Practical#8

Name: Saurin Anilkumar Prajapati

Roll No.: 19BCE239

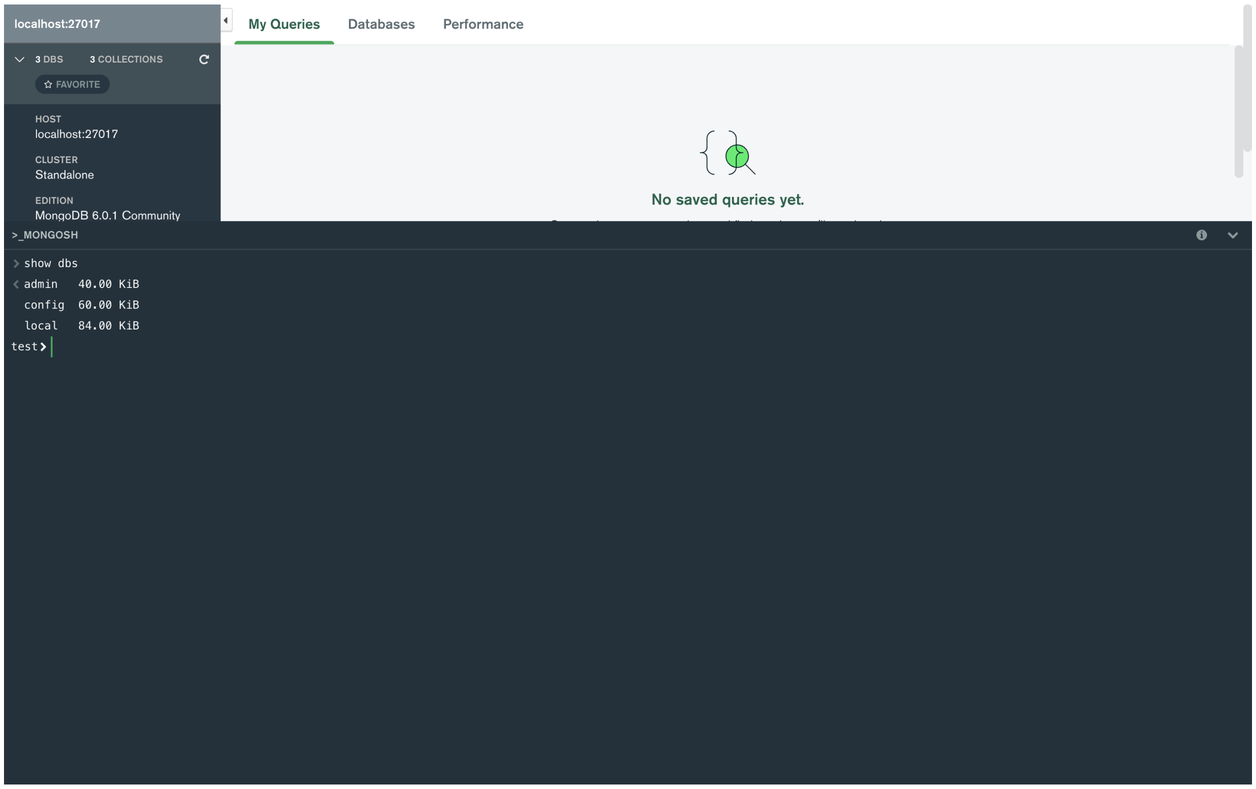
Course Code and Name: 2CS702 Big Data Analytics

Batch: D1

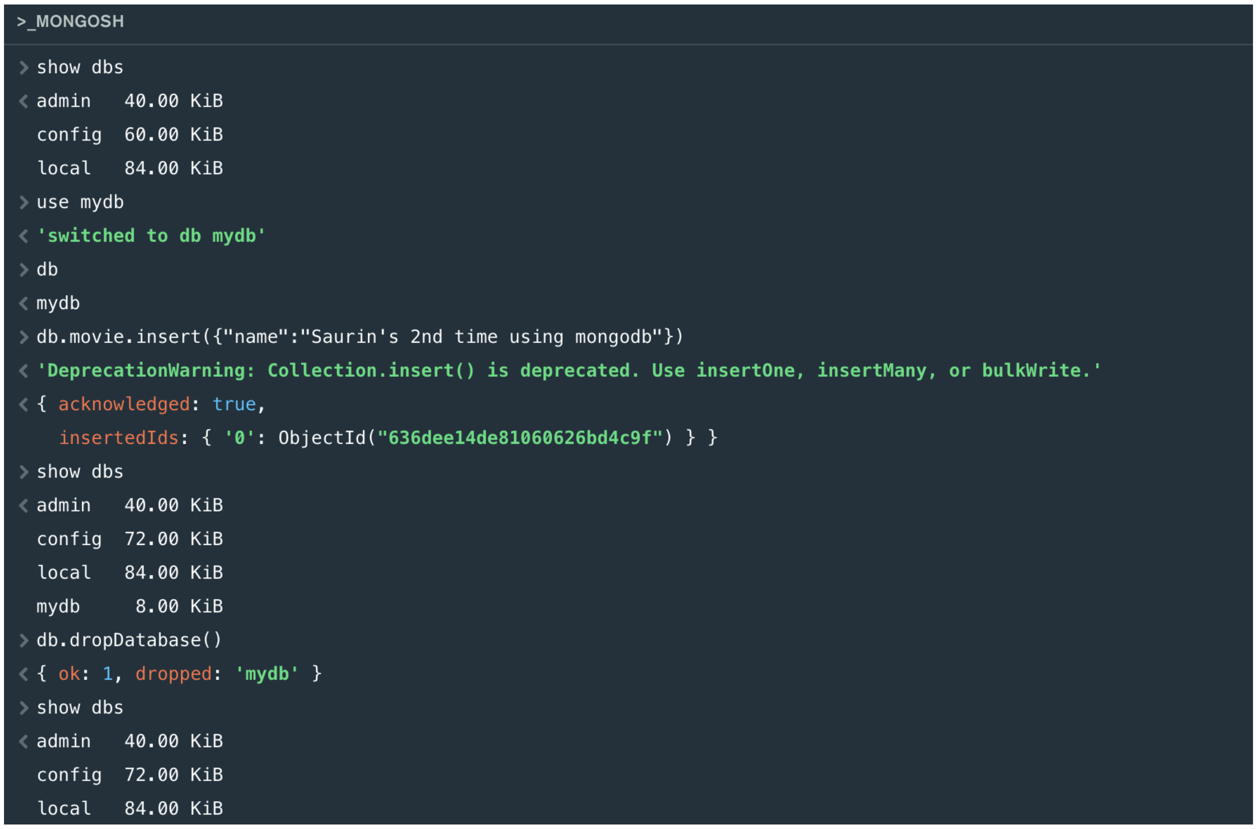
Aim:

