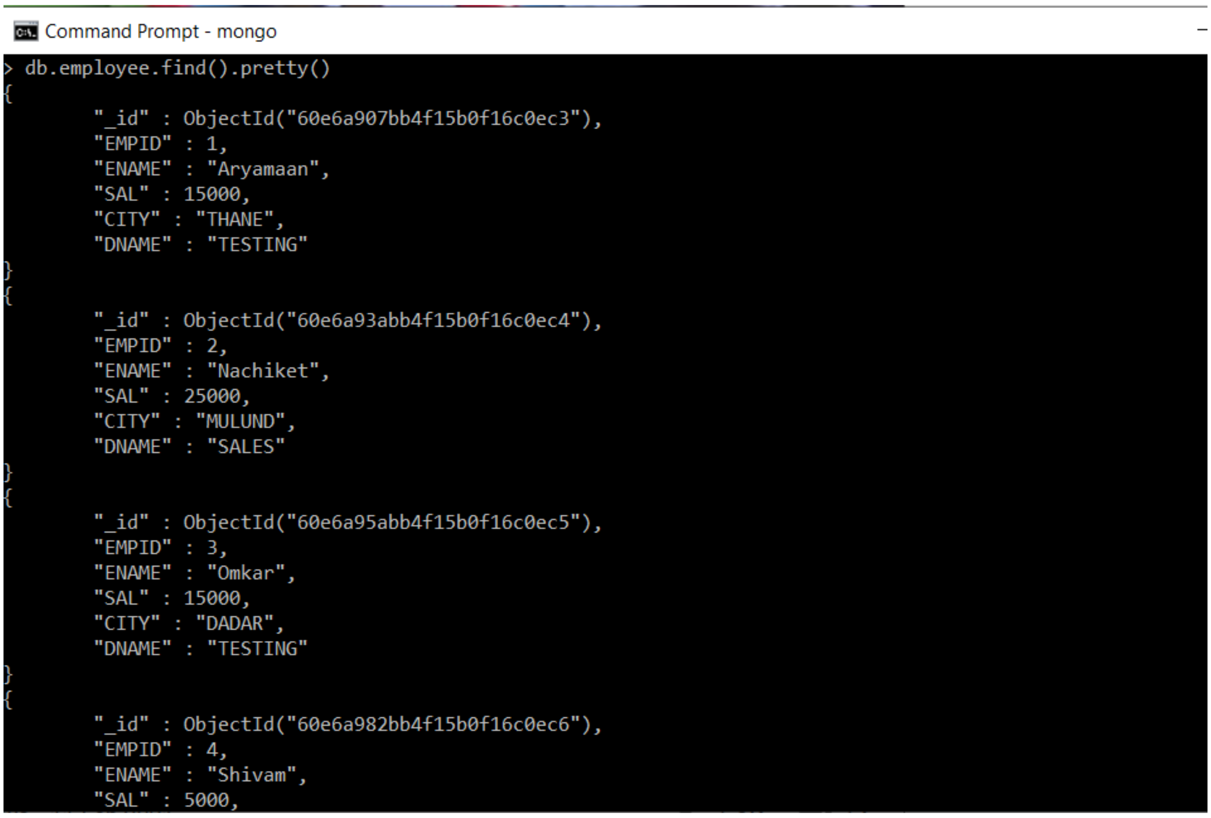
Practical 1-Basic Queries

Date: 08-07-2021

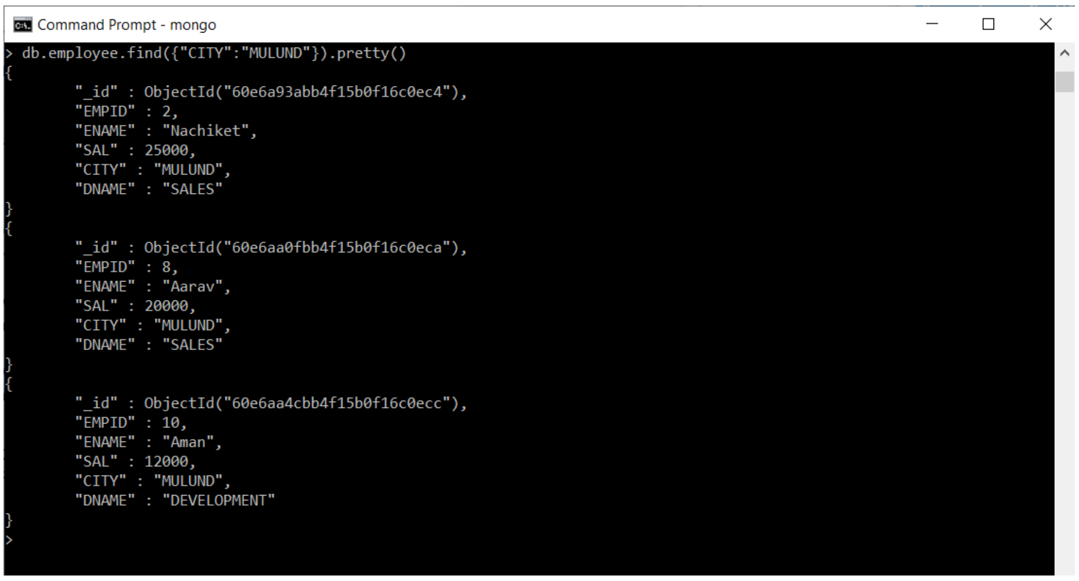
Q.1. Insert and display the documents in the Employee collection.



Command Prompt - mongo

```
> db.employee.find().pretty()
{
        "_id" : ObjectId("60e6a907bb4f15b0f16c0ec3"),
        "EMPID" : 1,
        "ENAME" : "Aryamaan",
        "SAL" : 15000,
        "CITY" : "THANE",
        "DNAME" : "TESTING"
}
{
        "_id" : ObjectId("60e6a93abb4f15b0f16c0ec4"),
        "EMPID" : 2,
        "ENAME" : "Nachiket",
        "SAL" : 25000,
        "CITY" : "MULUND",
        "DNAME" : "SALES"
}
{
        "_id" : ObjectId("60e6a95abb4f15b0f16c0ec5"),
        "EMPID" : 3,
        "ENAME" : "Omkar",
        "SAL" : 15000,
        "CITY" : "DADAR",
        "DNAME" : "TESTING"
}
{
        "_id" : ObjectId("60e6a982bb4f15b0f16c0ec6"),
        "EMPID" : 4,
        "ENAME" : "Shivam",
        "SAL" : 5000,
```

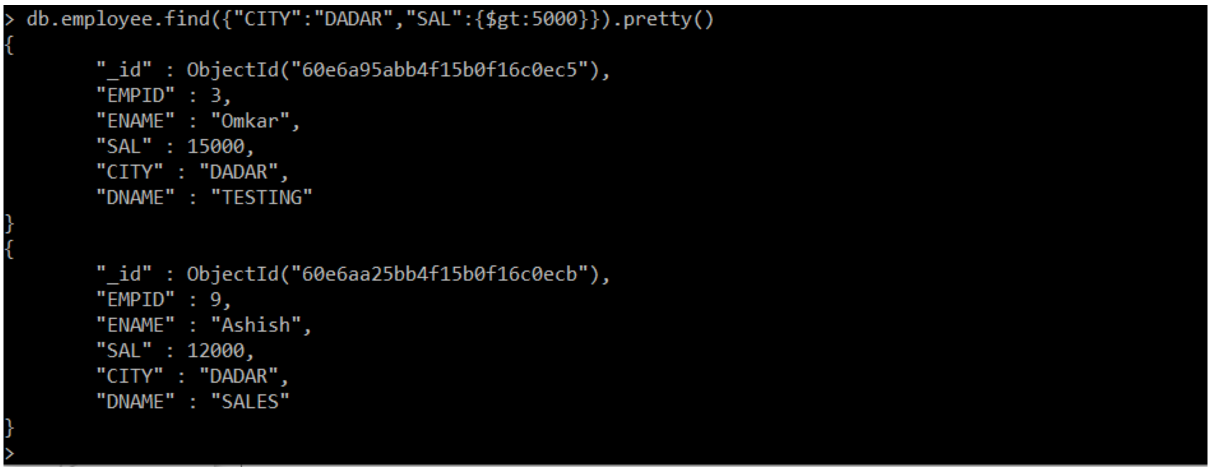
Q.2 Display details of employees who live in Mulund.



Command Prompt - mongo

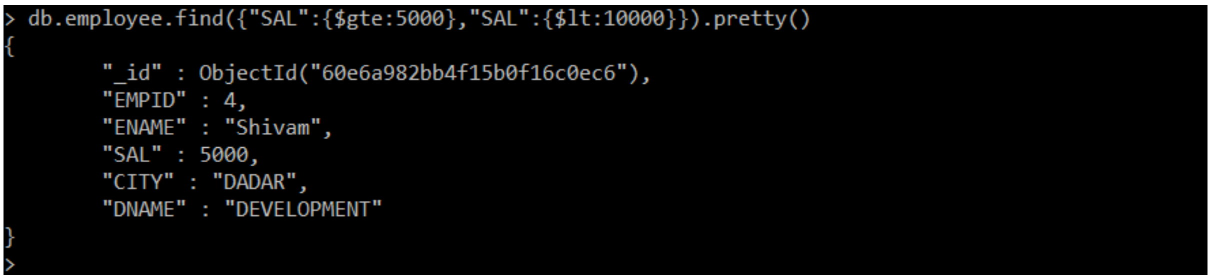
```
> db.employee.find({"CITY":"MULUND"}).pretty()
{
        "_id" : ObjectId("60e6a93abb4f15b0f16c0ec4"),
        "EMPID" : 2,
        "ENAME" : "Nachiket",
        "SAL" : 25000,
        "CITY" : "MULUND",
        "DNAME" : "SALES"
}
{
        "_id" : ObjectId("60e6aa0fbb4f15b0f16c0eca"),
        "EMPID" : 8,
        "ENAME" : "Aarav",
        "SAL" : 20000,
        "CITY" : "MULUND",
        "DNAME" : "SALES"
}
{
        "_id" : ObjectId("60e6aa4cbb4f15b0f16c0ecc"),
        "EMPID" : 10,
        "ENAME" : "Aman",
        "SAL" : 12000,
        "CITY" : "MULUND",
        "DNAME" : "DEVELOPMENT"
}
>
```

Q.3 Display details of employee staying in dadar and salary greater than 5000.



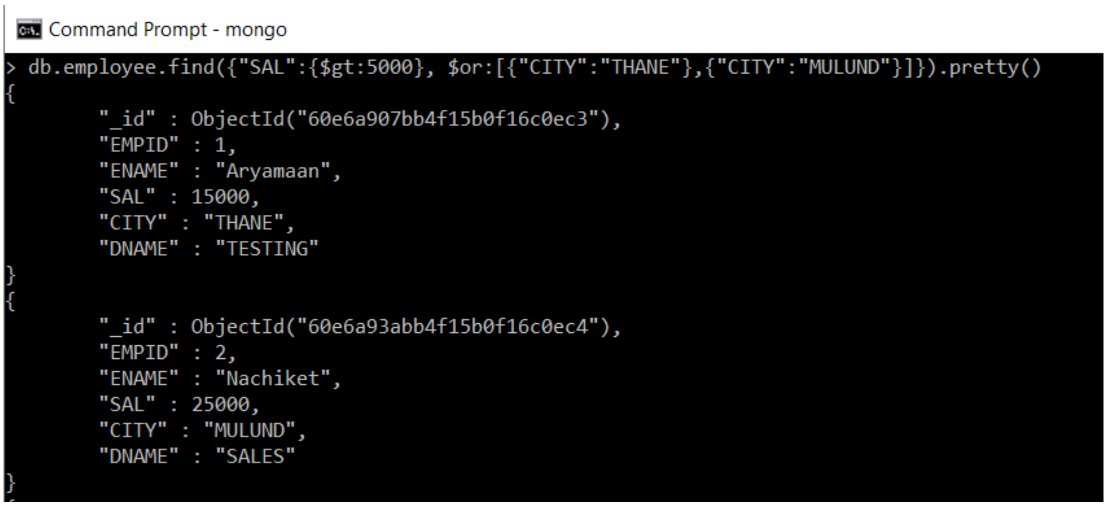
```
> db.employee.find({"CITY":"DADAR","SAL":{$gt:5000}}).pretty()
{
        "_id" : ObjectId("60e6a95abb4f15b0f16c0ec5"),
        "EMPID" : 3,
        "ENAME" : "Omkar",
        "SAL" : 15000,
        "CITY" : "DADAR",
        "DNAME" : "TESTING"
}
{
        "_id" : ObjectId("60e6aa25bb4f15b0f16c0ecb"),
        "EMPID" : 9,
        "ENAME" : "Ashish",
        "SAL" : 12000,
        "CITY" : "DADAR",
        "DNAME" : "SALES"
}
>
```

Q.4 Display details of the employee whose salary is greater than equal 5000 but less than 10000.



```
> db.employee.find({"SAL":{$gte:5000},"SAL":{$lt:10000}}).pretty()
{
        "_id" : ObjectId("60e6a982bb4f15b0f16c0ec6"),
        "EMPID" : 4,
        "ENAME" : "Shivam",
        "SAL" : 5000,
        "CITY" : "DADAR",
        "DNAME" : "DEVELOPMENT"
}
>
```

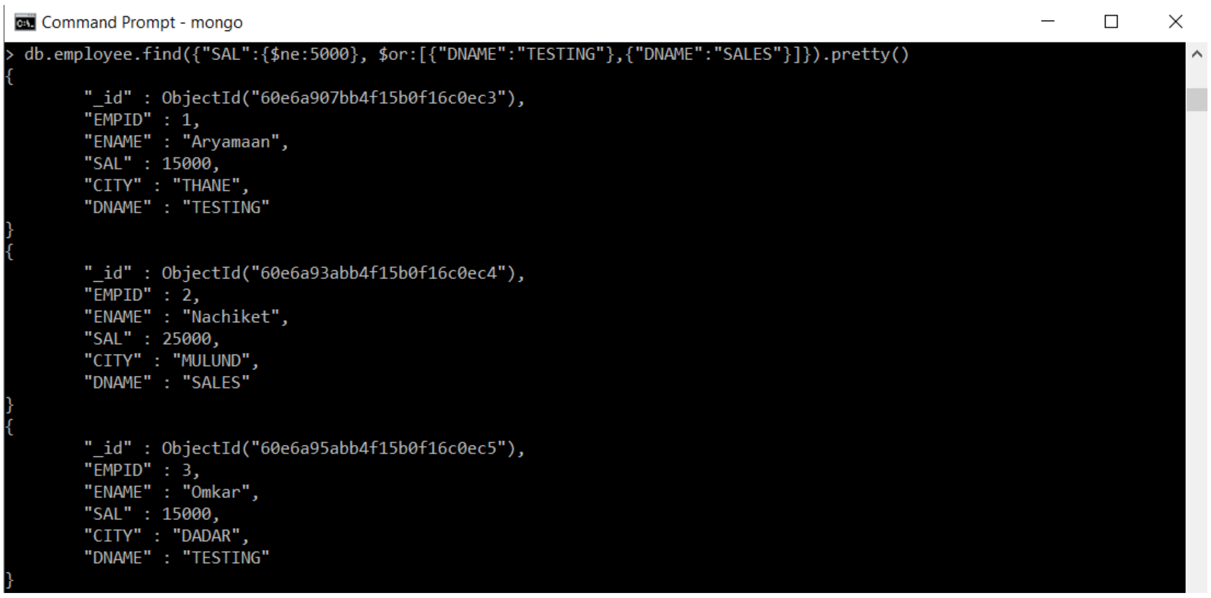
Q.5 Display details of employees staying in either thane or mulund and salary greater than 5000.



Command Prompt - mongo

```
> db.employee.find({"SAL":{$gt:5000}, $or:[{"CITY":"THANE"},{"CITY":"MULUND"}]}).pretty()
{
        "_id" : ObjectId("60e6a907bb4f15b0f16c0ec3"),
        "EMPID" : 1,
        "ENAME" : "Aryamaan",
        "SAL" : 15000,
        "CITY" : "THANE",
        "DNAME" : "TESTING"
}
{
        "_id" : ObjectId("60e6a93abb4f15b0f16c0ec4"),
        "EMPID" : 2,
        "ENAME" : "Nachiket",
        "SAL" : 25000,
        "CITY" : "MULUND",
        "DNAME" : "SALES"
}
```

Q.6. Display details of employee whose salary is not equal to 5000 and working in either testing or sales department.



