40369535" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7ab8"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The River Cafe", "restaurant_id" : "40369608" }
Type "it" for more
>
```

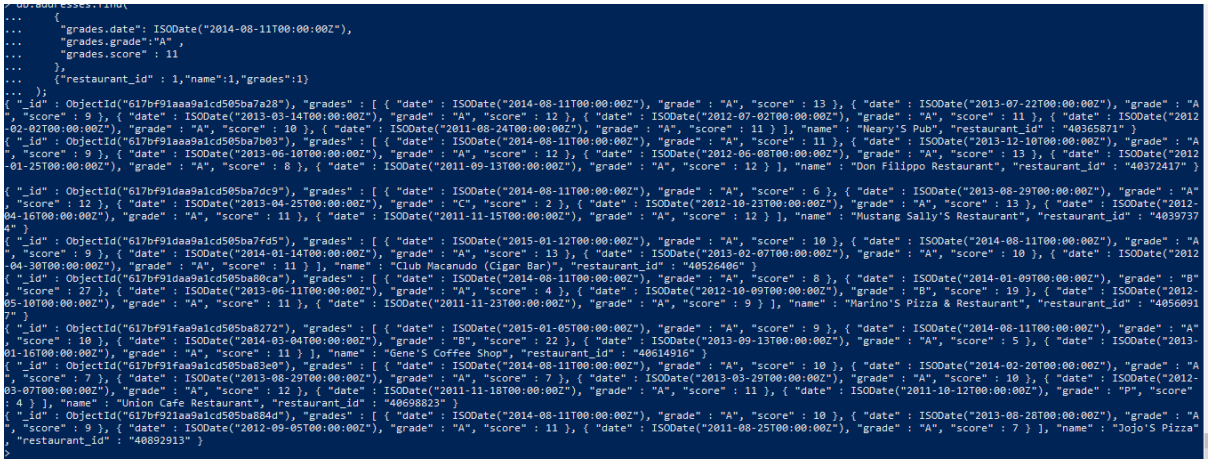
21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil

```
db.addresses.find(
   {$or: [{name: /^Wil/},
      {"$and": [
         {"cuisine" : {$ne :"American "}},
         {"cuisine" : {$ne :"Chinese"}}
      ]}
   ]},
   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
   );
```



22. 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}
);
```

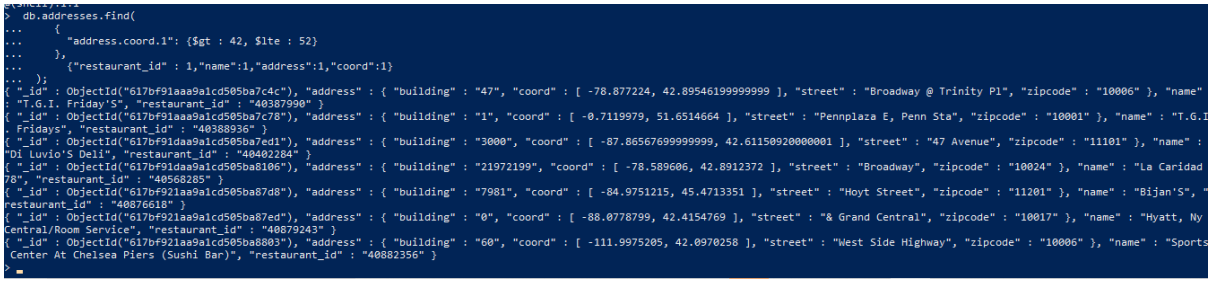


23. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z"

```
b.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}
);
```