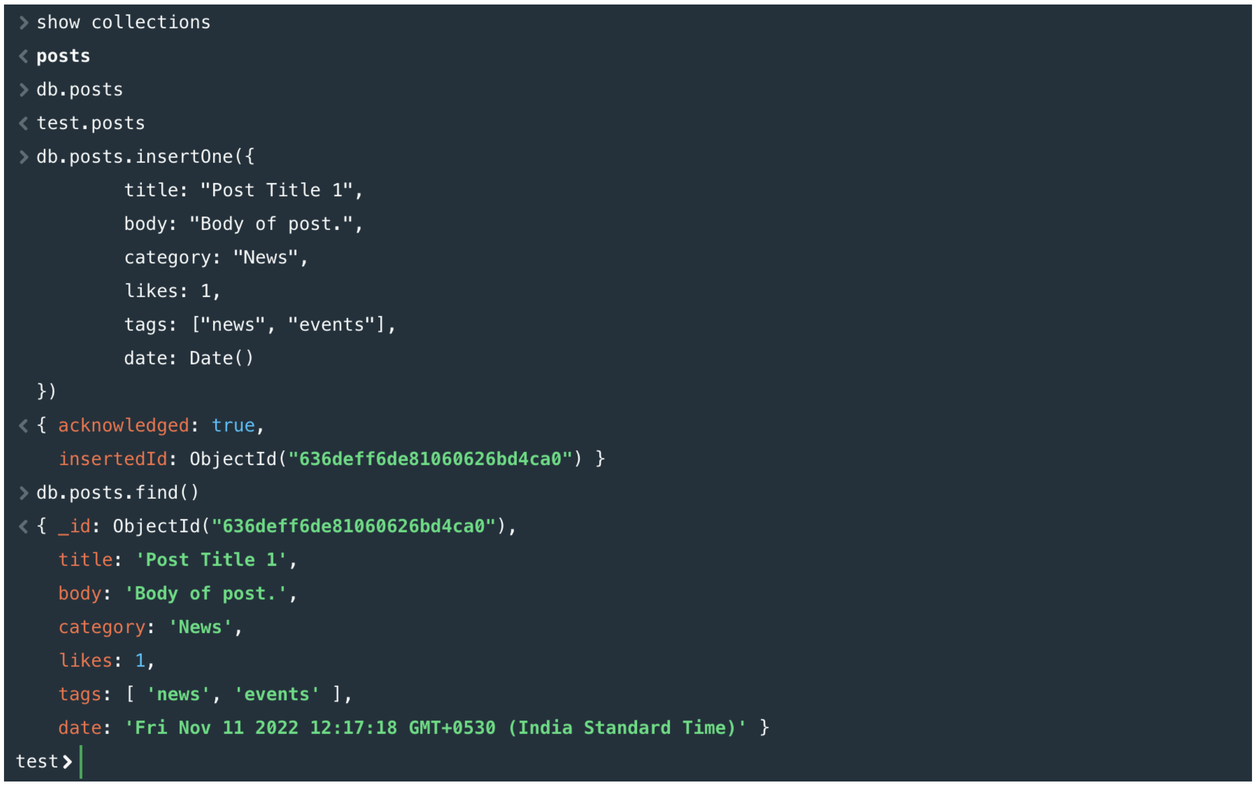
Setup MongoDB environment in your system. Import Restaurant Dataset and perform CRUD operation.

Basics

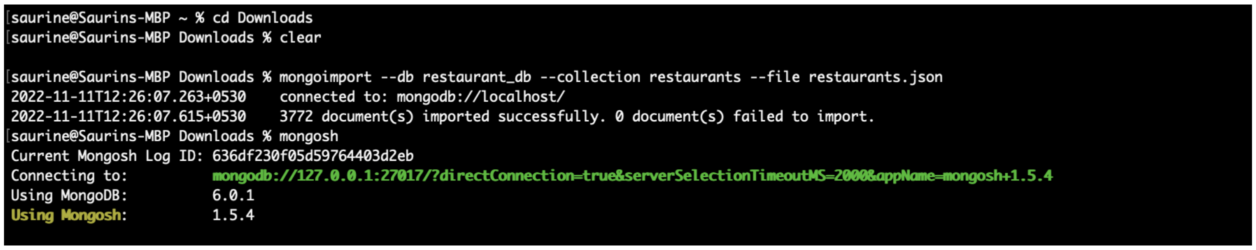


```
> show dbs
< admin   40.00 KiB
  config  60.00 KiB
  local   84.00 KiB
test>
```