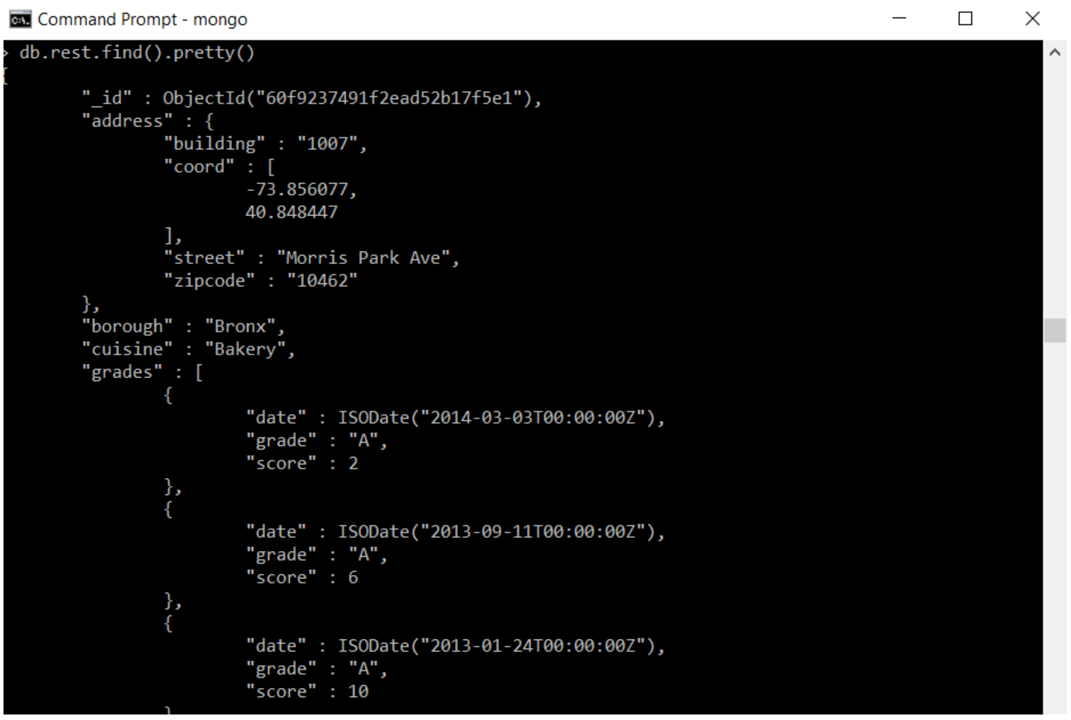
Command Prompt - mongo

```
> db.employee.find({"SAL":{$ne:5000}, $or:[{"DNAME":"TESTING"},{"DNAME":"SALES"}]}).pretty()
{
        "_id" : ObjectId("60e6a907bb4f15b0f16c0ec3"),
        "EMPID" : 1,
        "ENAME" : "Aryamaan",
        "SAL" : 15000,
        "CITY" : "THANE",
        "DNAME" : "TESTING"
}
{
        "_id" : ObjectId("60e6a93abb4f15b0f16c0ec4"),
        "EMPID" : 2,
        "ENAME" : "Nachiket",
        "SAL" : 25000,
        "CITY" : "MULUND",
        "DNAME" : "SALES"
}
{
        "_id" : ObjectId("60e6a95abb4f15b0f16c0ec5"),
        "EMPID" : 3,
        "ENAME" : "Omkar",
        "SAL" : 15000,
        "CITY" : "DADAR",
        "DNAME" : "TESTING"
}
```

Practical 2A – Basic Queries

Date: 13-07-2021

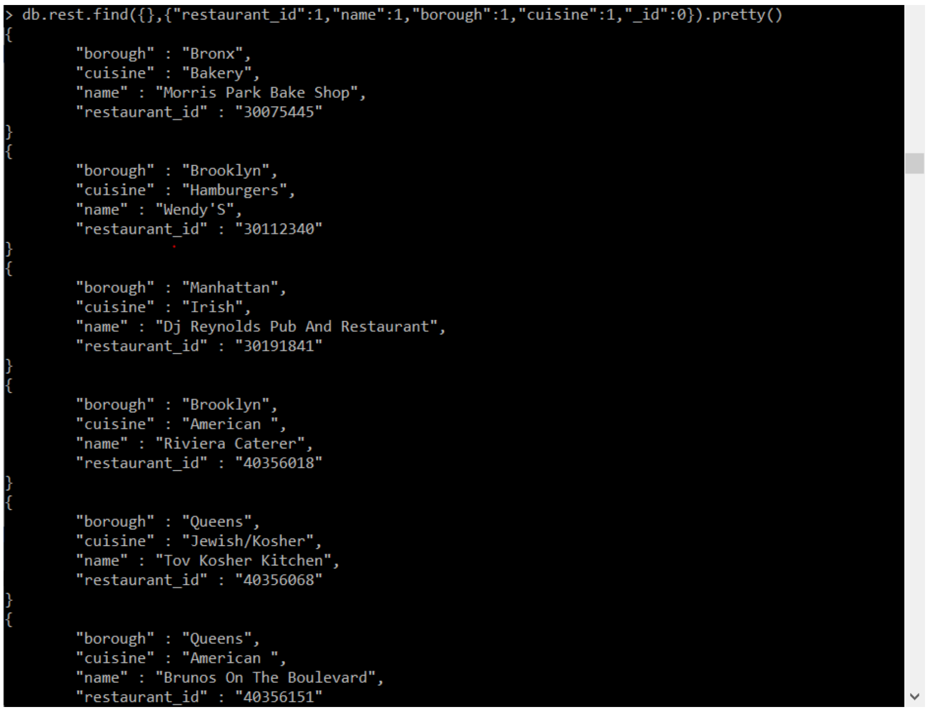
1. Write a MongoDB query to display all the documents in the collection test.



Command Prompt - mongo

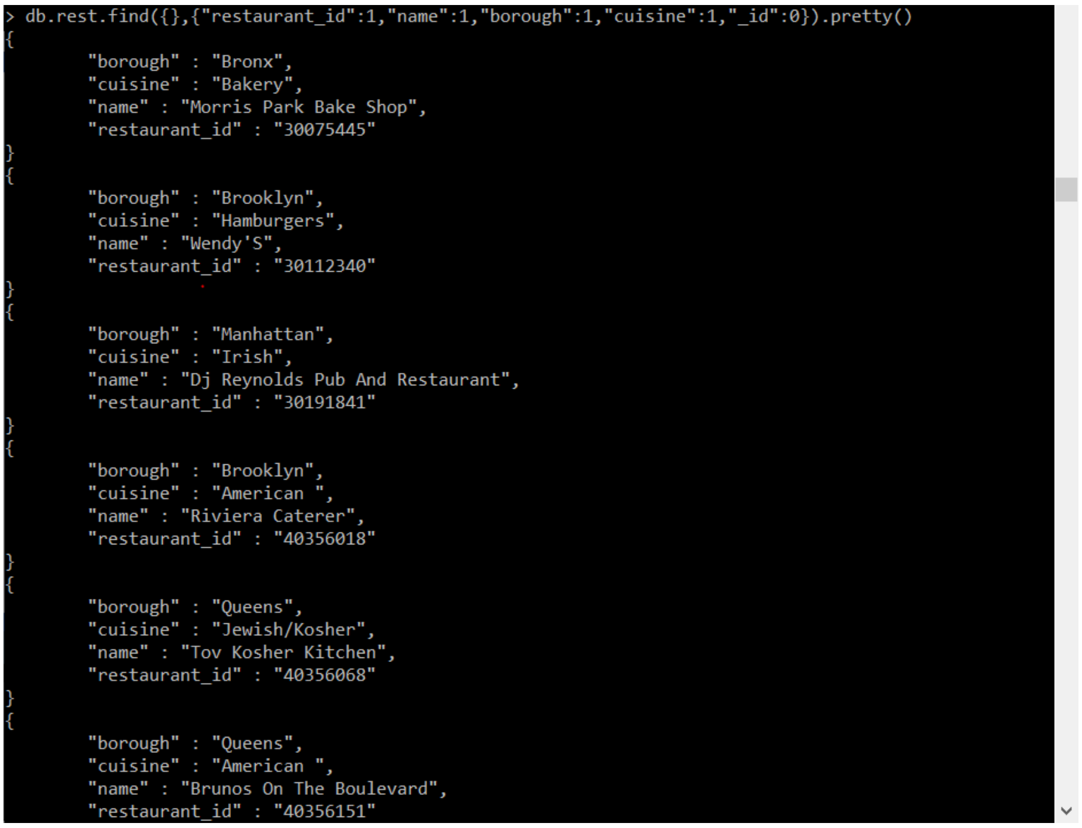
```
> db.rest.find().pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
```

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



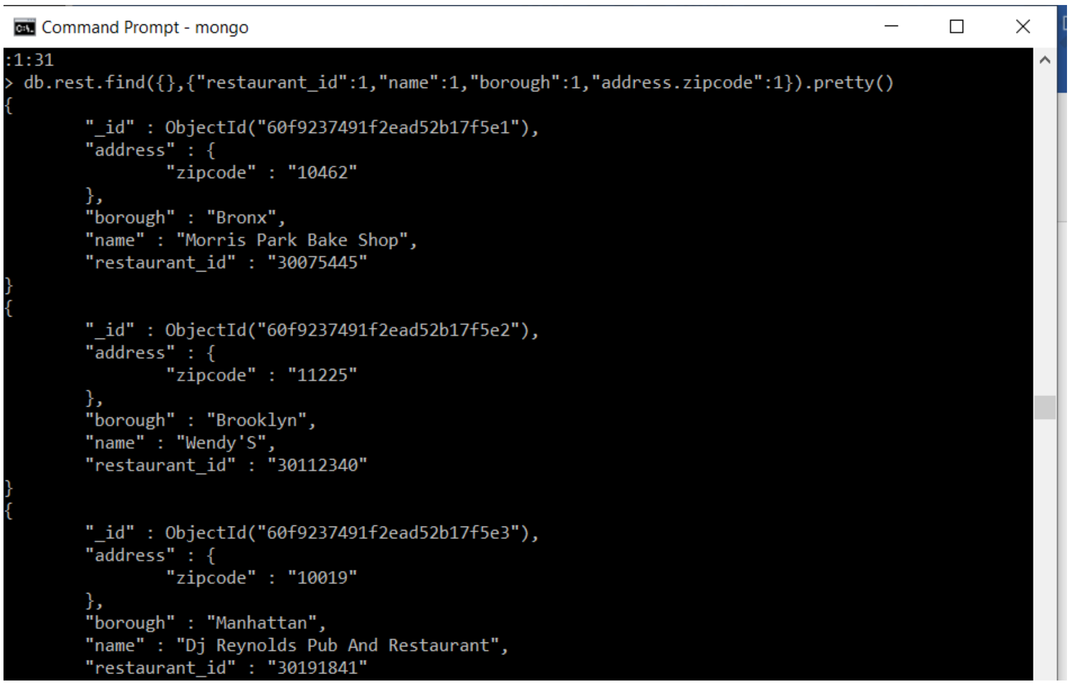
```
> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0}).pretty()
{
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "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"
```

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 rest.



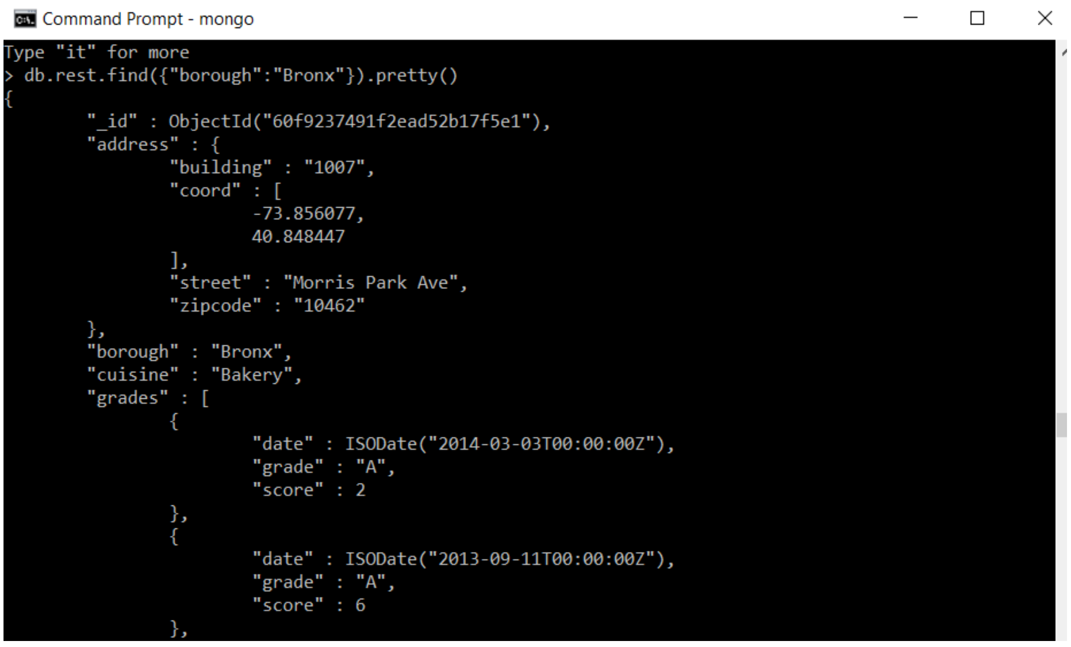
```
> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0}).pretty()
{
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "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"
```

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 collectionrest.



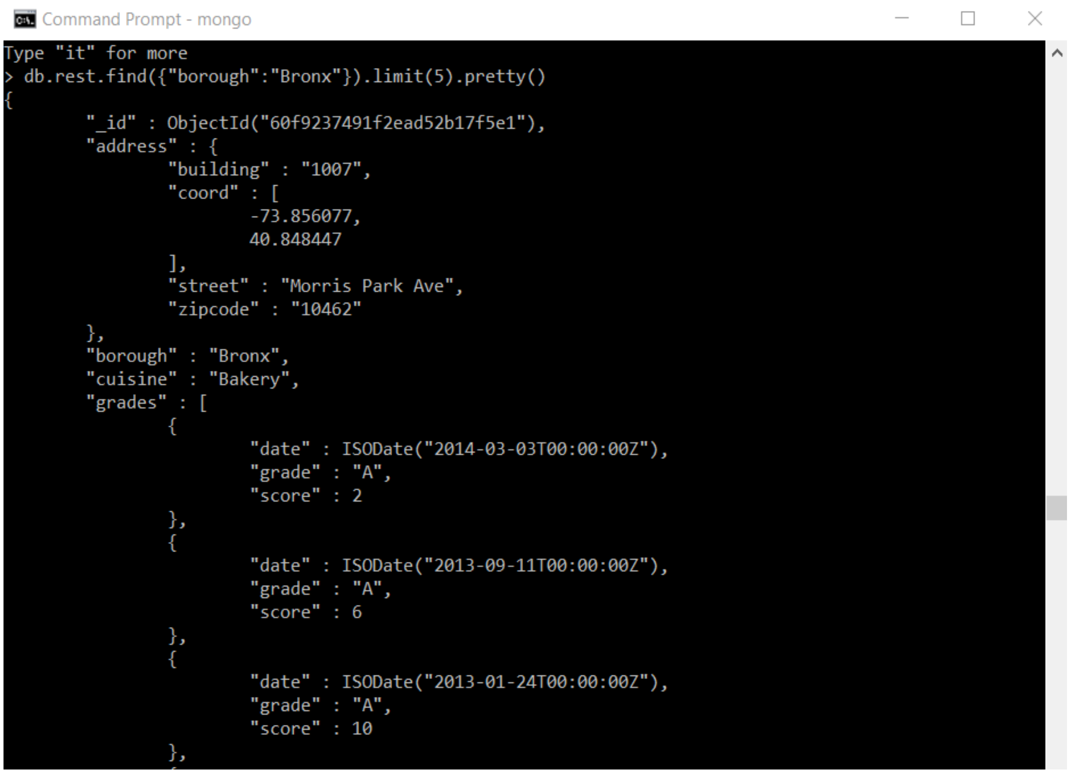
```
:1:31
> db.rest.find({},{"restaurant_id":1,"name":1,"borough":1,"address.zipcode":1}).pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
        "address" : {
                "zipcode" : "11225"
        },
        "borough" : "Brooklyn",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
        "address" : {
                "zipcode" : "10019"
        },
        "borough" : "Manhattan",
        "name" : "Dj Reynolds Pub And Restaurant",
        "restaurant_id" : "30191841"
```