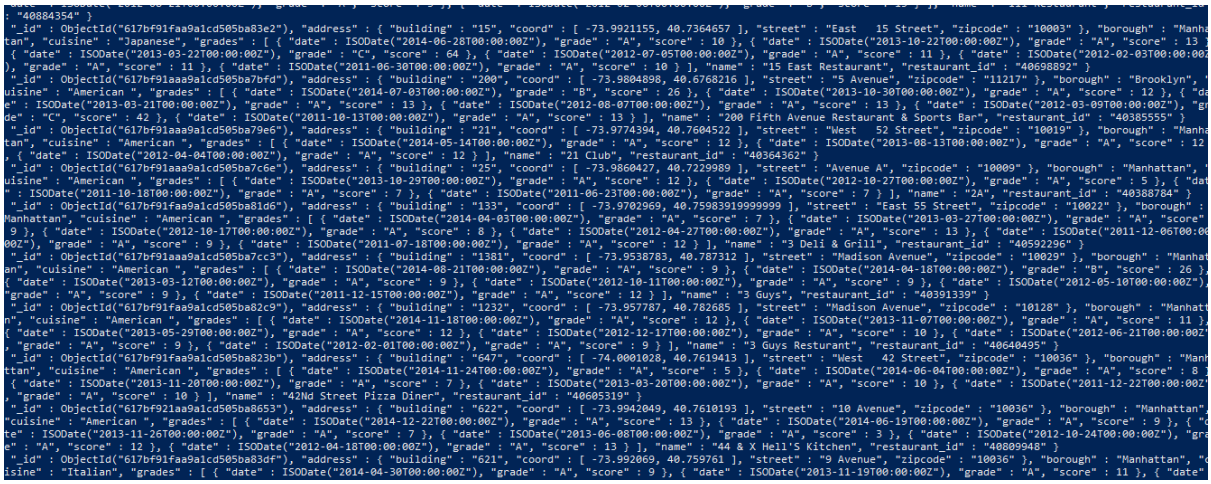
24. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

```
db.addresses.find(
   {
    "address.coord.1": {$gt : 42, $lte : 52}
   },
    {"restaurant_id" : 1,"name":1,"address":1,"coord":1}
);
```



25. 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});



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

db.addresses.find().sort(

{"name":-1}

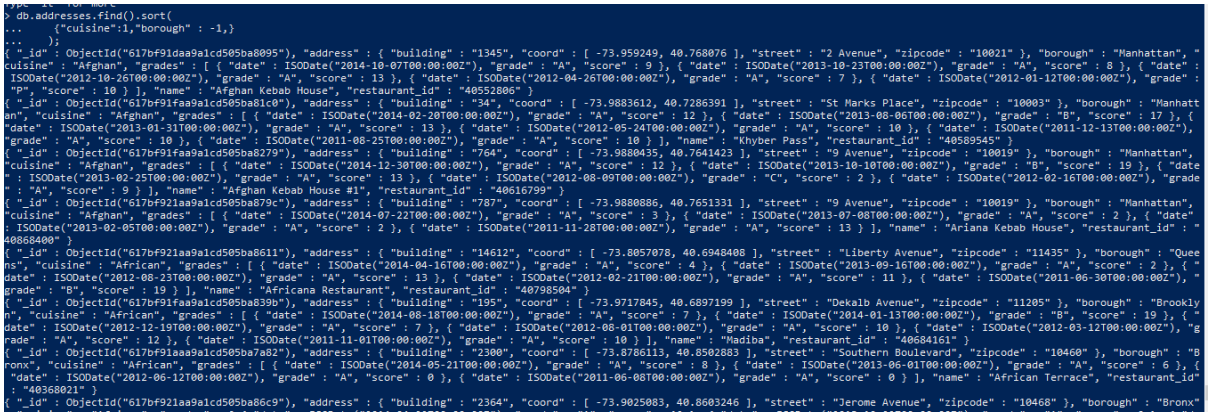
);

27. 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(

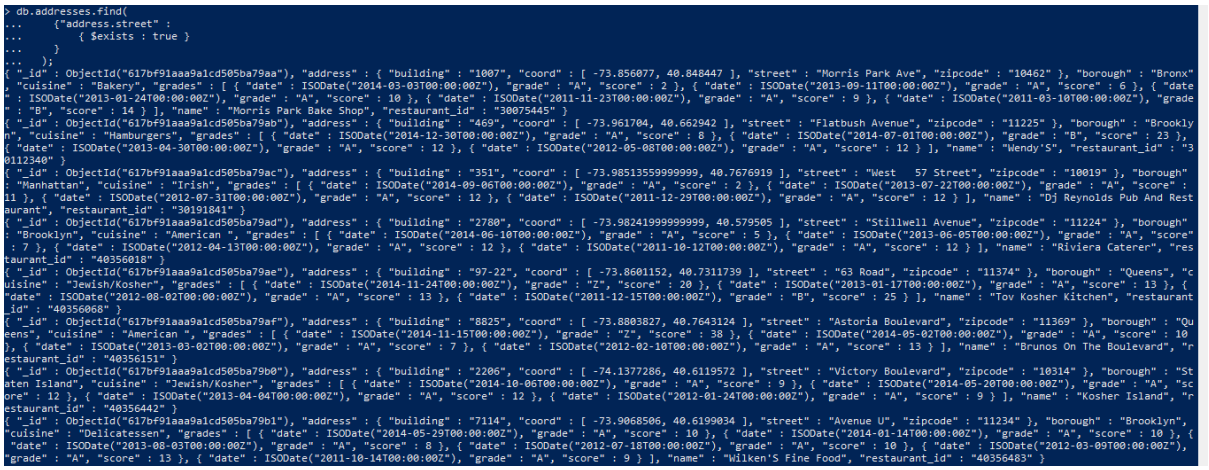
{"cuisine":1,"borough" : -1,}

);