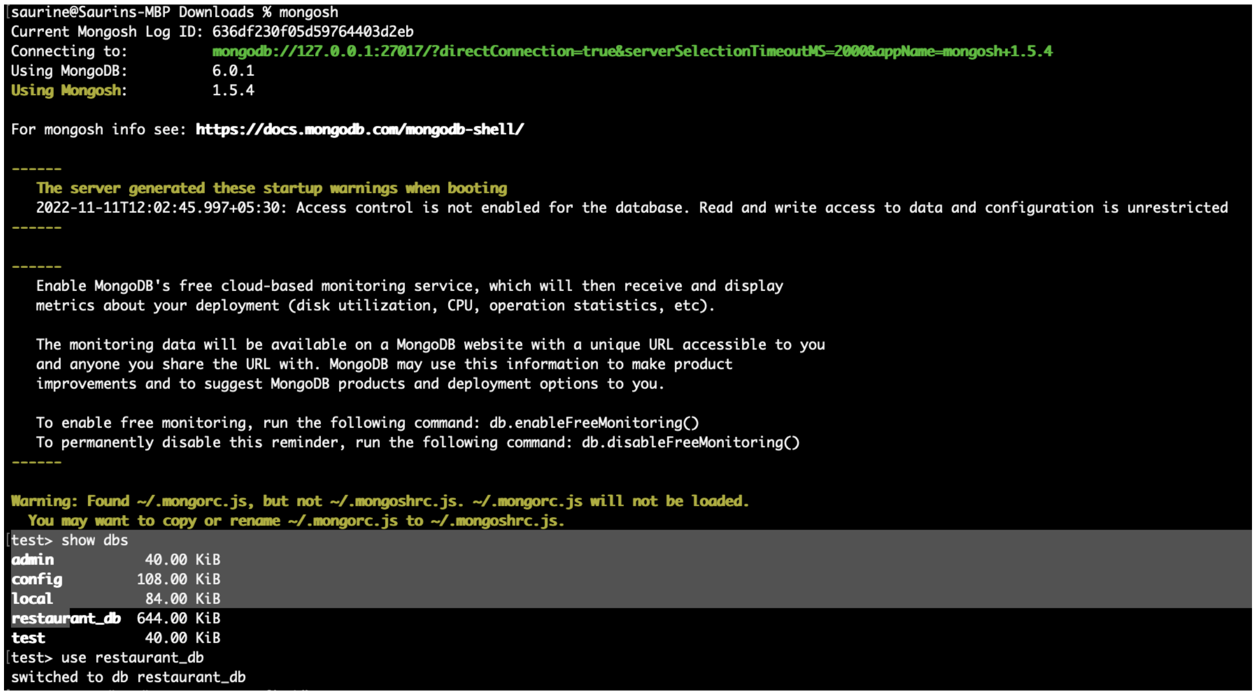

```
>_MONGOSH
> show dbs
< admin   40.00 KiB
  config  60.00 KiB
  local   84.00 KiB
> use mydb
< 'switched to db mydb'
> db
< mydb
> db.movie.insert({"name":"Saurin's 2nd time using mongodb"})
< 'DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.'
< { acknowledged: true,
    insertedIds: { '0': ObjectId("636dee14de81060626bd4c9f") } }
> show dbs
< admin   40.00 KiB
  config  72.00 KiB
  local   84.00 KiB
  mydb     8.00 KiB
> db.dropDatabase()
< { ok: 1, dropped: 'mydb' }
> show dbs
< admin   40.00 KiB
  config  72.00 KiB
  local   84.00 KiB
```



```
> show collections
< posts
> db.posts
< test.posts
> db.posts.insertOne({
          title: "Post Title 1",
          body: "Body of post.",
          category: "News",
          likes: 1,
          tags: ["news", "events"],
          date: Date()
  })
< { acknowledged: true,
    insertedId: ObjectId("636deff6de81060626bd4ca0") }
> db.posts.find()
< { _id: ObjectId("636deff6de81060626bd4ca0"),
    title: 'Post Title 1',
    body: 'Body of post.',
    category: 'News',
    likes: 1,
    tags: [ 'news', 'events' ],
    date: 'Fri Nov 11 2022 12:17:18 GMT+0530 (India Standard Time)' }
test>
```



```
saurine@Saurins-MBP ~ % cd Downloads
saurine@Saurins-MBP Downloads % clear

saurine@Saurins-MBP Downloads % mongoimport --db restaurant_db --collection restaurants --file restaurants.json
2022-11-11T12:26:07.263+0530    connected to: mongodb://localhost/
2022-11-11T12:26:07.615+0530    3772 document(s) imported successfully. 0 document(s) failed to import.
saurine@Saurins-MBP Downloads % mongosh
Current Mongosh Log ID: 636df230f05d59764403d2eb
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.5.4
Using MongoDB:          6.0.1
Using Mongosh:          1.5.4
```



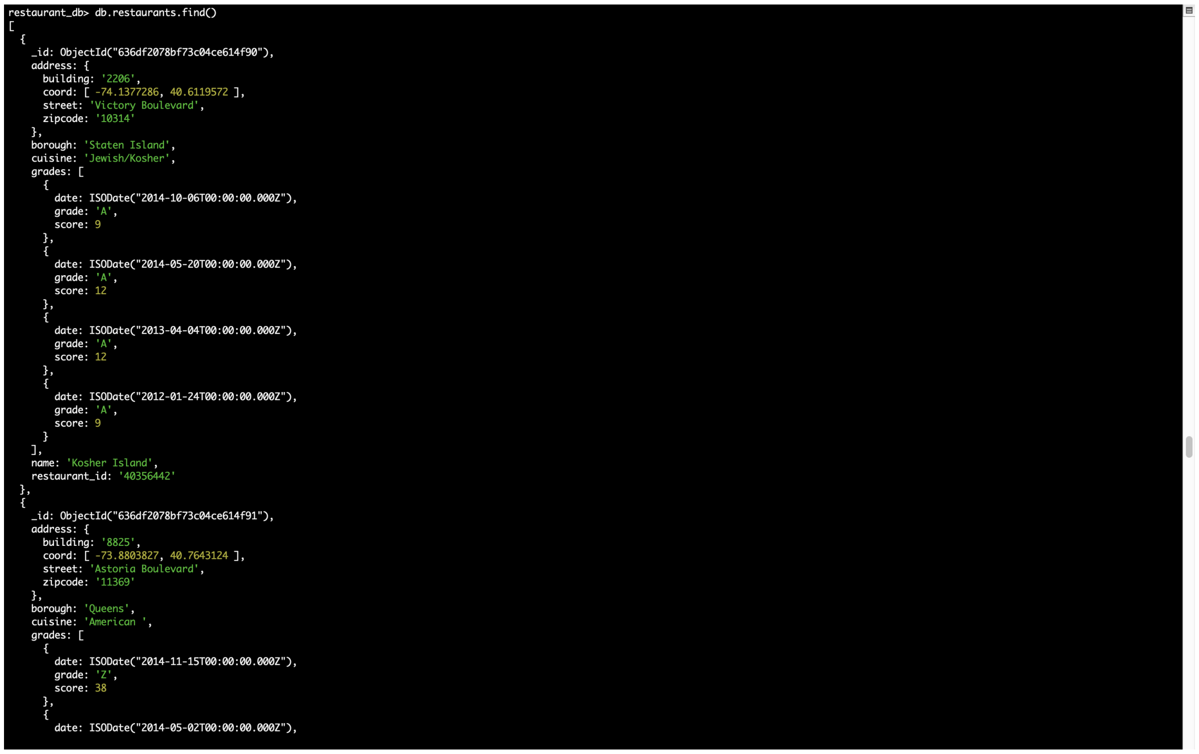
```
saurine@Saurins-MBP Downloads % mongosh
Current Mongosh Log ID: 636df230f05d59764403d2eb
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.5.4
Using MongoDB:          6.0.1
Using Mongosh:          1.5.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-11-11T12:02:45.997+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
test> show dbs
admin           40.00 KiB
config         108.00 KiB
local           84.00 KiB
restaurant_db  644.00 KiB
test            40.00 KiB
test> use restaurant_db
switched to db restaurant_db
```

Exercise

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

   Command:

   ```
   db.restaurants.find()
   ```

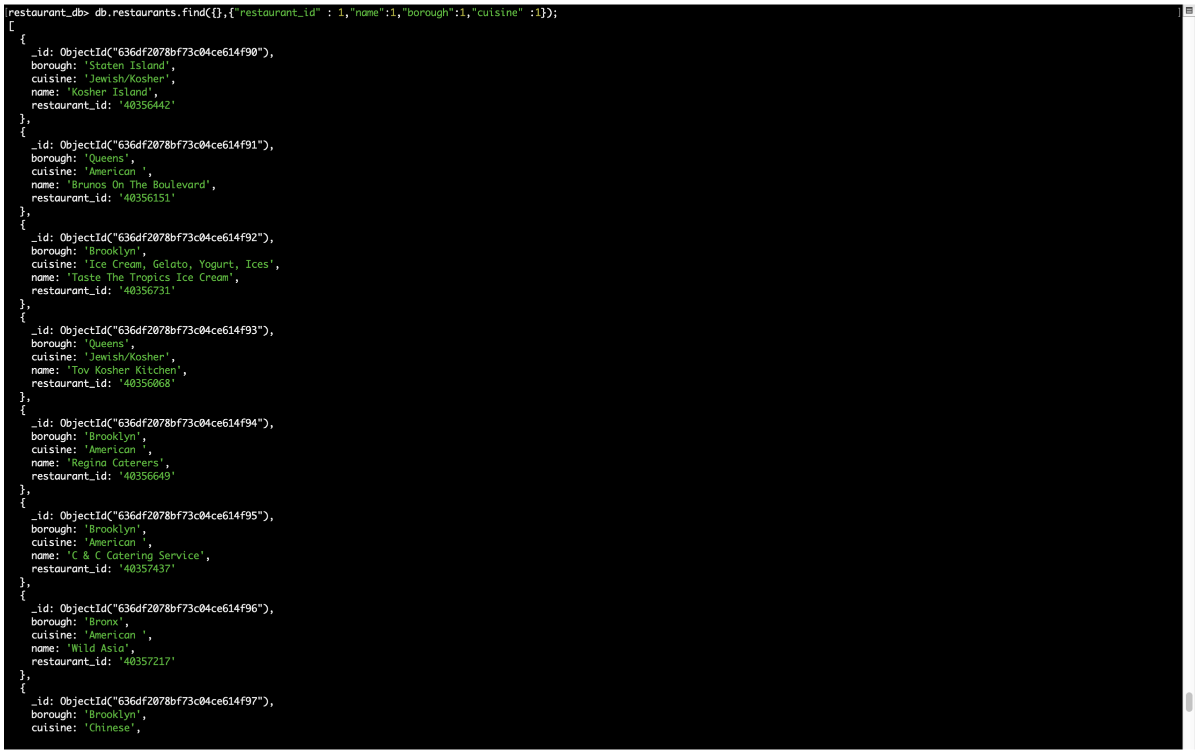


```
restaurant_db> db.restaurants.find()
[
  {
    _id: ObjectId("636df2078bf73c04ce614f90"),
    address: {
      building: '2206',
      coord: [ -74.1377286, 40.6119572 ],
      street: 'Victory Boulevard',
      zipcode: '10314'
    },
    borough: 'Staten Island',
    cuisine: 'Jewish/Kosher',
    grades: [
      {
        date: ISODate("2014-10-06T00:00:00.000Z"),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate("2014-05-20T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate("2013-04-04T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate("2012-01-24T00:00:00.000Z"),
        grade: 'A',
        score: 9
      }
    ],
    name: 'Kosher Island',
    restaurant_id: '40356442'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f91"),
    address: {
      building: '8825',
      coord: [ -73.8803827, 40.7643124 ],
      street: 'Astoria Boulevard',
      zipcode: '11369'
    },
    borough: 'Queens',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-11-15T00:00:00.000Z"),
        grade: 'Z',
        score: 38
      },
      {
        date: ISODate("2014-05-02T00:00:00.000Z"),
```

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

   Command:

```
db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1})
```

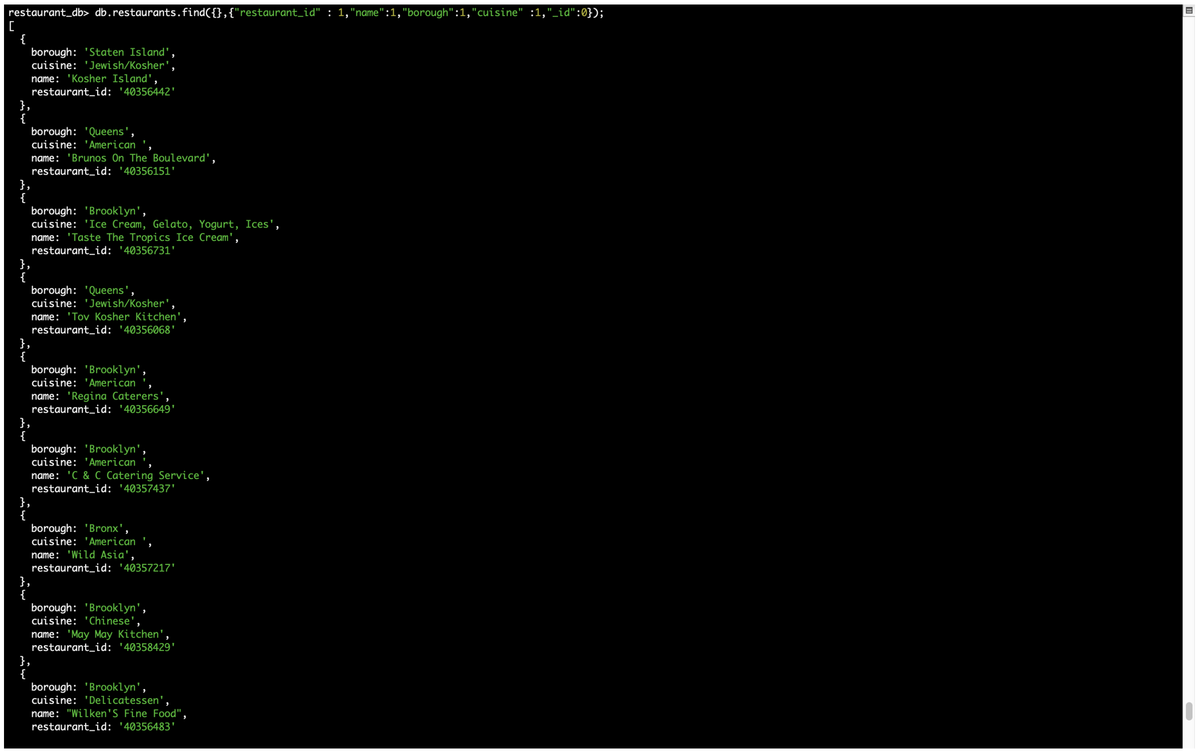


```
restaurant_db> db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1});
[
  {
    _id: ObjectId("636df2078bf73c04ce614f90"),
    borough: 'Staten Island',
    cuisine: 'Jewish/Kosher',
    name: 'Kosher Island',
    restaurant_id: '40356442'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f91"),
    borough: 'Queens',
    cuisine: 'American ',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f92"),
    borough: 'Brooklyn',
    cuisine: 'Ice Cream, Gelato, Yogurt, Ices',
    name: 'Taste The Tropics Ice Cream',
    restaurant_id: '40356731'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f93"),
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f94"),
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'Regina Caterers',
    restaurant_id: '40356649'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f95"),
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'C & C Catering Service',
    restaurant_id: '40357437'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f96"),
    borough: 'Bronx',
    cuisine: 'American ',
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f97"),
    borough: 'Brooklyn',
    cuisine: 'Chinese',
```

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.