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



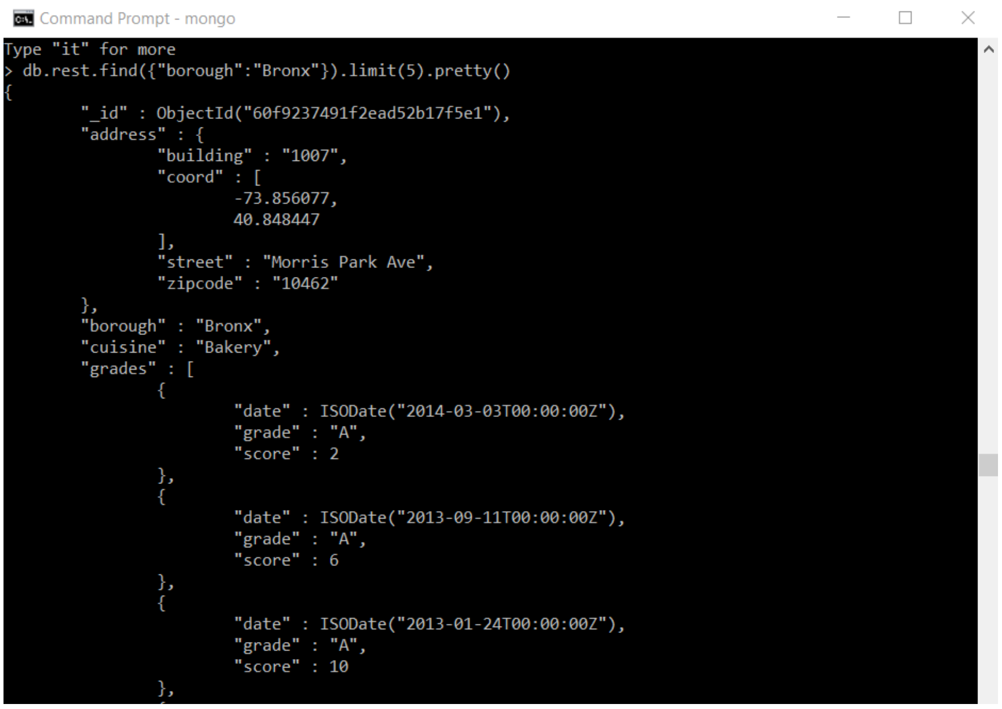
```
Type "it" for more
> db.rest.find({"borough":"Bronx"}).pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
```

6. Write a MongoDB query to display the first 5 records which is in the borough Bronx.



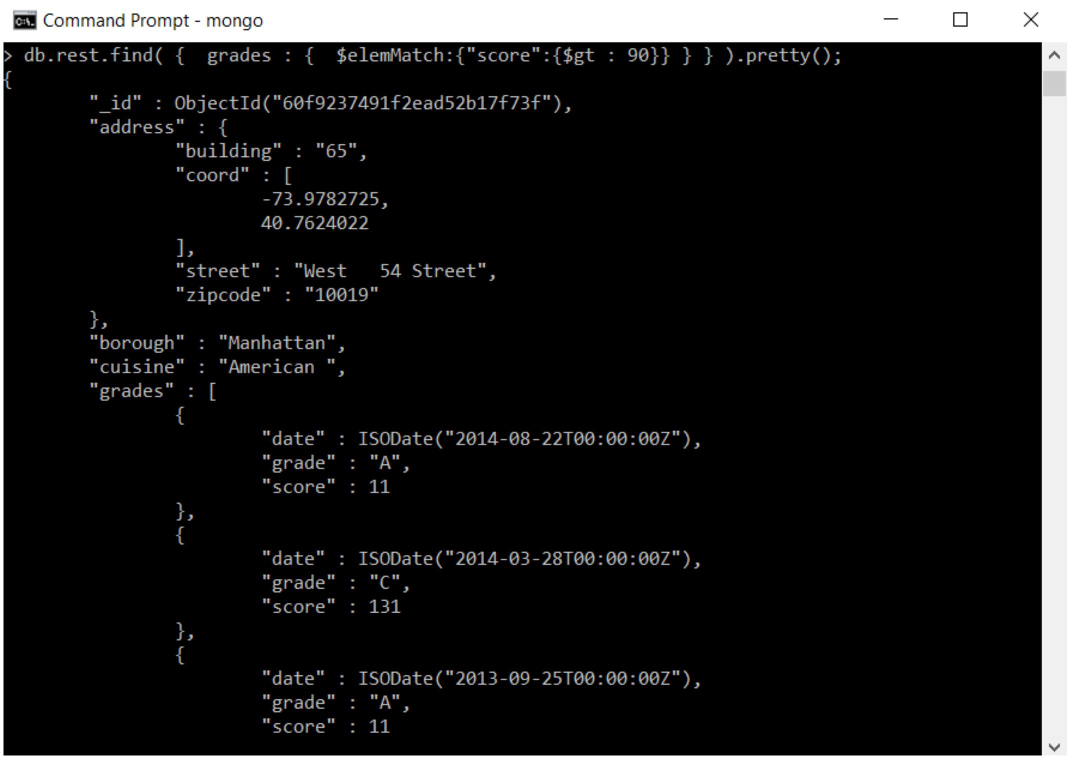
```
Type "it" for more
> db.rest.find({"borough":"Bronx"}).limit(5).pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
```

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



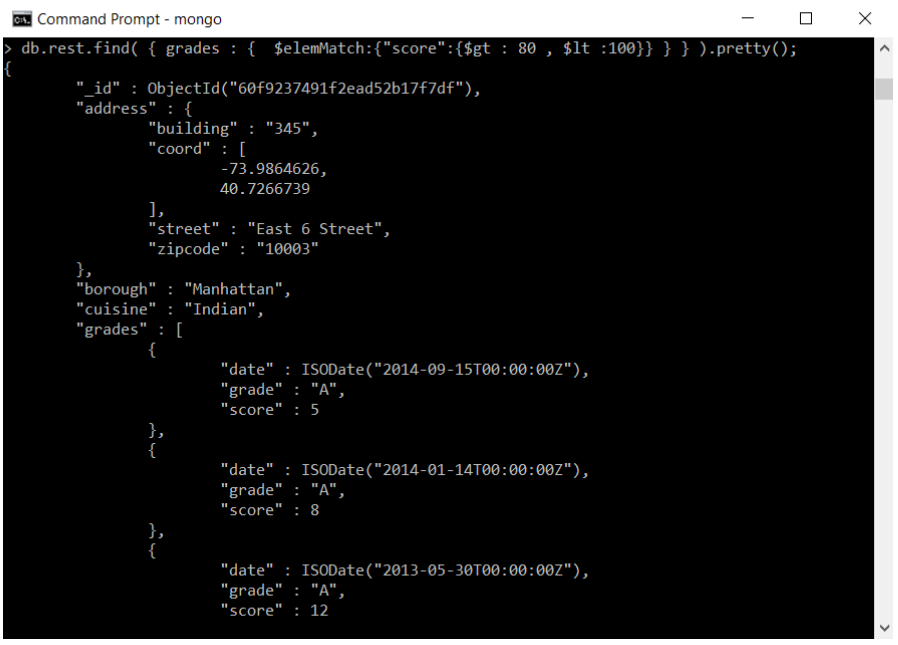
```
Type "it" for more
> db.rest.find({"borough":"Bronx"}).limit(5).pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
```

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



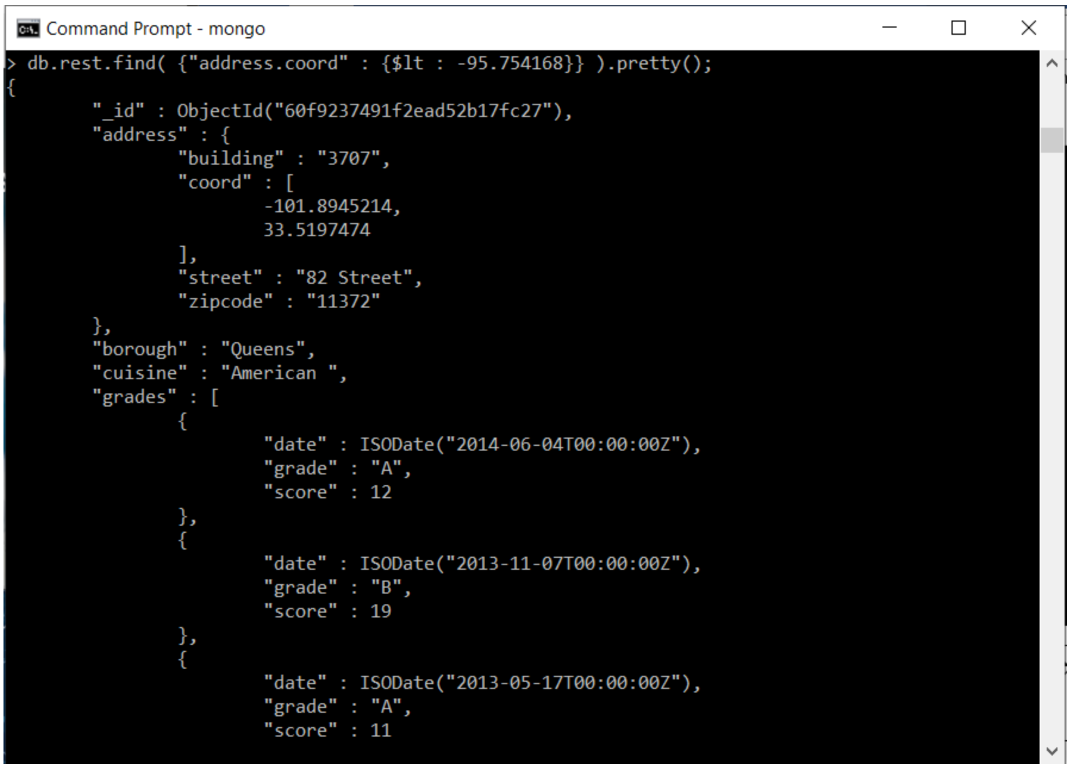
```
> db.rest.find( {  grades : {  $elemMatch:{"score":{$gt : 90}} } } ).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f73f"),
        "address" : {
                "building" : "65",
                "coord" : [
                        -73.9782725,
                        40.7624022
                ],
                "street" : "West   54 Street",
                "zipcode" : "10019"
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "grades" : [
                {
                        "date" : ISODate("2014-08-22T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
                },
                {
                        "date" : ISODate("2014-03-28T00:00:00Z"),
                        "grade" : "C",
                        "score" : 131
                },
                {
                        "date" : ISODate("2013-09-25T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
```

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



```
> db.rest.find( { grades : {  $elemMatch:{"score":{$gt : 80 , $lt :100}} } } ).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f7df"),
        "address" : {
                "building" : "345",
                "coord" : [
                        -73.9864626,
                        40.7266739
                ],
                "street" : "East 6 Street",
                "zipcode" : "10003"
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "grades" : [
                {
                        "date" : ISODate("2014-09-15T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2013-05-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
```

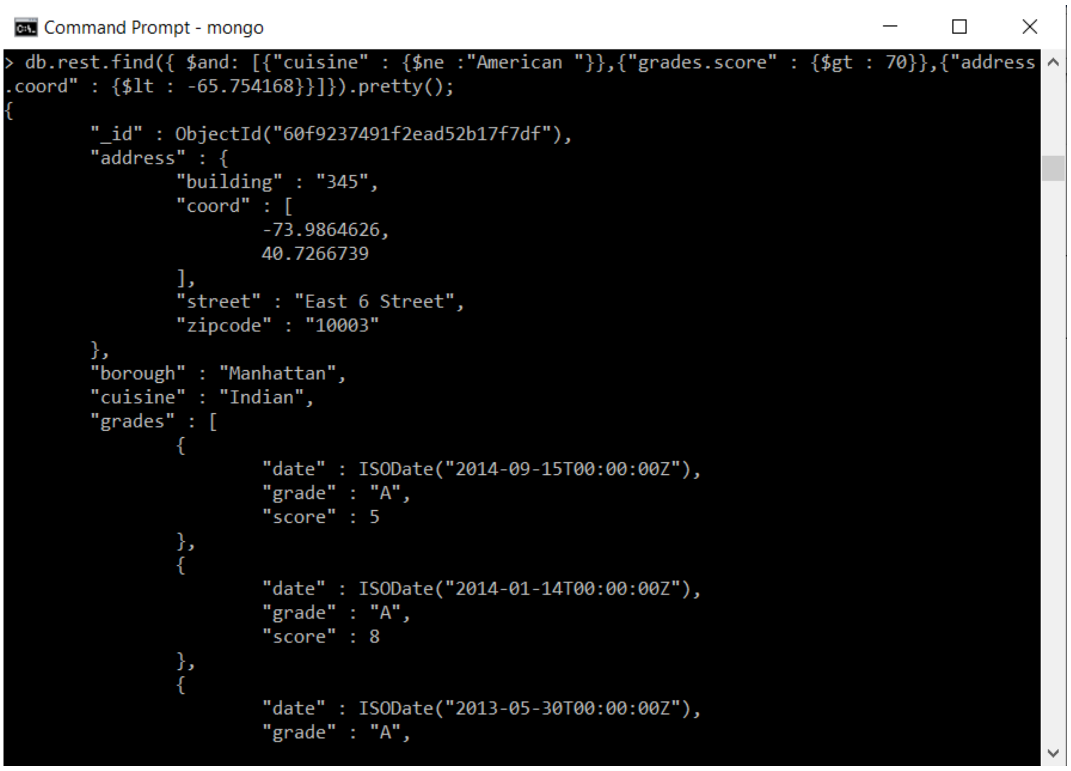
10. Write a MongoDB query to find the rest which locate in latitude value less than -95.754168.



Command Prompt - mongo

```
> db.rest.find( {"address.coord" : {$lt : -95.754168}} ).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17fc27"),
        "address" : {
                "building" : "3707",
                "coord" : [
                        -101.8945214,
                        33.5197474
                ],
                "street" : "82 Street",
                "zipcode" : "11372"
        },
        "borough" : "Queens",
        "cuisine" : "American ",
        "grades" : [
                {
                        "date" : ISODate("2014-06-04T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2013-11-07T00:00:00Z"),
                        "grade" : "B",
                        "score" : 19
                },
                {
                        "date" : ISODate("2013-05-17T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
```

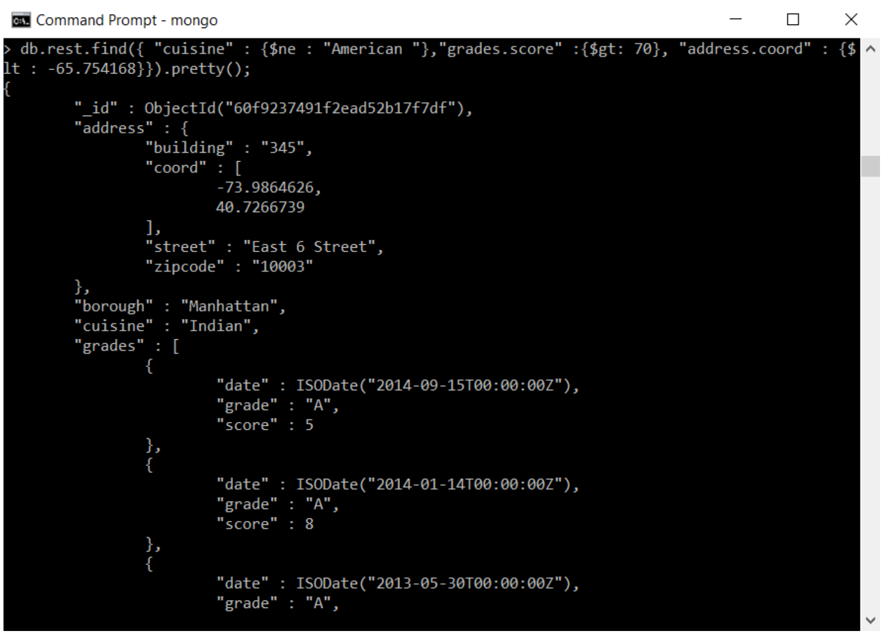
11. Write a MongoDB query to find the rest that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than 65.754168