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

```
db.addresses.find(
   {"address.street" :
      { $exists : true }
   }
 );
```



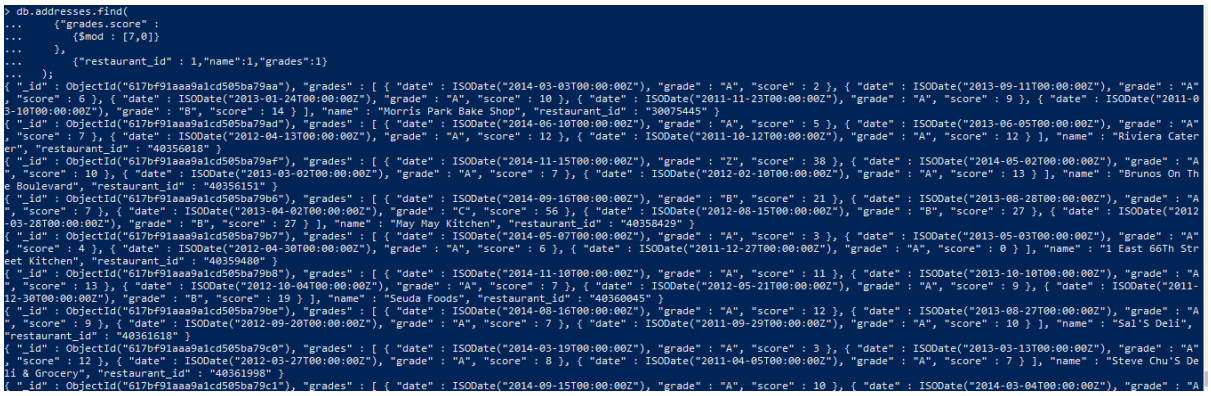
29. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

```
. db.addresses.find(
   {"address.coord" :
      {$type : 1}
   }
  );
```

30. 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

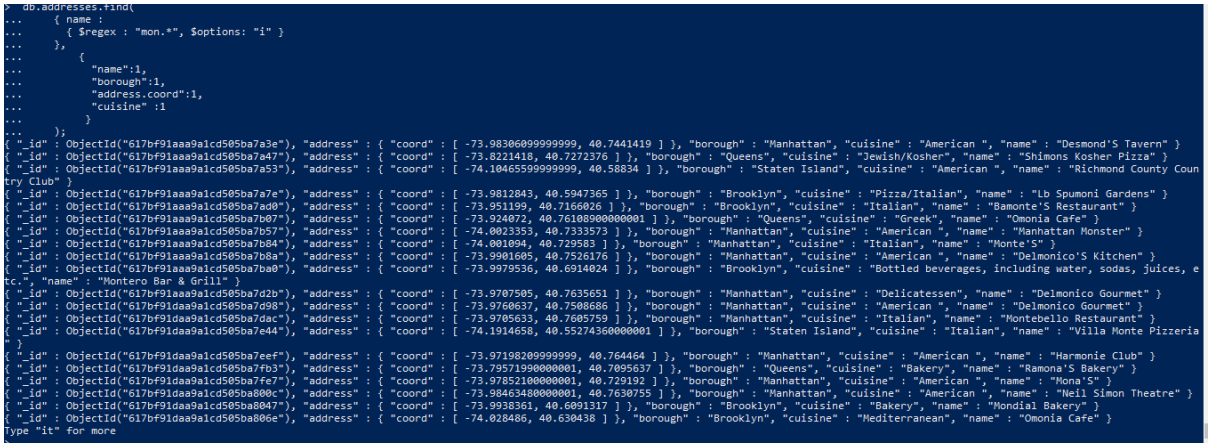
```
db.addresses.find(
```

```
{"grades.score" :
  {$mod : [7,0]}
},
  {"restaurant_id" : 1,"name":1,"grades":1}
);
```



31. 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
     }
  );
```



32. 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
      }
   );
```