   Command:

```
db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1,"_id":
0})
```

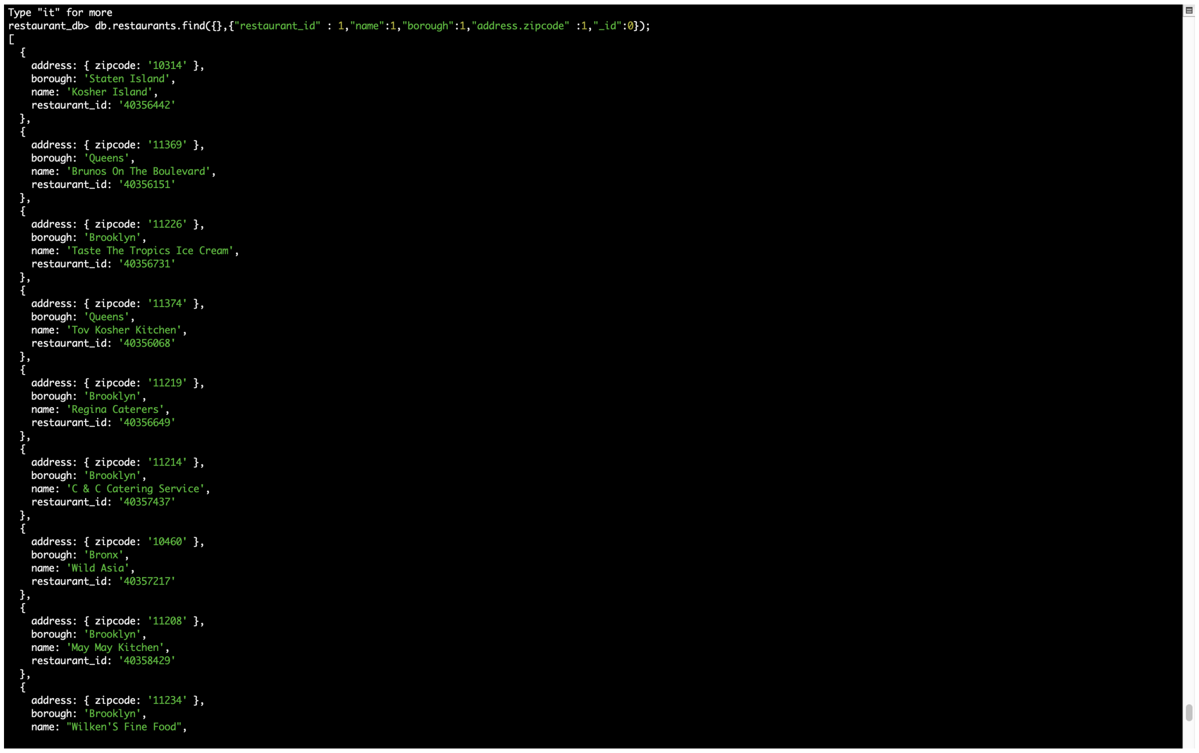


```
restaurant_db> db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1,"_id":0});
[
  {
    borough: 'Staten Island',
    cuisine: 'Jewish/Kosher',
    name: 'Kosher Island',
    restaurant_id: '40356442'
  },
  {
    borough: 'Queens',
    cuisine: 'American ',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'Ice Cream, Gelato, Yogurt, Ices',
    name: 'Taste The Tropics Ice Cream',
    restaurant_id: '40356731'
  },
  {
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'Regina Caterers',
    restaurant_id: '40356649'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'C & C Catering Service',
    restaurant_id: '40357437'
  },
  {
    borough: 'Bronx',
    cuisine: 'American ',
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'Chinese',
    name: 'May May Kitchen',
    restaurant_id: '40358429'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'Delicatessen',
    name: "Wilken'S Fine Food",
    restaurant_id: '40356483'
```

4. 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.

   Command:

   ```
   db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :
   1,"_id":0})
   ```

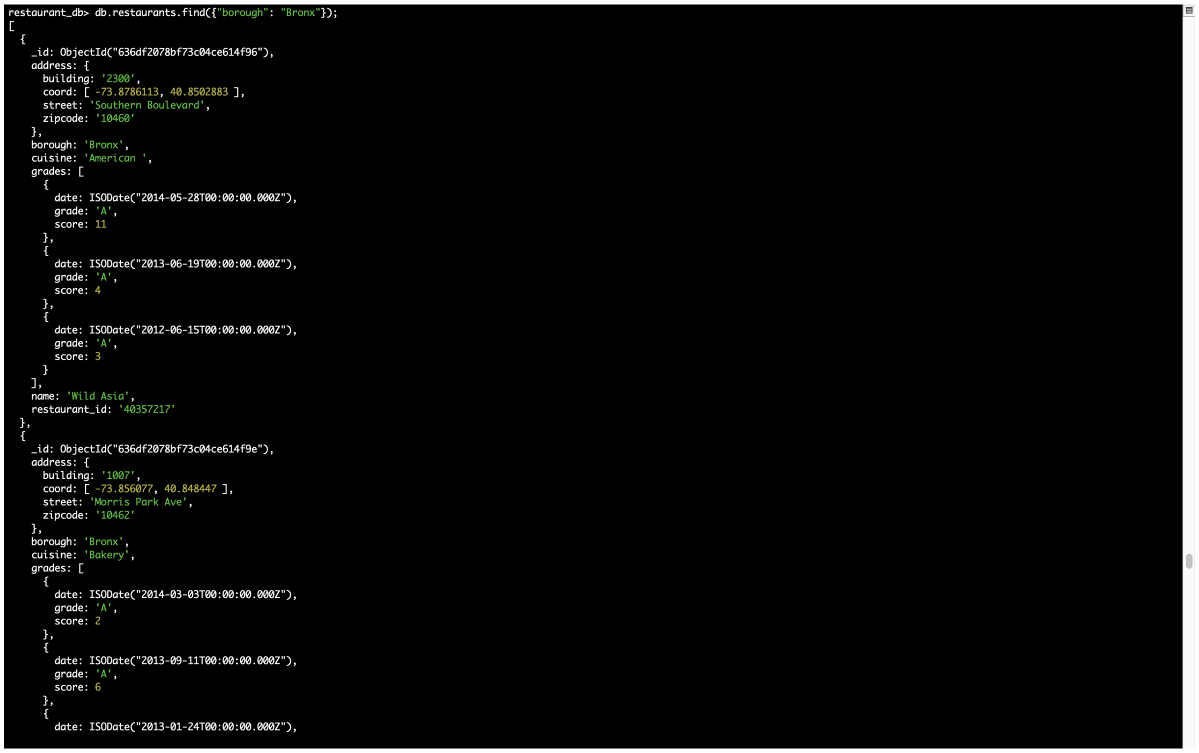


```
Type "it" for more
restaurant_db> db.restaurants.find({},{"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"_id":0});
[
  {
    address: { zipcode: '10314' },
    borough: 'Staten Island',
    name: 'Kosher Island',
    restaurant_id: '40356442'
  },
  {
    address: { zipcode: '11369' },
    borough: 'Queens',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  },
  {
    address: { zipcode: '11226' },
    borough: 'Brooklyn',
    name: 'Taste The Tropics Ice Cream',
    restaurant_id: '40356731'
  },
  {
    address: { zipcode: '11374' },
    borough: 'Queens',
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
  },
  {
    address: { zipcode: '11219' },
    borough: 'Brooklyn',
    name: 'Regina Caterers',
    restaurant_id: '40356649'
  },
  {
    address: { zipcode: '11214' },
    borough: 'Brooklyn',
    name: 'C & C Catering Service',
    restaurant_id: '40357437'
  },
  {
    address: { zipcode: '10460' },
    borough: 'Bronx',
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
  {
    address: { zipcode: '11208' },
    borough: 'Brooklyn',
    name: 'May May Kitchen',
    restaurant_id: '40358429'
  },
  {
    address: { zipcode: '11234' },
    borough: 'Brooklyn',
    name: "Wilken'S Fine Food",
```