Command Prompt - mongo

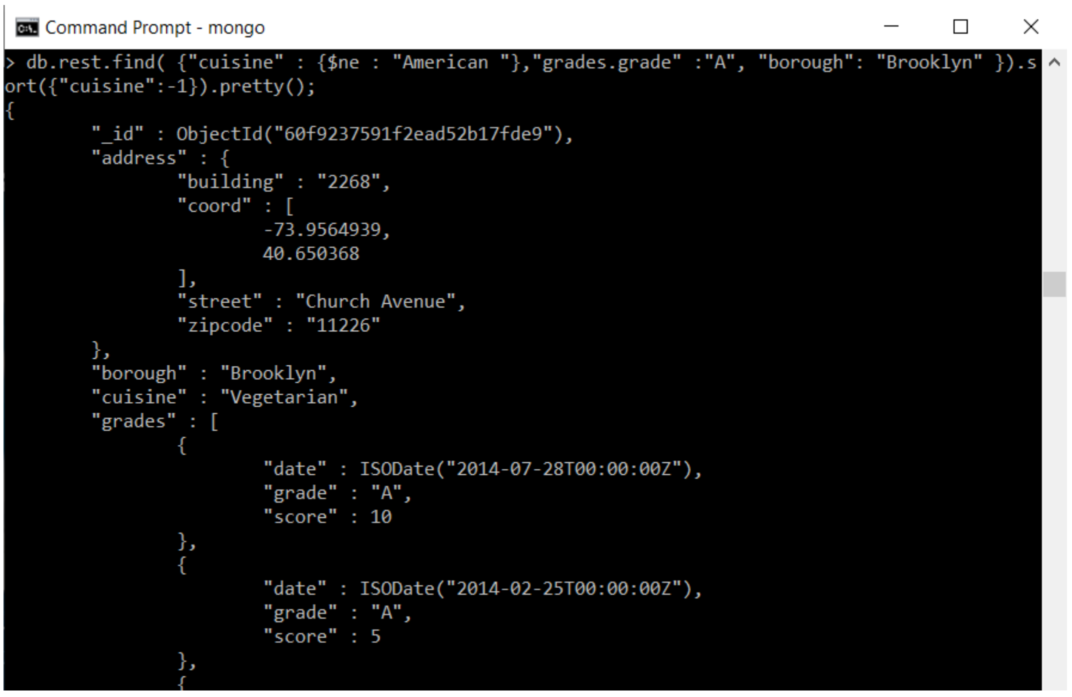
```
> db.rest.find({ $and: [{"cuisine" : {$ne :"American "}},{"grades.score" : {$gt : 70}},{"address
.coord" : {$lt : -65.754168}}]}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f7df"),
        "address" : {
                "building" : "345",
                "coord" : [
                        -73.9864626,
                        40.7266739
                ],
                "street" : "East 6 Street",
                "zipcode" : "10003"
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "grades" : [
                {
                        "date" : ISODate("2014-09-15T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2013-05-30T00:00:00Z"),
                        "grade" : "A",
```

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


Command Prompt - mongo

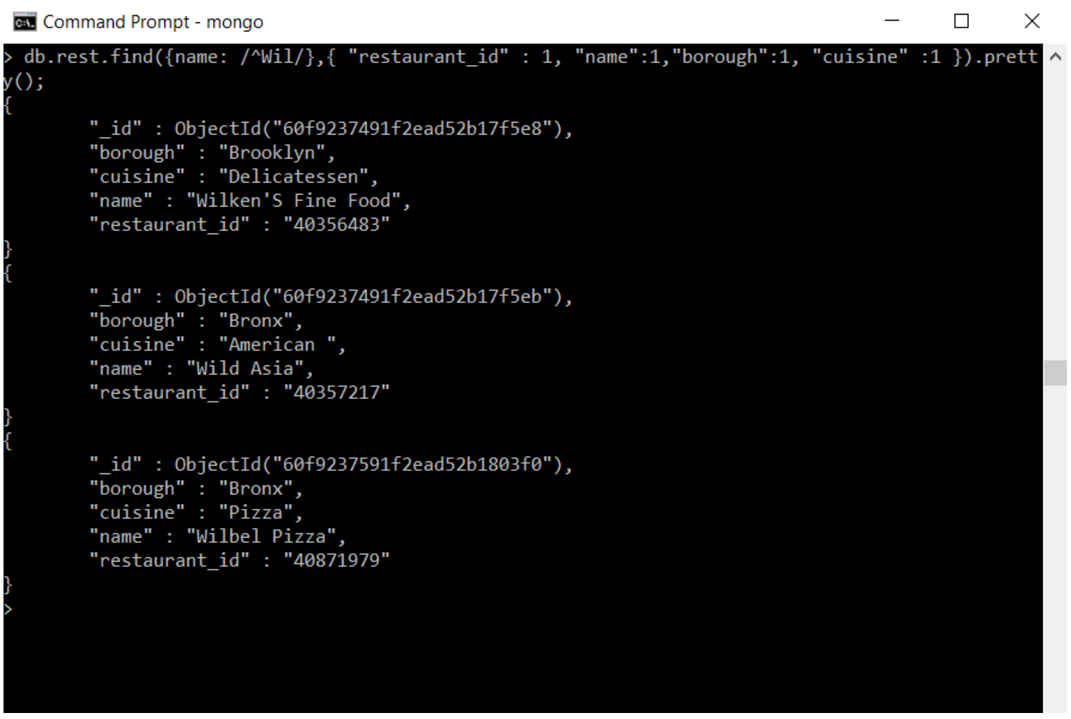
```
> db.rest.find({ "cuisine" : {$ne : "American "},"grades.score" :{$gt: 70}, "address.coord" : {$lt : -65.754168}}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f7df"),
        "address" : {
                "building" : "345",
                "coord" : [
                        -73.9864626,
                        40.7266739
                ],
                "street" : "East 6 Street",
                "zipcode" : "10003"
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "grades" : [
                {
                        "date" : ISODate("2014-09-15T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2013-05-30T00:00:00Z"),
                        "grade" : "A",
```

13. Write a MongoDB query to find the rest 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.


Command Prompt - mongo

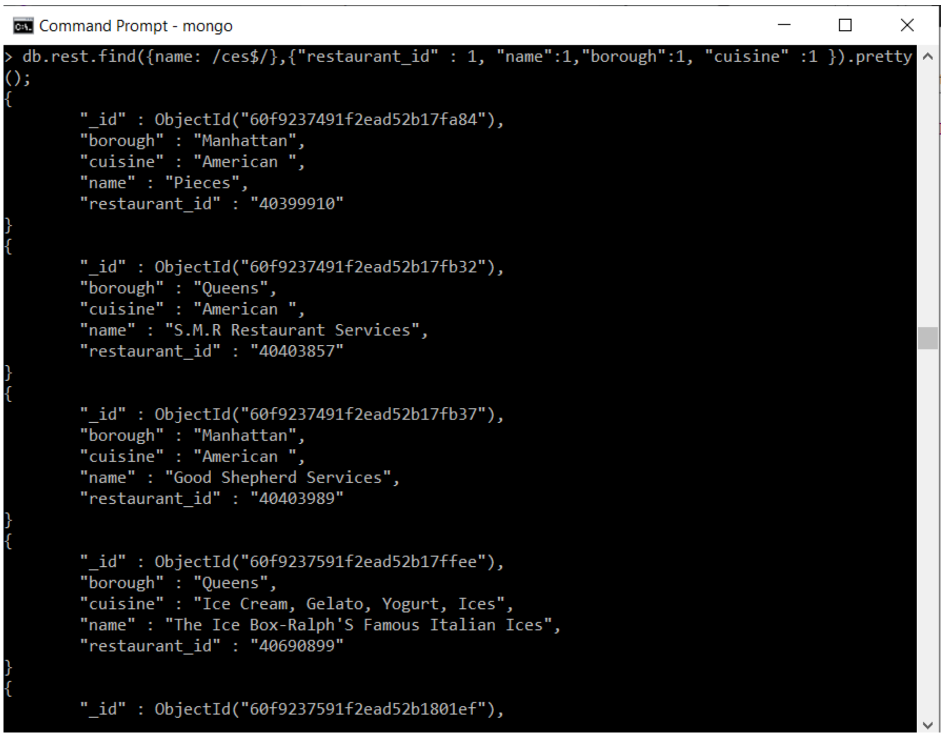
```
> db.rest.find( {"cuisine" : {$ne : "American "},"grades.grade" :"A", "borough": "Brooklyn" }).sort({"cuisine":-1}).pretty();
{
        "_id" : ObjectId("60f9237591f2ead52b17fde9"),
        "address" : {
                "building" : "2268",
                "coord" : [
                        -73.9564939,
                        40.650368
                ],
                "street" : "Church Avenue",
                "zipcode" : "11226"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Vegetarian",
        "grades" : [
                {
                        "date" : ISODate("2014-07-28T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2014-02-25T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
```

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


Command Prompt - mongo

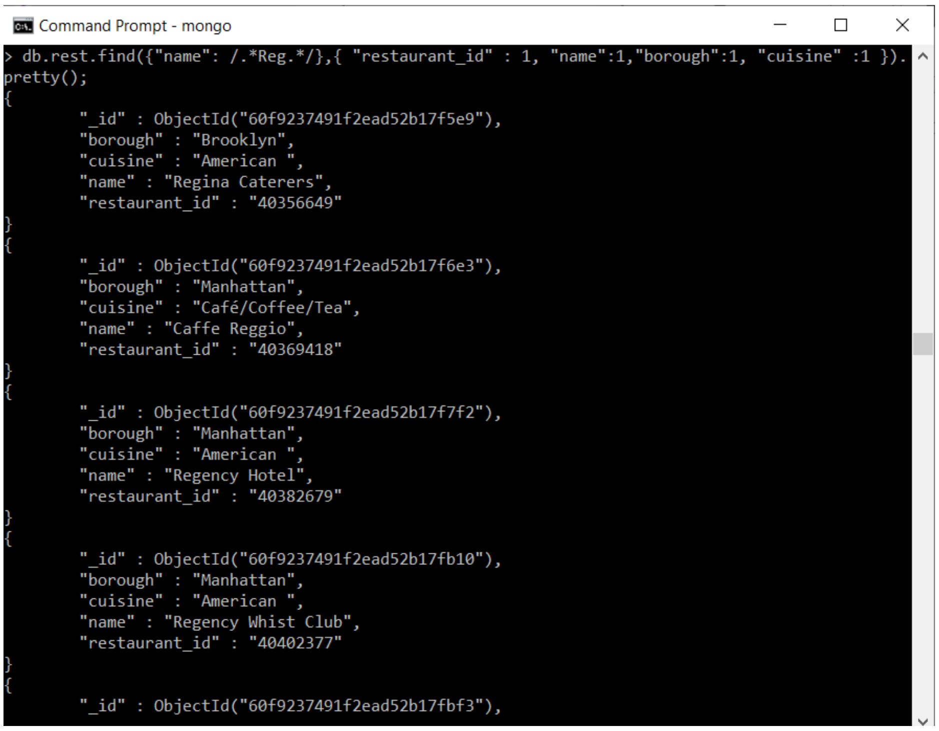
```
> db.rest.find({name: /^Wil/},{ "restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1 }).prett
y();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e8"),
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Wilken'S Fine Food",
        "restaurant_id" : "40356483"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5eb"),
        "borough" : "Bronx",
        "cuisine" : "American ",
        "name" : "Wild Asia",
        "restaurant_id" : "40357217"
}
{
        "_id" : ObjectId("60f9237591f2ead52b1803f0"),
        "borough" : "Bronx",
        "cuisine" : "Pizza",
        "name" : "Wilbel Pizza",
        "restaurant_id" : "40871979"
}
>
```

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


Command Prompt - mongo

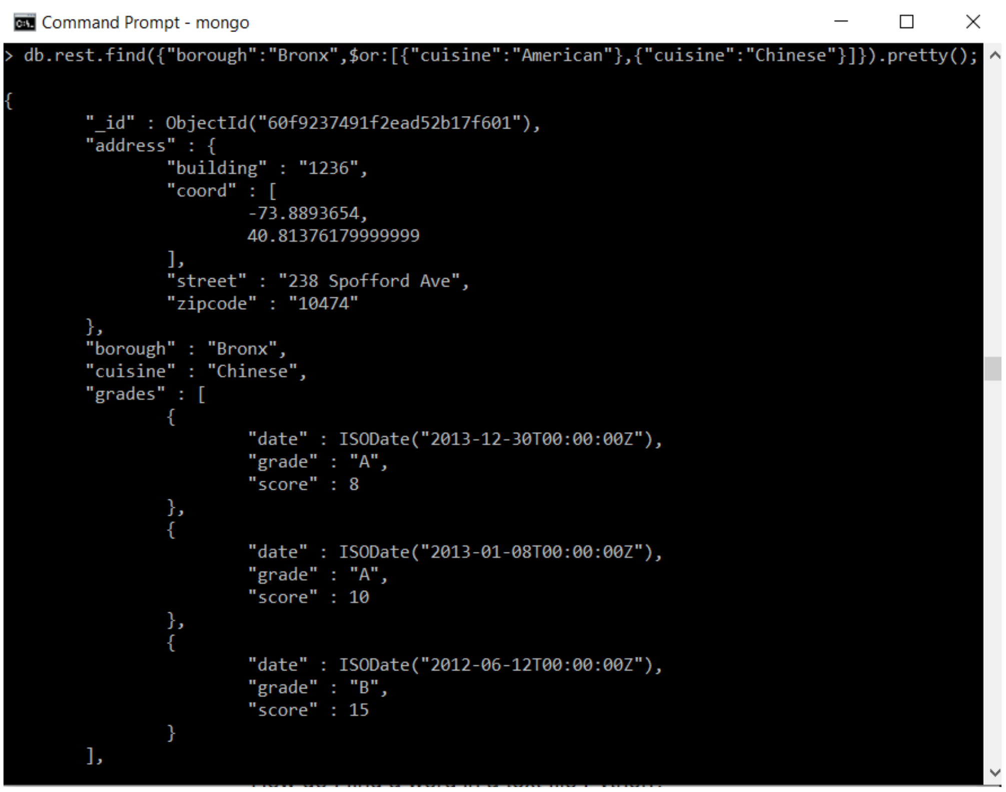
```
> db.rest.find({name: /ces$/},{"restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1 }).pretty
();
{
        "_id" : ObjectId("60f9237491f2ead52b17fa84"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Pieces",
        "restaurant_id" : "40399910"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fb32"),
        "borough" : "Queens",
        "cuisine" : "American ",
        "name" : "S.M.R Restaurant Services",
        "restaurant_id" : "40403857"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fb37"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Good Shepherd Services",
        "restaurant_id" : "40403989"
}
{
        "_id" : ObjectId("60f9237591f2ead52b17ffee"),
        "borough" : "Queens",
        "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
        "name" : "The Ice Box-Ralph'S Famous Italian Ices",
        "restaurant_id" : "40690899"
}
{
        "_id" : ObjectId("60f9237591f2ead52b1801ef"),
```

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



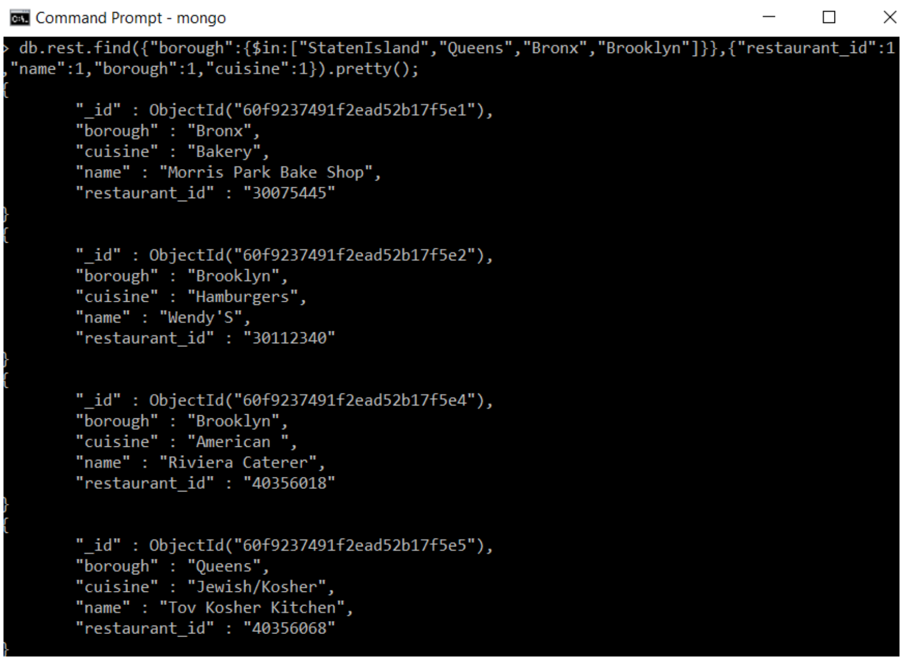
```
> db.rest.find({"name": /.*Reg.*/},{ "restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1 }).
pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e9"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Regina Caterers",
        "restaurant_id" : "40356649"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f6e3"),
        "borough" : "Manhattan",
        "cuisine" : "Café/Coffee/Tea",
        "name" : "Caffe Reggio",
        "restaurant_id" : "40369418"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f7f2"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Regency Hotel",
        "restaurant_id" : "40382679"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fb10"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Regency Whist Club",
        "restaurant_id" : "40402377"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fbf3"),
```

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



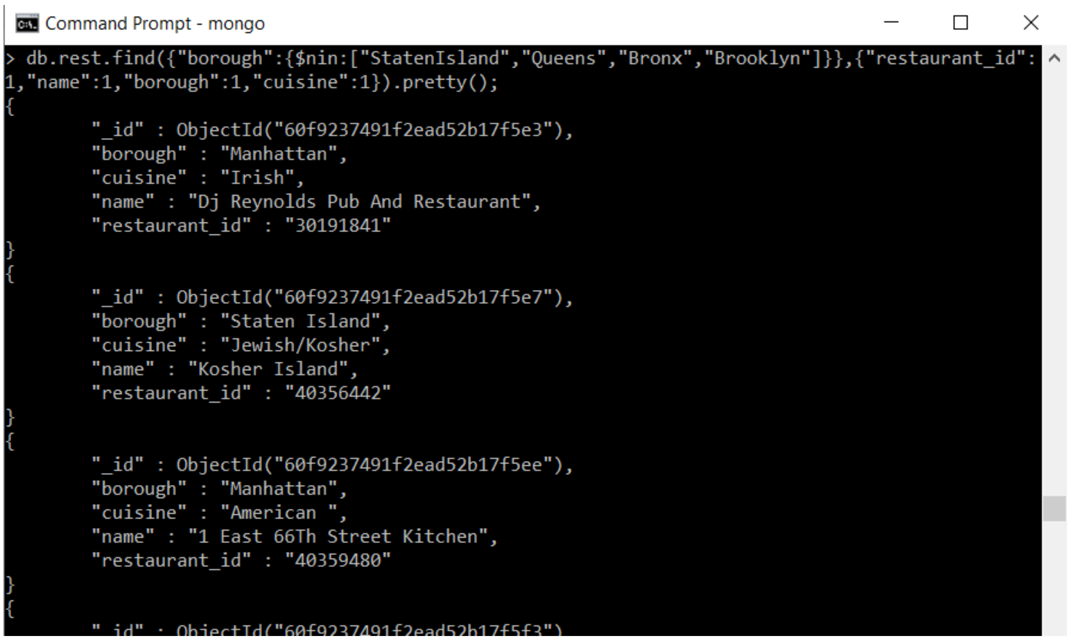
```
> db.rest.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"cuisine":"Chinese"}]}).pretty();

{
        "_id" : ObjectId("60f9237491f2ead52b17f601"),
        "address" : {
                "building" : "1236",
                "coord" : [
                        -73.8893654,
                        40.81376179999999
                ],
                "street" : "238 Spofford Ave",
                "zipcode" : "10474"
        },
        "borough" : "Bronx",
        "cuisine" : "Chinese",
        "grades" : [
                {
                        "date" : ISODate("2013-12-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2013-01-08T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2012-06-12T00:00:00Z"),
                        "grade" : "B",
                        "score" : 15
                }
        ],
```

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



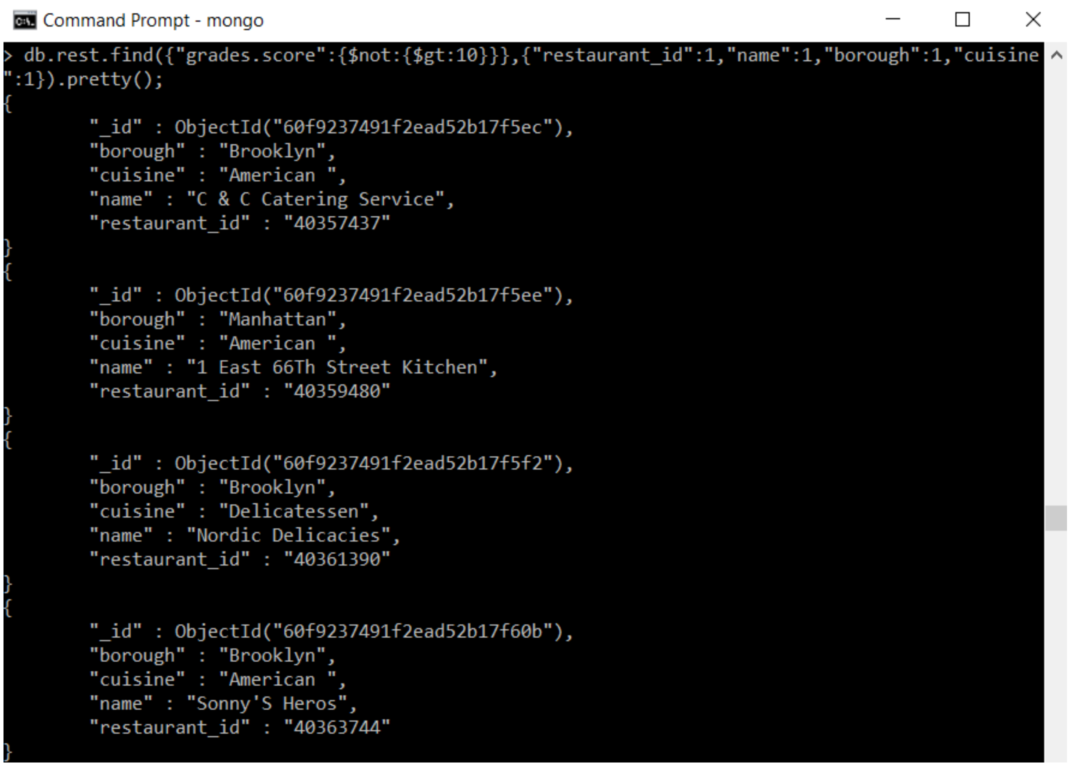
```
> db.rest.find({"borough":{$in:["StatenIsland","Queens","Bronx","Brooklyn"]}},{"restaurant_id":1
,"name":1,"borough":1,"cuisine":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e4"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e5"),
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
```

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



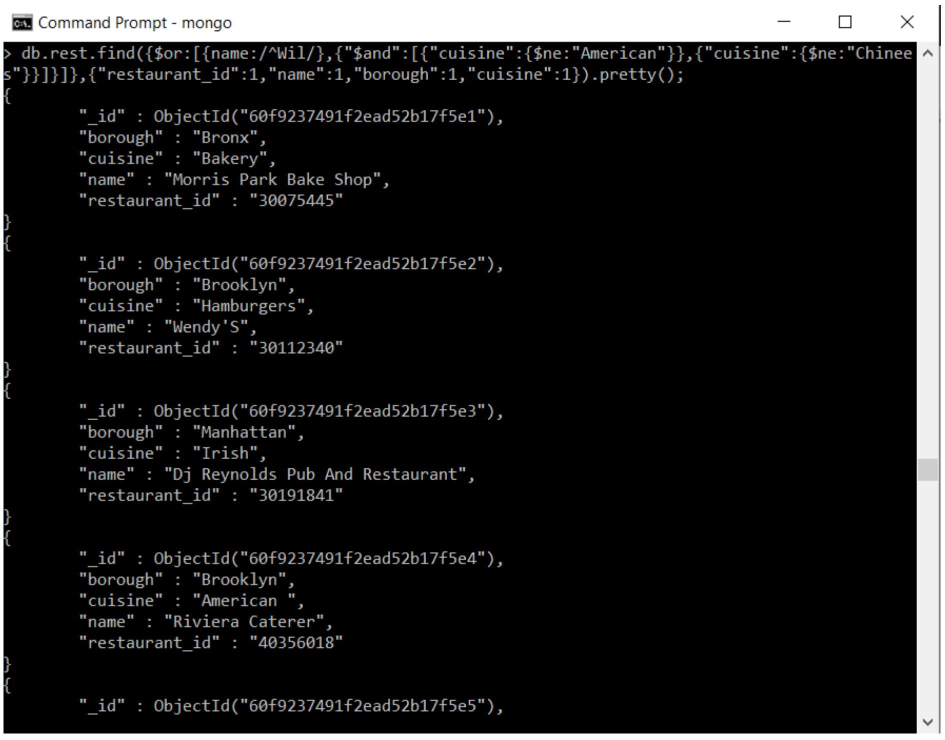
```
> db.rest.find({"borough":{$nin:["StatenIsland","Queens","Bronx","Brooklyn"]}},{"restaurant_id":
1,"name":1,"borough":1,"cuisine":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
        "borough" : "Manhattan",
        "cuisine" : "Irish",
        "name" : "Dj Reynolds Pub And Restaurant",
        "restaurant_id" : "30191841"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e7"),
        "borough" : "Staten Island",
        "cuisine" : "Jewish/Kosher",
        "name" : "Kosher Island",
        "restaurant_id" : "40356442"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5ee"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "1 East 66Th Street Kitchen",
        "restaurant_id" : "40359480"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5f3")
```

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



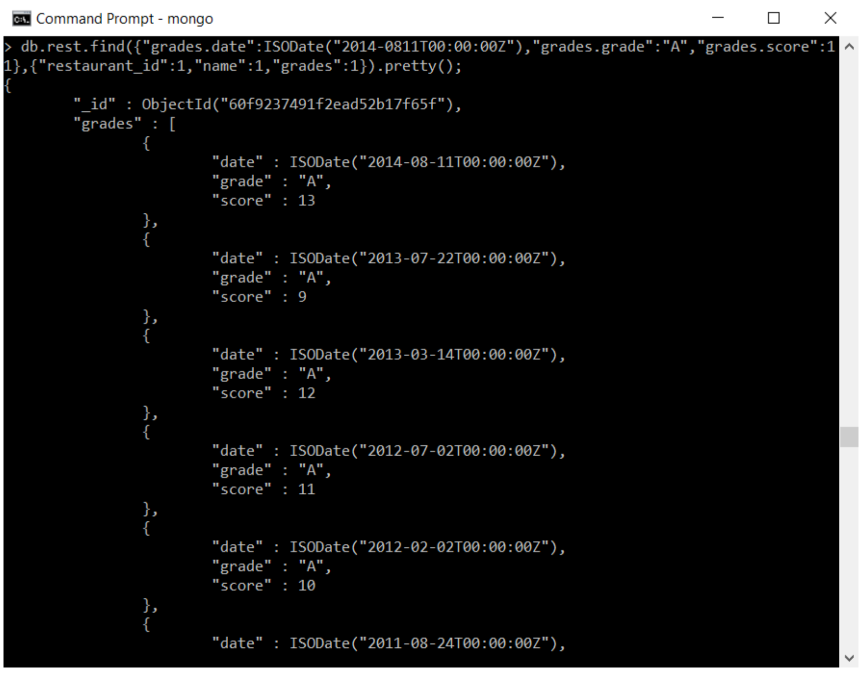
```
> db.rest.find({"grades.score":{$not:{$gt:10}}},{"restaurant_id":1,"name":1,"borough":1,"cuisine
":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5ec"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "C & C Catering Service",
        "restaurant_id" : "40357437"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5ee"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "1 East 66Th Street Kitchen",
        "restaurant_id" : "40359480"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5f2"),
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Nordic Delicacies",
        "restaurant_id" : "40361390"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f60b"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Sonny'S Heros",
        "restaurant_id" : "40363744"
}
```

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



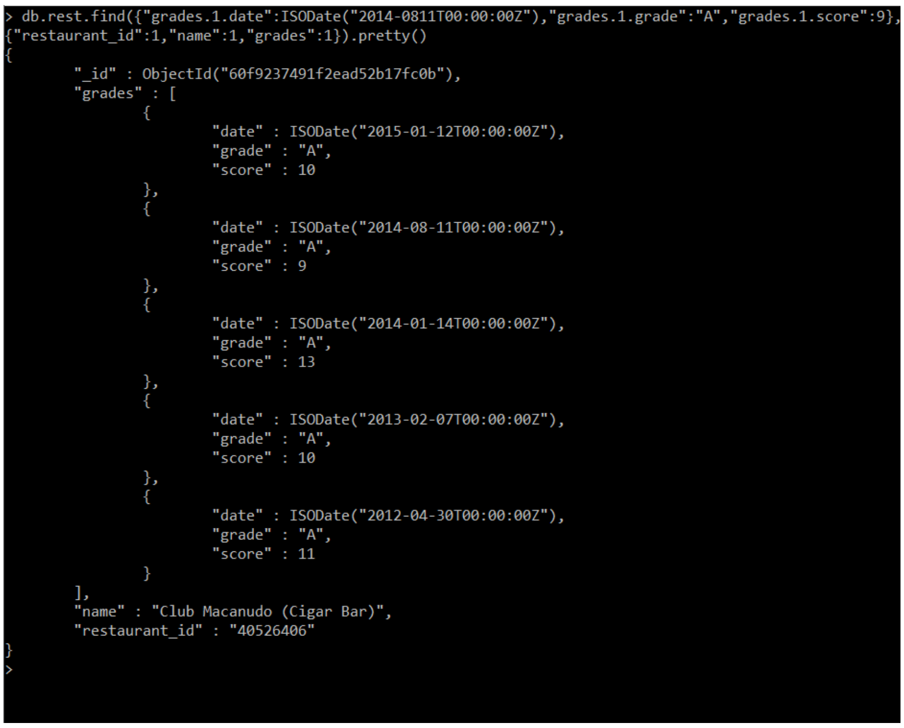
```
> db.rest.find({$or:[{name:/^Wil/},{"$and":[{"cuisine":{$ne:"American"}},{"cuisine":{$ne:"Chinee
s"}}]}]},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e2"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e3"),
        "borough" : "Manhattan",
        "cuisine" : "Irish",
        "name" : "Dj Reynolds Pub And Restaurant",
        "restaurant_id" : "30191841"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e4"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e5"),
```

22. Write a MongoDB query to find the restaurant Id, name, and grades for those rest which achieved a grade of "A" and scored 11 on an ISODate "201408-11T00:00:00Z" among many of survey dates.


Command Prompt - mongo

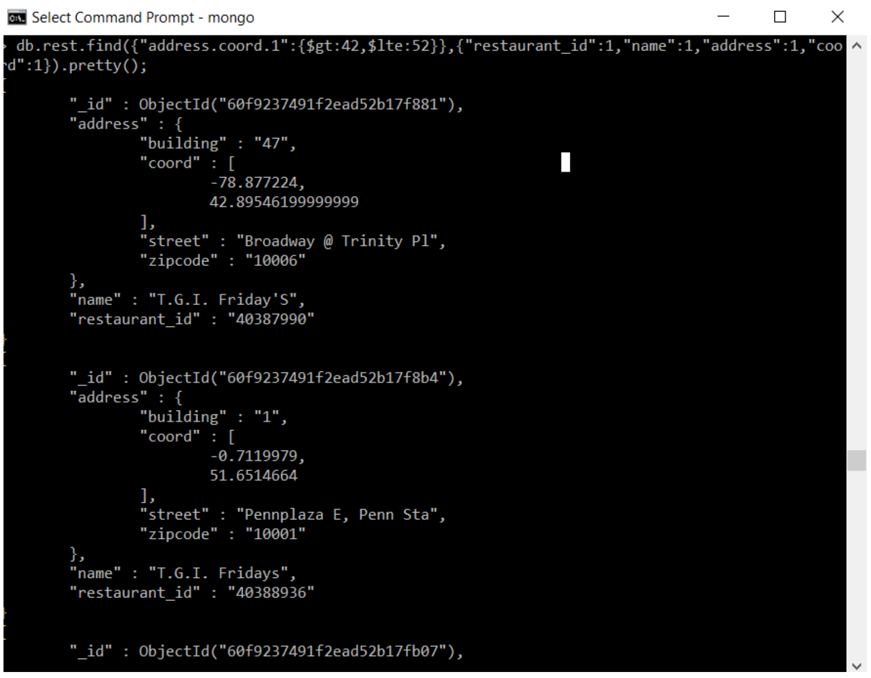
```
> db.rest.find({"grades.date":ISODate("2014-0811T00:00:00Z"),"grades.grade":"A","grades.score":1
1},{"restaurant_id":1,"name":1,"grades":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f65f"),
        "grades" : [
                {
                        "date" : ISODate("2014-08-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
                        "date" : ISODate("2013-07-22T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2013-03-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-07-02T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
                },
                {
                        "date" : ISODate("2012-02-02T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-08-24T00:00:00Z"),
```

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".



```
> db.rest.find({"grades.1.date":ISODate("2014-0811T00:00:00Z"),"grades.1.grade":"A","grades.1.score":9},
{"restaurant_id":1,"name":1,"grades":1}).pretty()
{
        "_id" : ObjectId("60f9237491f2ead52b17fc0b"),
        "grades" : [
                {
                        "date" : ISODate("2015-01-12T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2014-08-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
                        "date" : ISODate("2013-02-07T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2012-04-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
                }
        ],
        "name" : "Club Macanudo (Cigar Bar)",
        "restaurant_id" : "40526406"
}
>
```