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

Command:

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

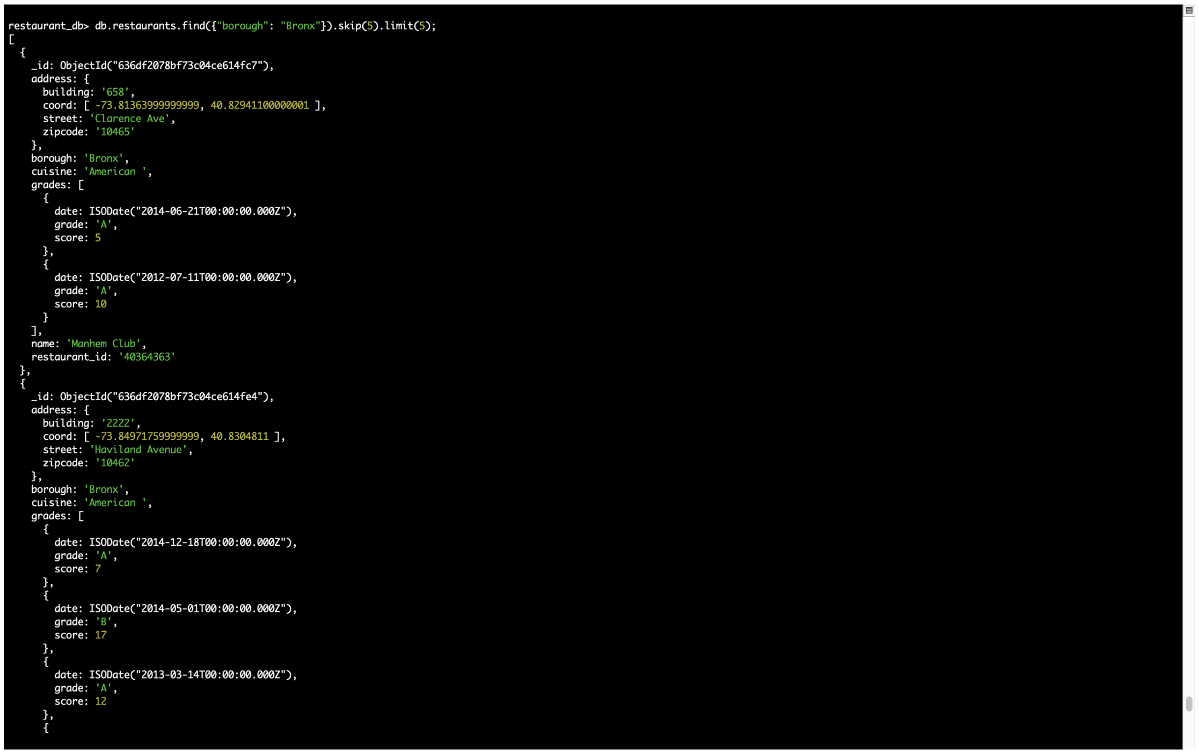


```
restaurant_db> db.restaurants.find({"borough": "Bronx"});
[
  {
    _id: ObjectId("636df2078bf73c04ce614f96"),
    address: {
      building: '2300',
      coord: [ -73.8786113, 40.8502883 ],
      street: 'Southern Boulevard',
      zipcode: '10460'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-05-28T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2013-06-19T00:00:00.000Z"),
        grade: 'A',
        score: 4
      },
      {
        date: ISODate("2012-06-15T00:00:00.000Z"),
        grade: 'A',
        score: 3
      }
    ],
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614f9e"),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate("2014-03-03T00:00:00.000Z"),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate("2013-09-11T00:00:00.000Z"),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate("2013-01-24T00:00:00.000Z"),
```

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

   Command:

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

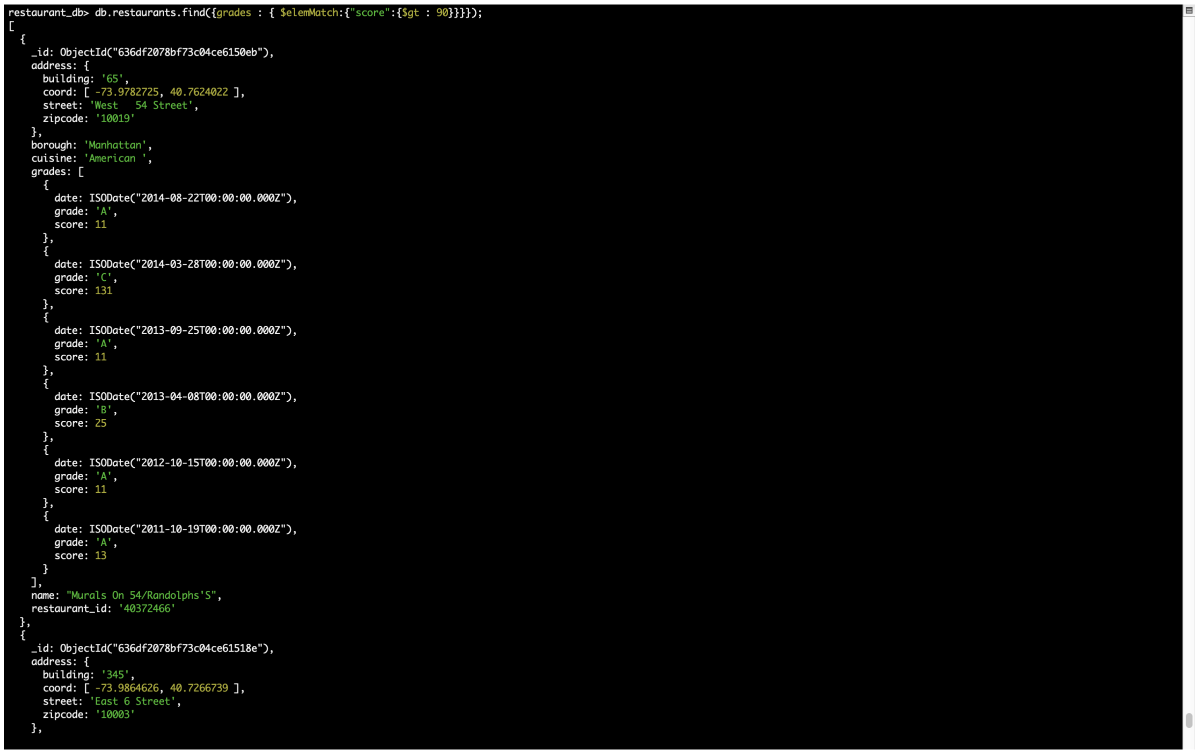


```
restaurant_db> db.restaurants.find({"borough": "Bronx"}).skip(5).limit(5);
[
  {
    _id: ObjectId("636df2078bf73c04ce614fc7"),
    address: {
      building: '658',
      coord: [ -73.81363999999999, 40.82941100000001 ],
      street: 'Clarence Ave',
      zipcode: '10465'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-06-21T00:00:00.000Z"),
        grade: 'A',
        score: 5
      },
      {
        date: ISODate("2012-07-11T00:00:00.000Z"),
        grade: 'A',
        score: 10
      }
    ],
    name: 'Manhem Club',
    restaurant_id: '40364363'
  },
  {
    _id: ObjectId("636df2078bf73c04ce614fe4"),
    address: {
      building: '2222',
      coord: [ -73.84971759999999, 40.8304811 ],
      street: 'Haviland Avenue',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-12-18T00:00:00.000Z"),
        grade: 'A',
        score: 7
      },
      {
        date: ISODate("2014-05-01T00:00:00.000Z"),
        grade: 'B',
        score: 17
      },
      {
        date: ISODate("2013-03-14T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
```

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

   Command:

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

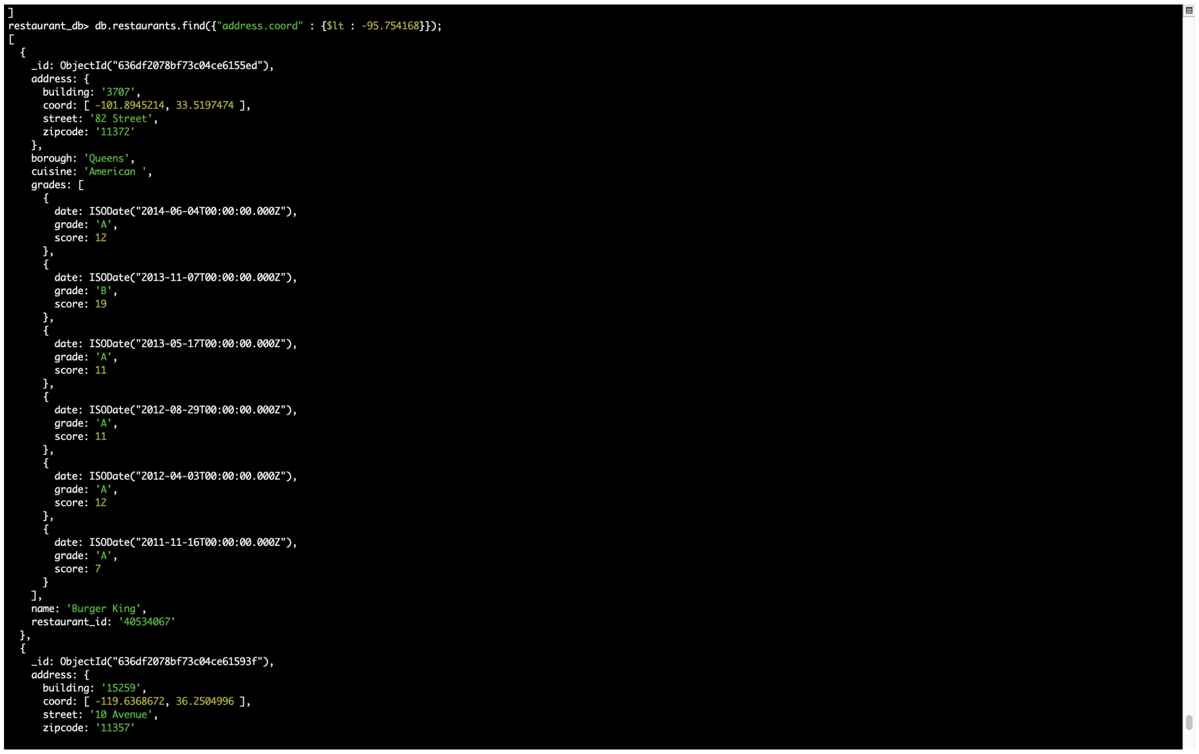


```
restaurant_db> db.restaurants.find({grades : { $elemMatch:{"score":{$gt : 90}}}});
[
  {
    _id: ObjectId("636df2078bf73c04ce6150eb"),
    address: {
      building: '65',
      coord: [ -73.9782725, 40.7624022 ],
      street: 'West   54 Street',
      zipcode: '10019'
    },
    borough: 'Manhattan',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-08-22T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2014-03-28T00:00:00.000Z"),
        grade: 'C',
        score: 131
      },
      {
        date: ISODate("2013-09-25T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2013-04-08T00:00:00.000Z"),
        grade: 'B',
        score: 25
      },
      {
        date: ISODate("2012-10-15T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2011-10-19T00:00:00.000Z"),
        grade: 'A',
        score: 13
      }
    ],
    name: "Murals On 54/Randolphs'S",
    restaurant_id: '40372466'
  },
  {
    _id: ObjectId("636df2078bf73c04ce61518e"),
    address: {
      building: '345',
      coord: [ -73.9864626, 40.7266739 ],
      street: 'East 6 Street',
      zipcode: '10003'
    },
```

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

   Command:

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

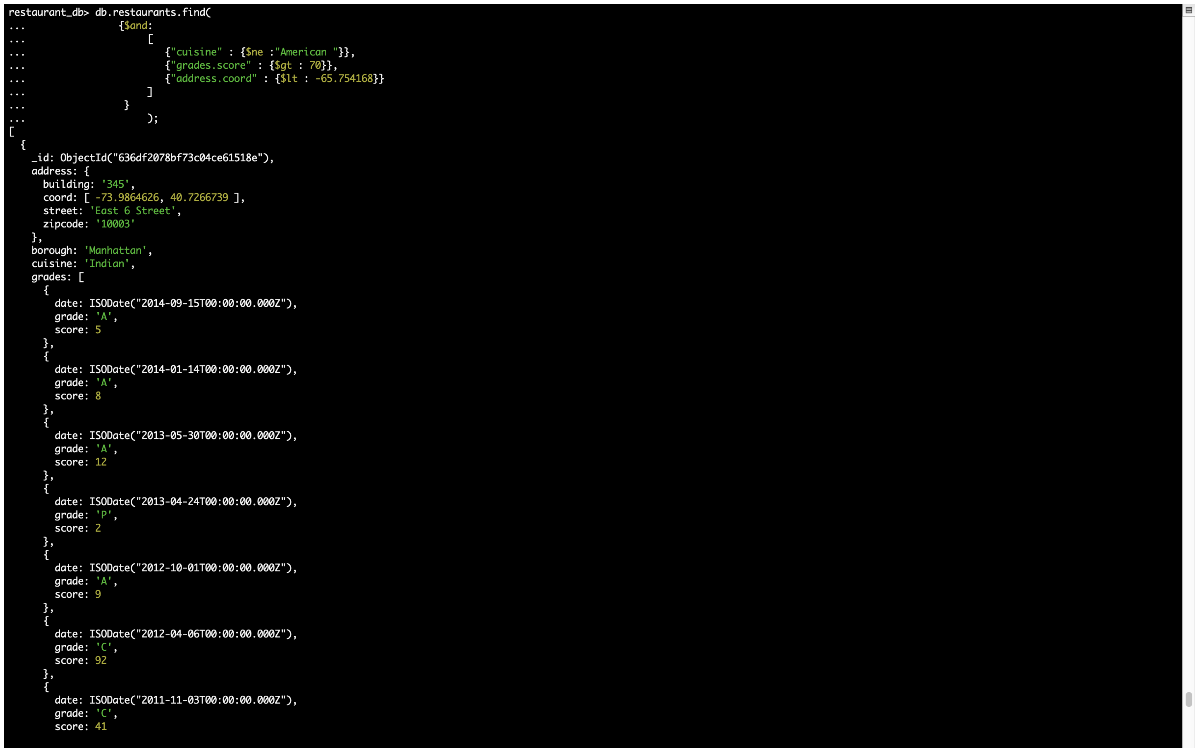


```
]
restaurant_db> db.restaurants.find({"address.coord" : {$lt : -95.754168}});
[
  {
    _id: ObjectId("636df2078bf73c04ce6155ed"),
    address: {
      building: '3707',
      coord: [ -101.8945214, 33.5197474 ],
      street: '82 Street',
      zipcode: '11372'
    },
    borough: 'Queens',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-06-04T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate("2013-11-07T00:00:00.000Z"),
        grade: 'B',
        score: 19
      },
      {
        date: ISODate("2013-05-17T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2012-08-29T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2012-04-03T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate("2011-11-16T00:00:00.000Z"),
        grade: 'A',
        score: 7
      }
    ],
    name: 'Burger King',
    restaurant_id: '40534067'
  },
  {
    _id: ObjectId("636df2078bf73c04ce61593f"),
    address: {
      building: '15259',
      coord: [ -119.6368672, 36.2504996 ],
      street: '10 Avenue',
      zipcode: '11357'
```

4. 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.

   Command:

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

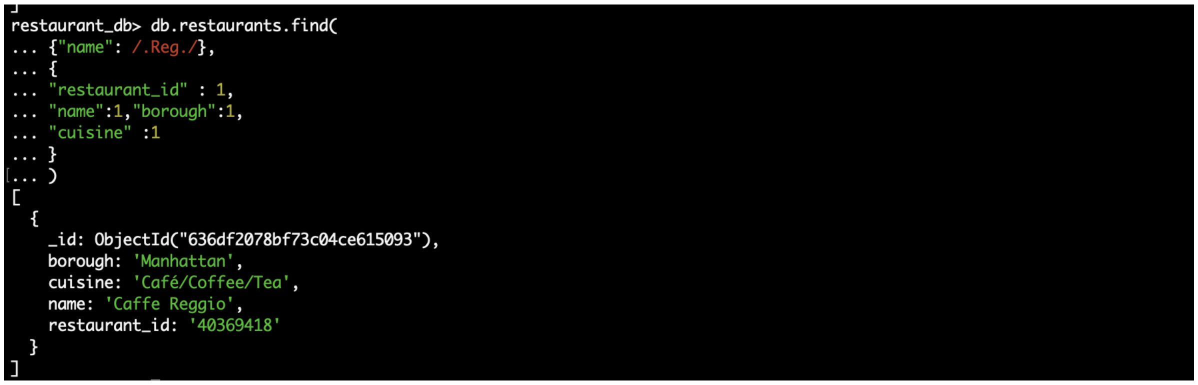


```
restaurant_db> db.restaurants.find(
...                 {$and:
...                     [
...                         {"cuisine" : {$ne :"American "}},
...                         {"grades.score" : {$gt : 70}},
...                         {"address.coord" : {$lt : -65.754168}}
...                     ]
...                  }
...                     );
[
  {
    _id: ObjectId("636df2078bf73c04ce61518e"),
    address: {
      building: '345',
      coord: [ -73.9864626, 40.7266739 ],
      street: 'East 6 Street',
      zipcode: '10003'
    },
    borough: 'Manhattan',
    cuisine: 'Indian',
    grades: [
      {
        date: ISODate("2014-09-15T00:00:00.000Z"),
        grade: 'A',
        score: 5
      },
      {
        date: ISODate("2014-01-14T00:00:00.000Z"),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate("2013-05-30T00:00:00.000Z"),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate("2013-04-24T00:00:00.000Z"),
        grade: 'P',
        score: 2
      },
      {
        date: ISODate("2012-10-01T00:00:00.000Z"),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate("2012-04-06T00:00:00.000Z"),
        grade: 'C',
        score: 92
      },
      {
        date: ISODate("2011-11-03T00:00:00.000Z"),
        grade: 'C',
        score: 41
```

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

   Command:

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



```
restaurant_db> db.restaurants.find(
... {"name": /.Reg./},
... {
... "restaurant_id" : 1,
... "name":1,"borough":1,
... "cuisine" :1
... }
... )
[
  {
    _id: ObjectId("636df2078bf73c04ce615093"),
    borough: 'Manhattan',
    cuisine: 'Café/Coffee/Tea',
    name: 'Caffe Reggio',
    restaurant_id: '40369418'
  }
]
```

Thank You