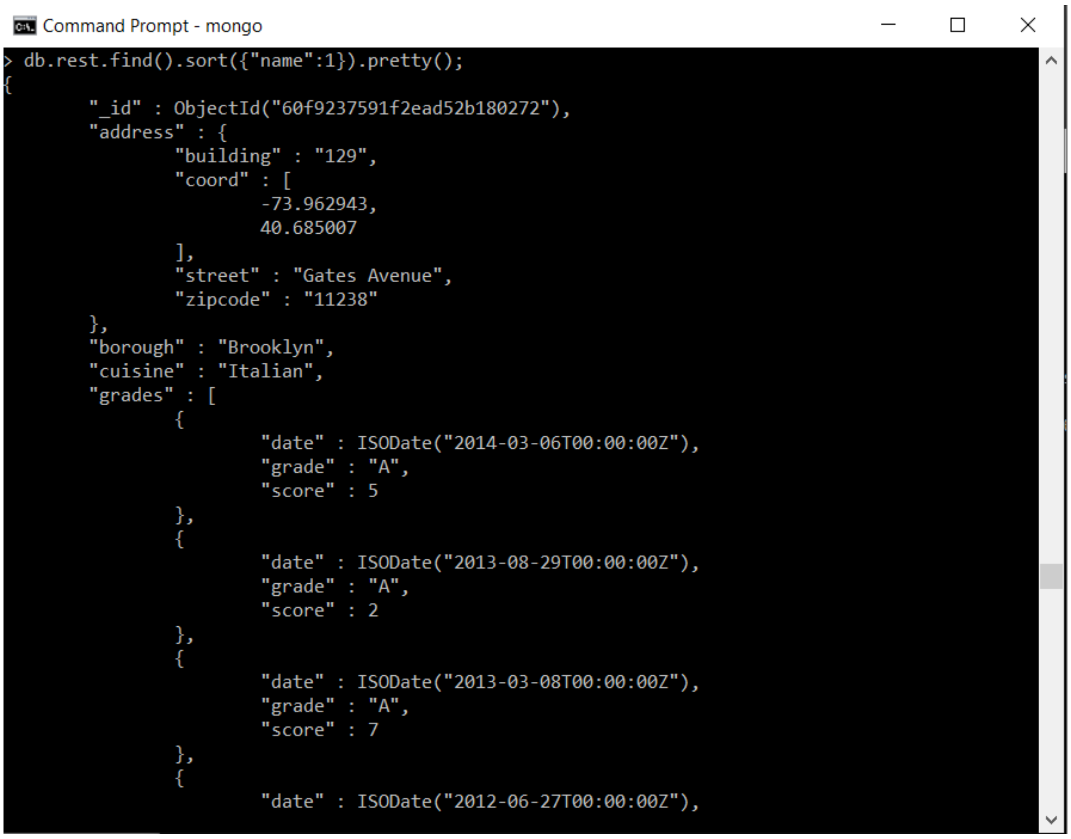
24. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those rest where 2nd element of coord array contains a value which is more than 42 and upto 52.



Select Command Prompt - mongo

```
> db.rest.find({"address.coord.1":{$gt:42,$lte:52}},{"restaurant_id":1,"name":1,"address":1,"coo
rd":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f881"),
        "address" : {
                "building" : "47",
                "coord" : [
                        -78.877224,
                        42.89546199999999
                ],
                "street" : "Broadway @ Trinity Pl",
                "zipcode" : "10006"
        },
        "name" : "T.G.I. Friday'S",
        "restaurant_id" : "40387990"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f8b4"),
        "address" : {
                "building" : "1",
                "coord" : [
                        -0.7119979,
                        51.6514664
                ],
                "street" : "Pennplaza E, Penn Sta",
                "zipcode" : "10001"
        },
        "name" : "T.G.I. Fridays",
        "restaurant_id" : "40388936"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fb07"),
```

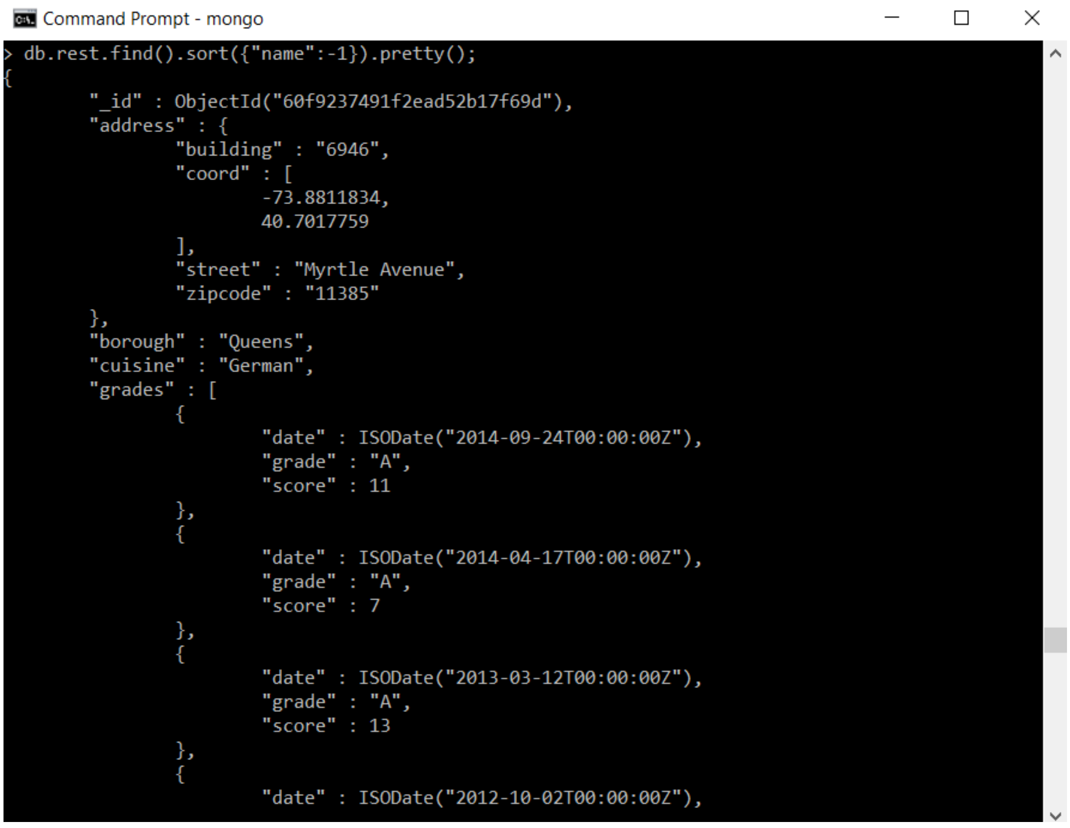
25. Write a MongoDB query to arrange the name of the rest in ascending order along with all the columns.



Command Prompt - mongo

```
> db.rest.find().sort({"name":1}).pretty();
{
        "_id" : ObjectId("60f9237591f2ead52b180272"),
        "address" : {
                "building" : "129",
                "coord" : [
                        -73.962943,
                        40.685007
                ],
                "street" : "Gates Avenue",
                "zipcode" : "11238"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Italian",
        "grades" : [
                {
                        "date" : ISODate("2014-03-06T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
                        "date" : ISODate("2013-08-29T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-03-08T00:00:00Z"),
                        "grade" : "A",
                        "score" : 7
                },
                {
                        "date" : ISODate("2012-06-27T00:00:00Z"),
```

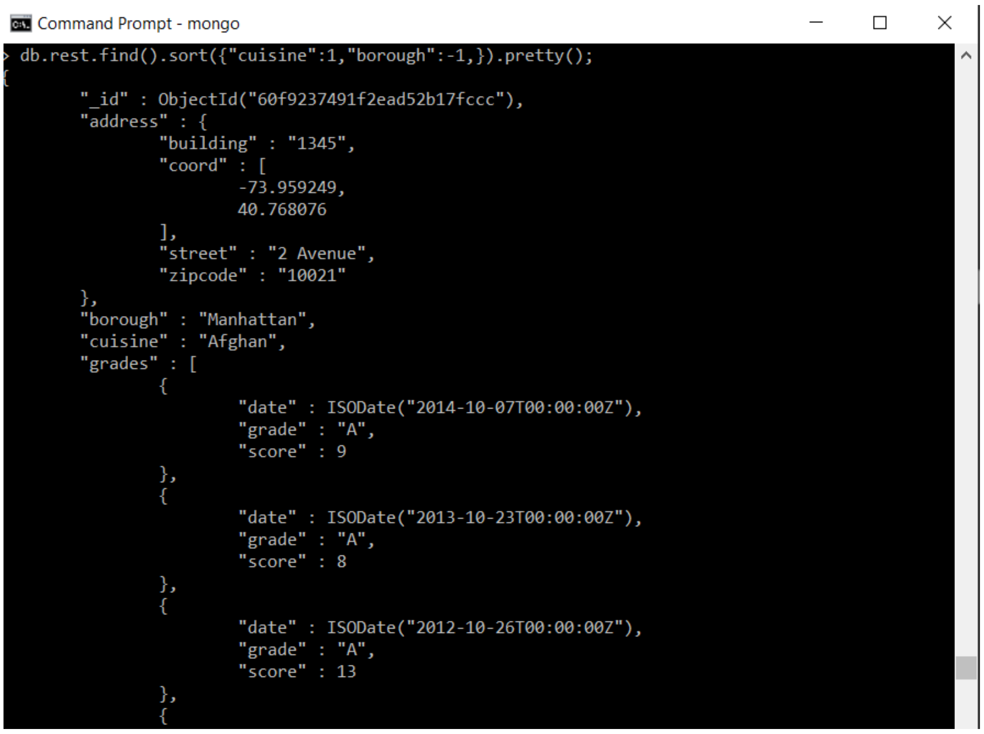
26. Write a MongoDB query to arrange the name of the rest in descending along with all the columns.



Command Prompt - mongo

```
> db.rest.find().sort({"name":-1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f69d"),
        "address" : {
                "building" : "6946",
                "coord" : [
                        -73.8811834,
                        40.7017759
                ],
                "street" : "Myrtle Avenue",
                "zipcode" : "11385"
        },
        "borough" : "Queens",
        "cuisine" : "German",
        "grades" : [
                {
                        "date" : ISODate("2014-09-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
                },
                {
                        "date" : ISODate("2014-04-17T00:00:00Z"),
                        "grade" : "A",
                        "score" : 7
                },
                {
                        "date" : ISODate("2013-03-12T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
                        "date" : ISODate("2012-10-02T00:00:00Z"),
```

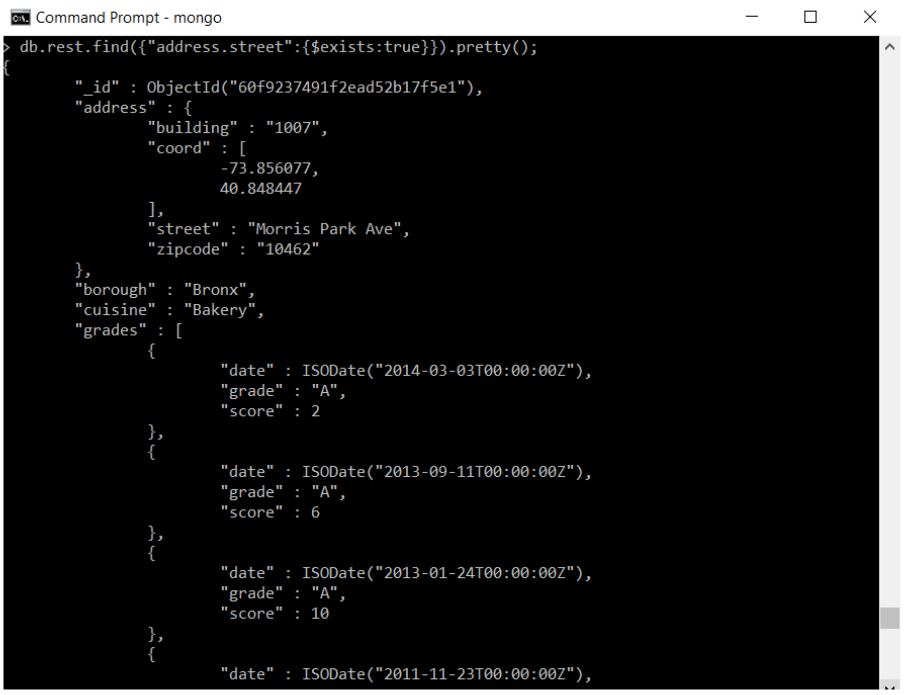
27. Write a MongoDB query to arranged the name of the cuisine in ascending order and borough should be in descending order .



Command Prompt - mongo

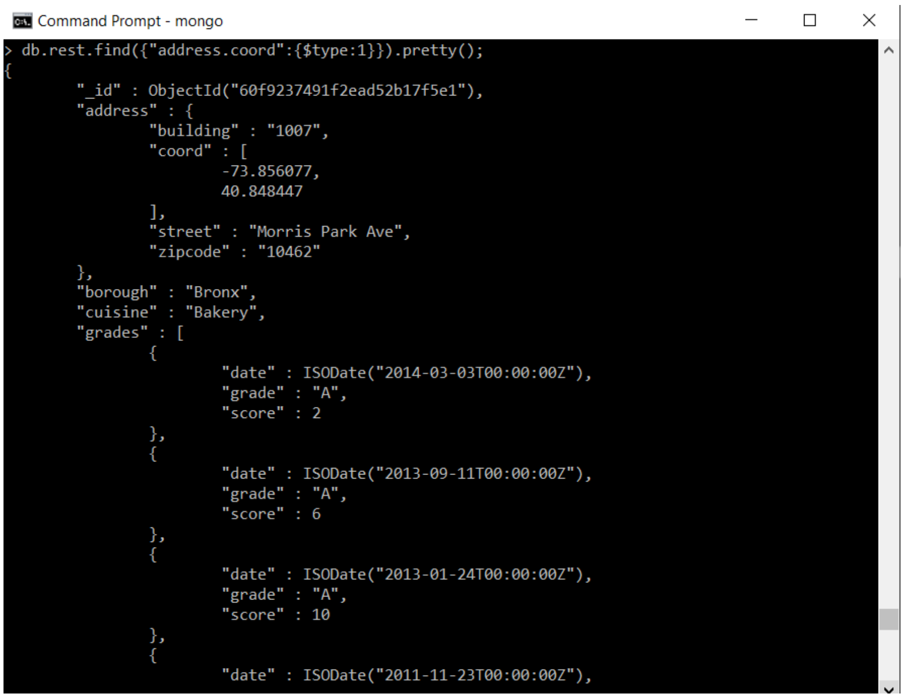
```
> db.rest.find().sort({"cuisine":1,"borough":-1,}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17fccc"),
        "address" : {
                "building" : "1345",
                "coord" : [
                        -73.959249,
                        40.768076
                ],
                "street" : "2 Avenue",
                "zipcode" : "10021"
        },
        "borough" : "Manhattan",
        "cuisine" : "Afghan",
        "grades" : [
                {
                        "date" : ISODate("2014-10-07T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2013-10-23T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2012-10-26T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
```

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



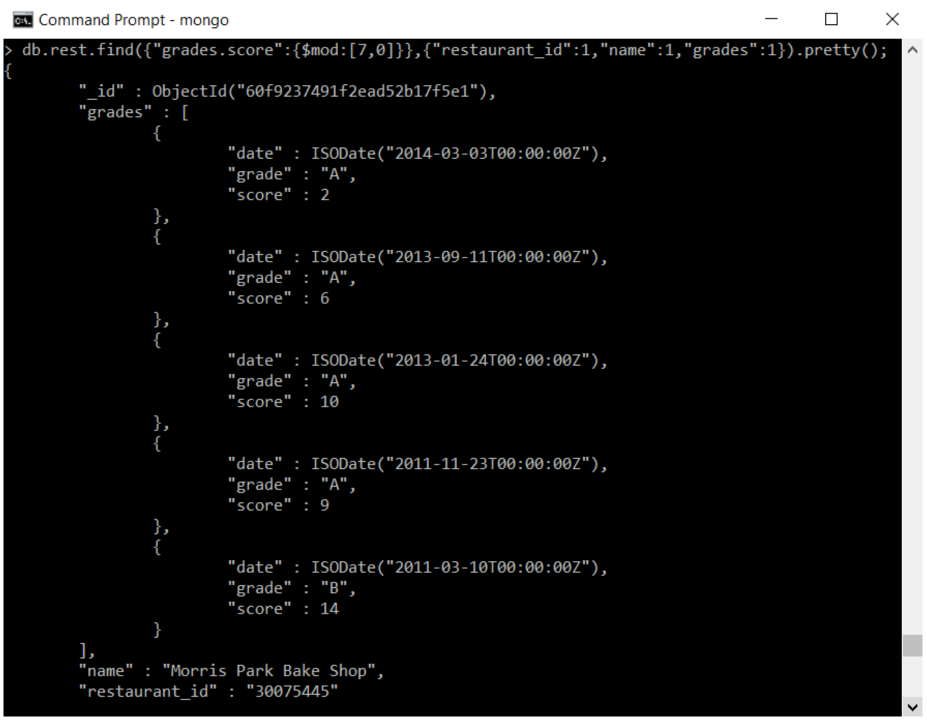
```
> db.rest.find({"address.street":{$exists:true}}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
```

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



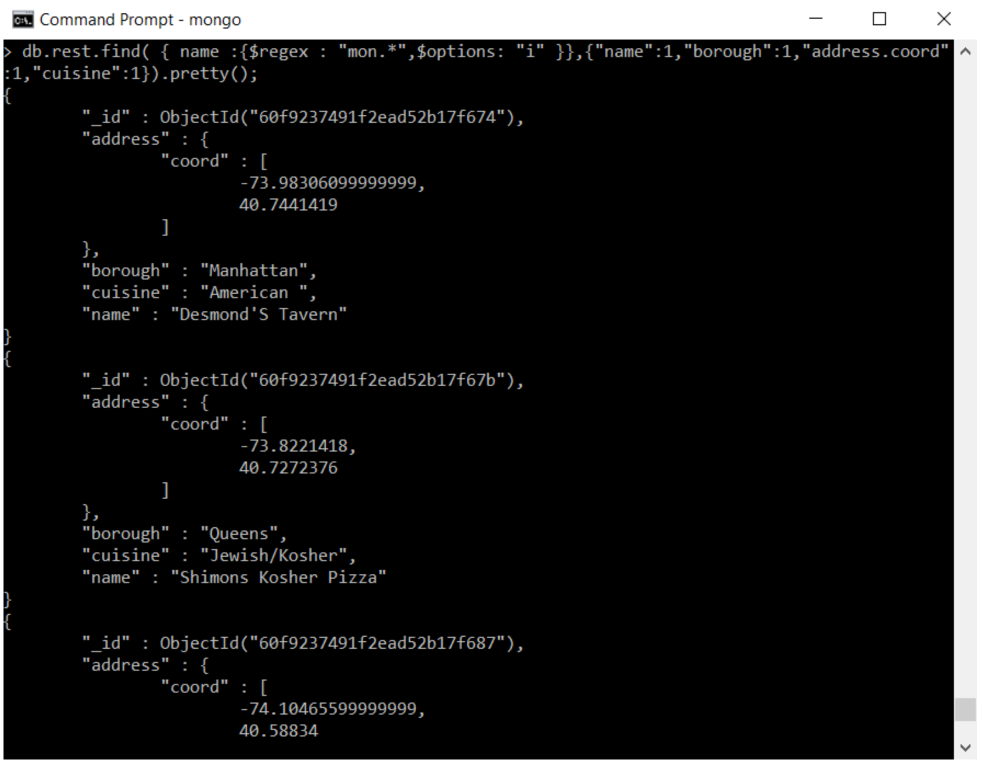
```
> db.rest.find({"address.coord":{$type:1}}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
```

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


Command Prompt - mongo

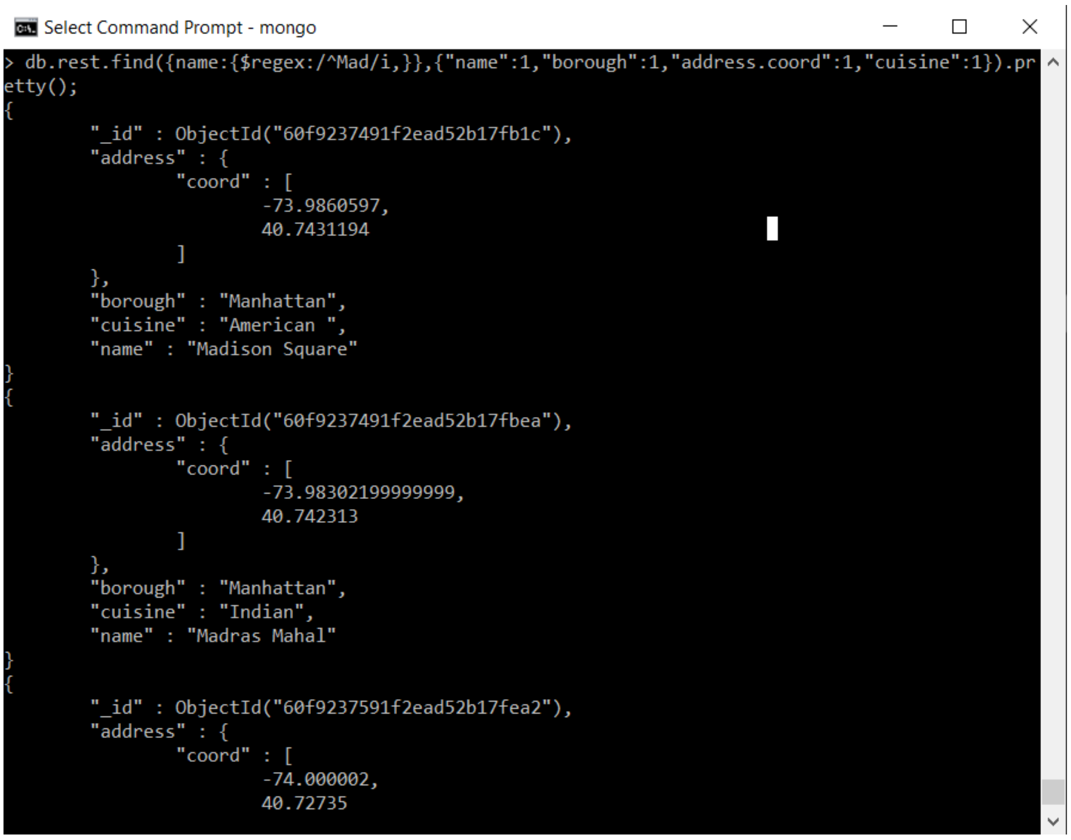
```
> db.rest.find({"grades.score":{$mod:[7,0]}},{"restaurant_id":1,"name":1,"grades":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f5e1"),
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2011-03-10T00:00:00Z"),
                        "grade" : "B",
                        "score" : 14
                }
        ],
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
```

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


Command Prompt - mongo

```
> db.rest.find( { name :{$regex : "mon.*",$options: "i" }},{"name":1,"borough":1,"address.coord"
:1,"cuisine":1}).pretty();
{
        "_id" : ObjectId("60f9237491f2ead52b17f674"),
        "address" : {
                "coord" : [
                        -73.98306099999999,
                        40.7441419
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Desmond'S Tavern"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f67b"),
        "address" : {
                "coord" : [
                        -73.8221418,
                        40.7272376
                ]
        },
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Shimons Kosher Pizza"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17f687"),
        "address" : {
                "coord" : [
                        -74.10465599999999,
                        40.58834
```

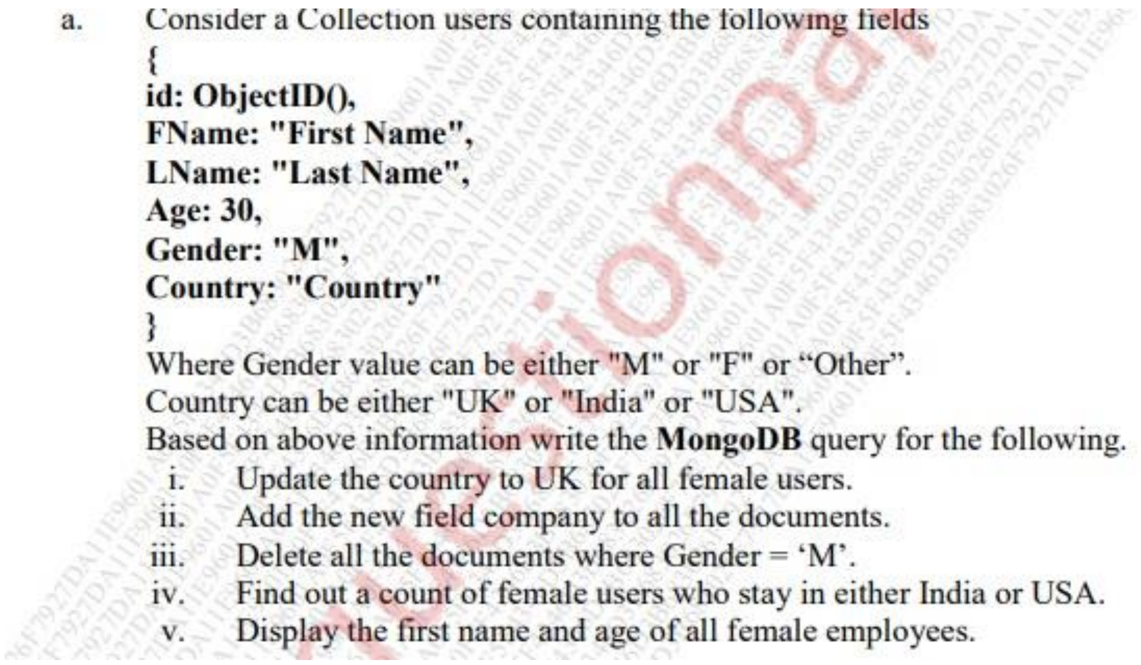
32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those rest which contain 'Mad' as first three letters of its name



```
> db.rest.find({name:{$regex:/^Mad/i,}},{"name":1,"borough":1,"address.coord":1,"cuisine":1}).pr
etty();
{
        "_id" : ObjectId("60f9237491f2ead52b17fb1c"),
        "address" : {
                "coord" : [
                        -73.9860597,
                        40.7431194
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Madison Square"
}
{
        "_id" : ObjectId("60f9237491f2ead52b17fbea"),
        "address" : {
                "coord" : [
                        -73.98302199999999,
                        40.742313
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "name" : "Madras Mahal"
}
{
        "_id" : ObjectId("60f9237591f2ead52b17fea2"),
        "address" : {
                "coord" : [
                        -74.000002,
                        40.72735
```

Practical -2B-Basic Queries

Date: 22-07-2021



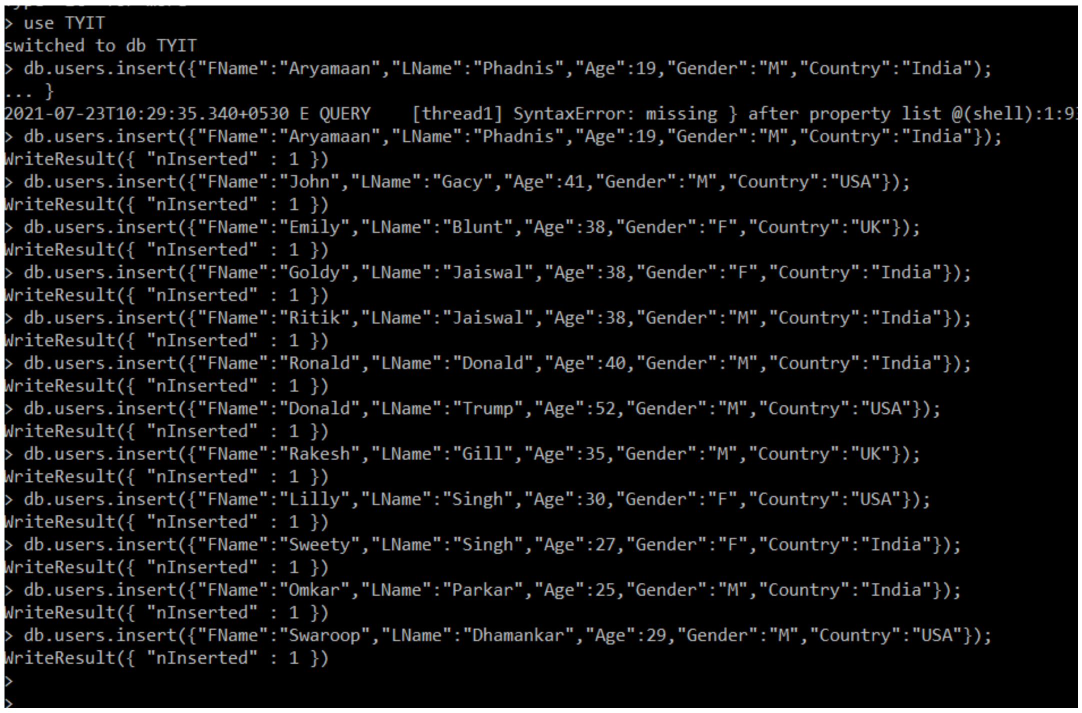
a. Consider a Collection users containing the following fields

{
id: ObjectID(),
FName: "First Name",
LName: "Last Name",
Age: 30,
Gender: "M",
Country: "Country"
}

Where Gender value can be either "M" or "F" or "Other".
Country can be either "UK" or "India" or "USA".
Based on above information write the **MongoDB** query for the following.

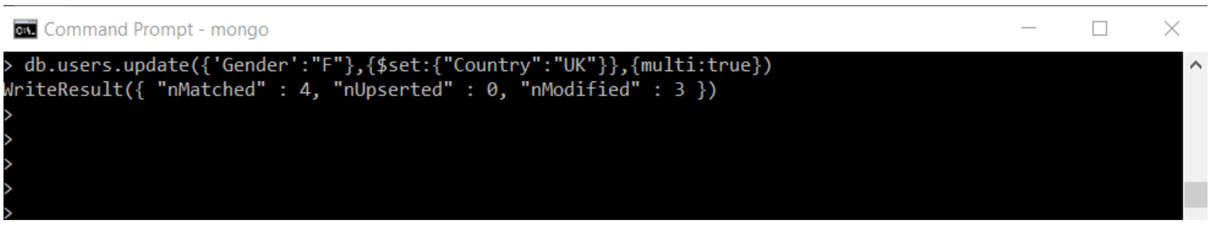
i. Update the country to UK for all female users.
ii. Add the new field company to all the documents.
iii. Delete all the documents where Gender = 'M'.
iv. Find out a count of female users who stay in either India or USA.
v. Display the first name and age of all female employees.

Inserted records



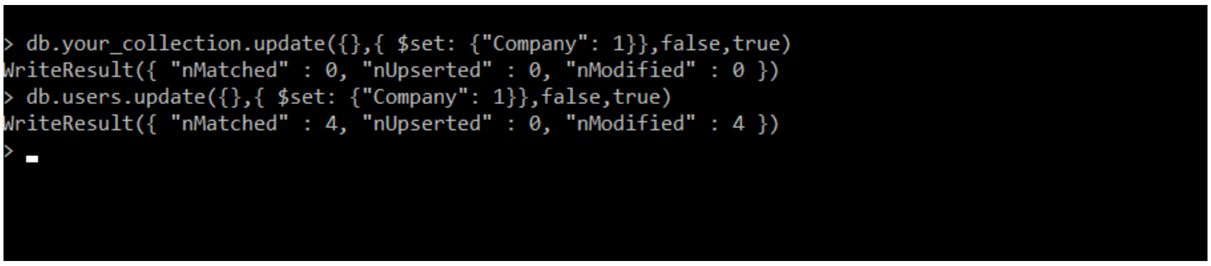
```
> use TYIT
switched to db TYIT
> db.users.insert({"FName":"Aryamaan","LName":"Phadnis","Age":19,"Gender":"M","Country":"India");
... }
2021-07-23T10:29:35.340+0530 E QUERY    [thread1] SyntaxError: missing } after property list @(shell):1:9
> db.users.insert({"FName":"Aryamaan","LName":"Phadnis","Age":19,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"John","LName":"Gacy","Age":41,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Emily","LName":"Blunt","Age":38,"Gender":"F","Country":"UK"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Goldy","LName":"Jaiswal","Age":38,"Gender":"F","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Ritik","LName":"Jaiswal","Age":38,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Ronald","LName":"Donald","Age":40,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Donald","LName":"Trump","Age":52,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Rakesh","LName":"Gill","Age":35,"Gender":"M","Country":"UK"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Lilly","LName":"Singh","Age":30,"Gender":"F","Country":"USA"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Sweety","LName":"Singh","Age":27,"Gender":"F","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Omkar","LName":"Parkar","Age":25,"Gender":"M","Country":"India"});
WriteResult({ "nInserted" : 1 })
> db.users.insert({"FName":"Swaroop","LName":"Dhamankar","Age":29,"Gender":"M","Country":"USA"});
WriteResult({ "nInserted" : 1 })
>
>
```

Q.1 Update the country to UK for all female users.



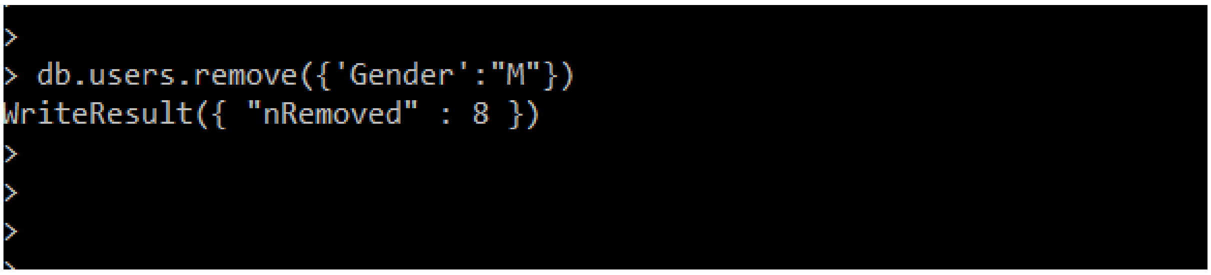
```
> db.users.update({'Gender':"F"},{$set:{"Country":"UK"}},{multi:true})
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 3 })
>
>
>
>
>
```

Q.2 Add the new field company to all documents.



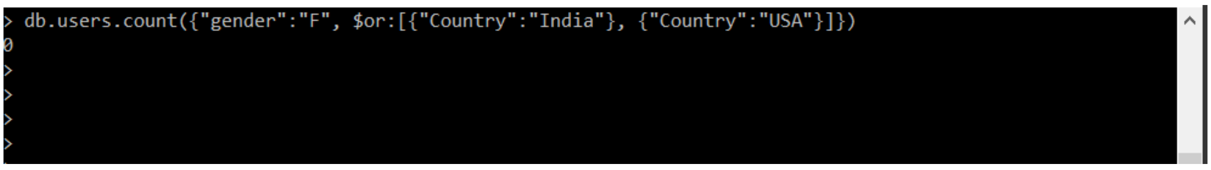
```
> db.your_collection.update({},{ $set: {"Company": 1}},false,true)
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.users.update({},{ $set: {"Company": 1}},false,true)
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 4 })
> _
```

Q.3 Delete all documents with gender= “M”.



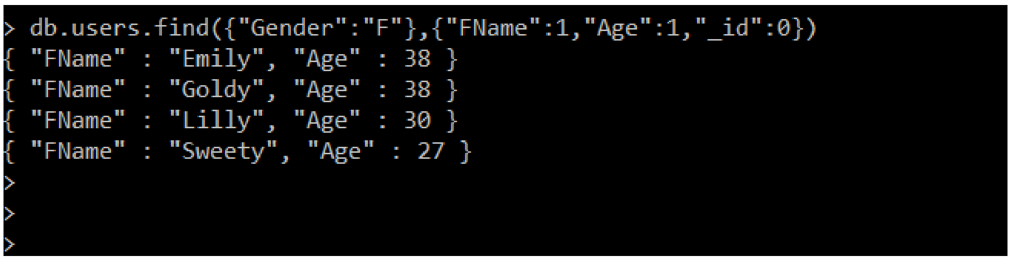
```
>
> db.users.remove({'Gender':"M"})
WriteResult({ "nRemoved" : 8 })
>
>
>
```

Q.4 Find out a count of female users who stay in either India or USA



```
> db.users.count({"gender":"F", $or:[{"Country":"India"}, {"Country":"USA"}]})
0
>
>
>
>
```

Q.5 Display the first name and age of all female employees.

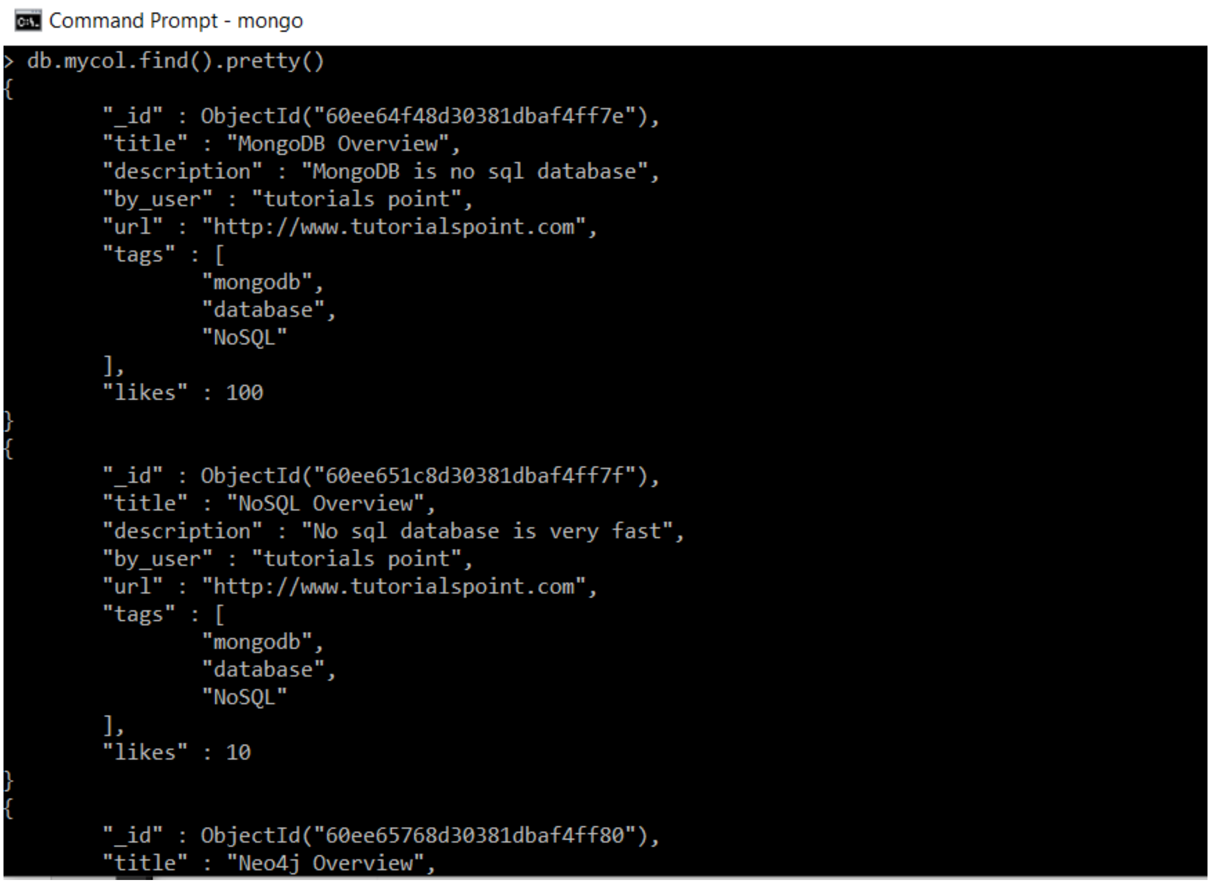


```
> db.users.find({"Gender":"F"},{"FName":1,"Age":1,"_id":0})
{ "FName" : "Emily", "Age" : 38 }
{ "FName" : "Goldy", "Age" : 38 }
{ "FName" : "Lilly", "Age" : 30 }
{ "FName" : "Sweety", "Age" : 27 }
>
>
>
```

Practical 3 – Aggregate functions

Date: 29-07-2021

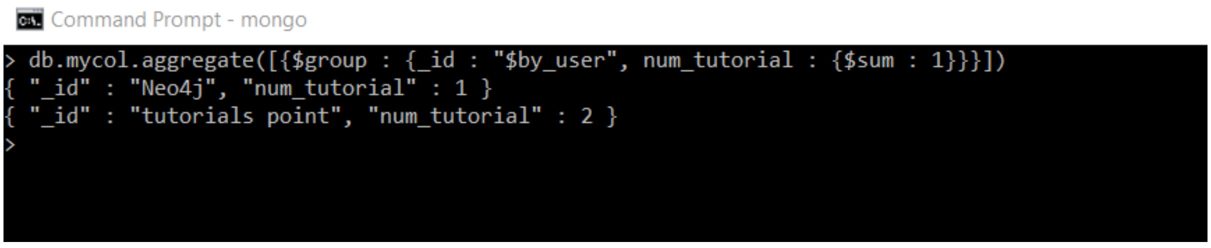
Insert collection in the database and display all the records.



Command Prompt - mongo

```
> db.mycol.find().pretty()
{
        "_id" : ObjectId("60ee64f48d30381dbaf4ff7e"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no sql database",
        "by_user" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
{
        "_id" : ObjectId("60ee651c8d30381dbaf4ff7f"),
        "title" : "NoSQL Overview",
        "description" : "No sql database is very fast",
        "by_user" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 10
}
{
        "_id" : ObjectId("60ee65768d30381dbaf4ff80"),
        "title" : "Neo4j Overview",
```

Q.1 Group by function to get count.



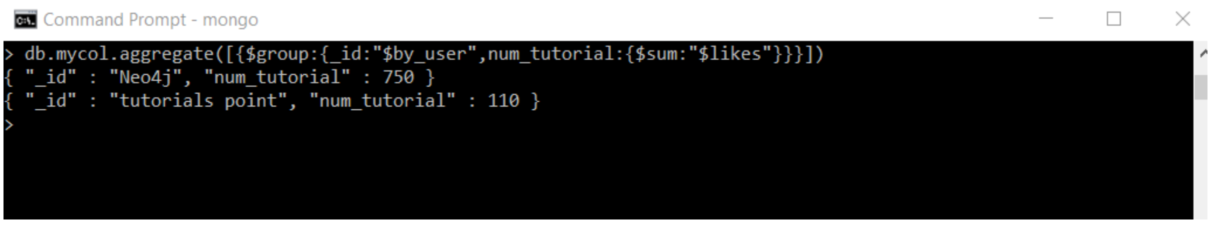
Command Prompt - mongo

```
> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : 1}}}])
{ "_id" : "Neo4j", "num_tutorial" : 1 }
{ "_id" : "tutorials point", "num_tutorial" : 2 }
>
```

Q.2 Sum function.

$sum

Sums up the defined value from all documents in the collection.

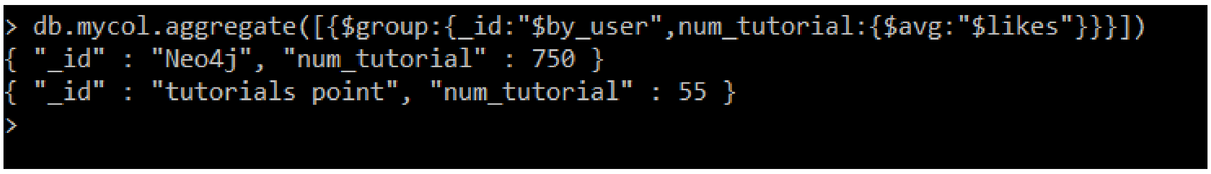

Command Prompt - mongo

```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$sum:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 110 }
>
```

Q.3 Avg function.

$avg

Calculates the average of all given values from all documents in the collection.

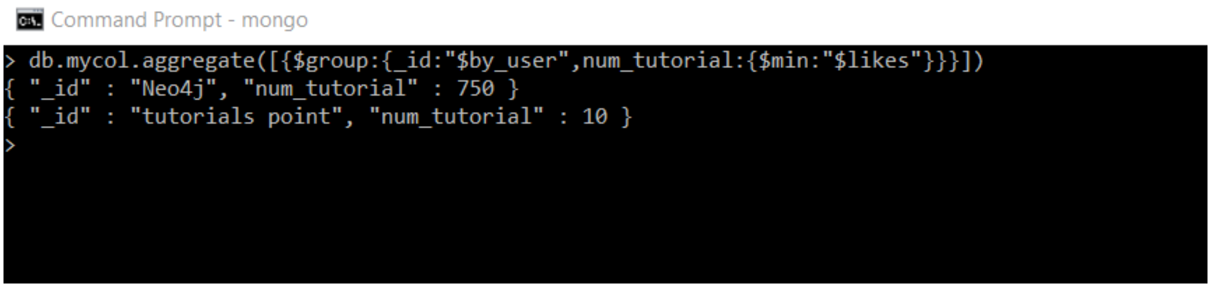


```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$avg:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 55 }
>
```

Q.4 Min function

$min

Gets the minimum of the corresponding values from all documents in the collection.

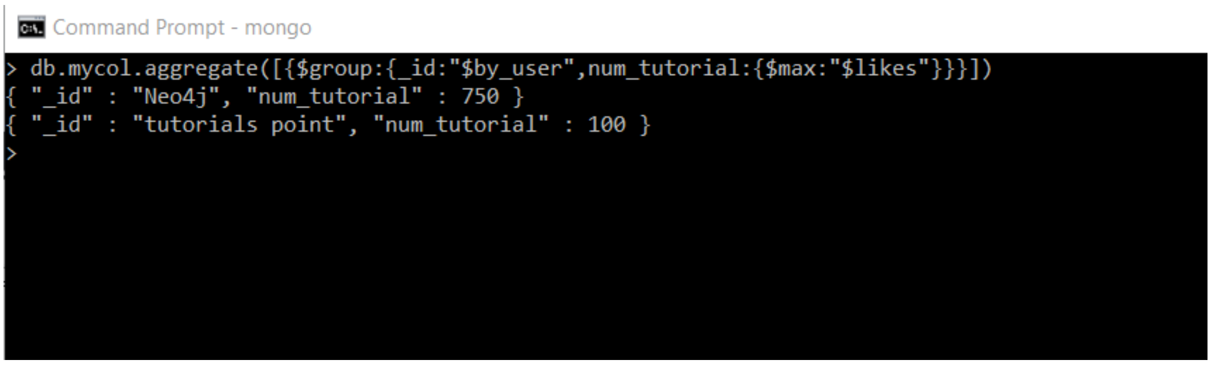

Command Prompt - mongo

```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$min:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 10 }
>
```

Q.5 Max function.

$max

Gets the maximum of the corresponding values from all documents in the collection.

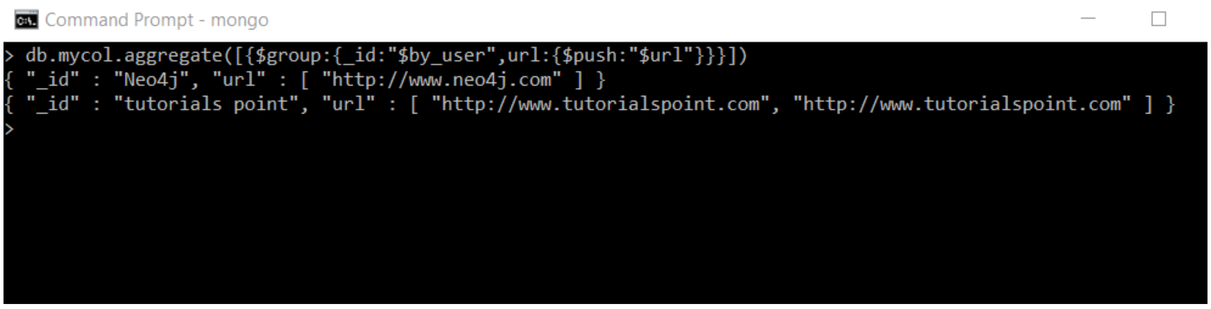


```
> db.mycol.aggregate([{$group:{_id:"$by_user",num_tutorial:{$max:"$likes"}}}])
{ "_id" : "Neo4j", "num_tutorial" : 750 }
{ "_id" : "tutorials point", "num_tutorial" : 100 }
>
```

Q.6 Push function

$push

Inserts the value to an array in the resulting document.

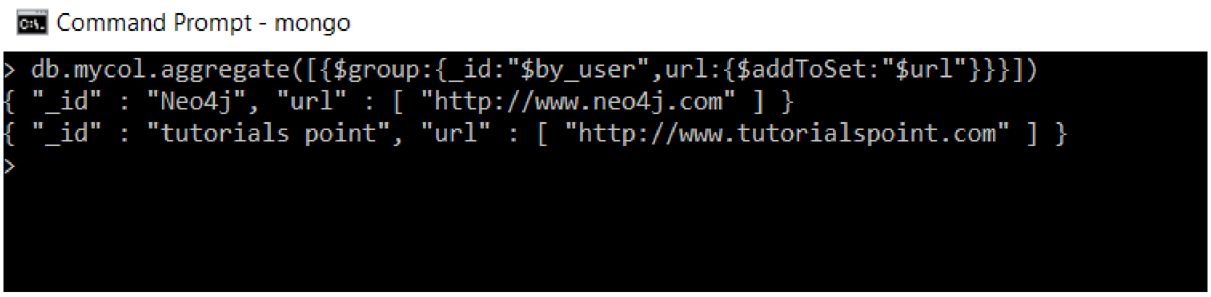


```
> db.mycol.aggregate([{$group:{_id:"$by_user",url:{$push:"$url"}}}])
{ "_id" : "Neo4j", "url" : [ "http://www.neo4j.com" ] }
{ "_id" : "tutorials point", "url" : [ "http://www.tutorialspoint.com", "http://www.tutorialspoint.com" ] }
>
```

Q.7 addToSet function

$addToSet

Inserts the value to an array in the resulting document but does not create duplicates.

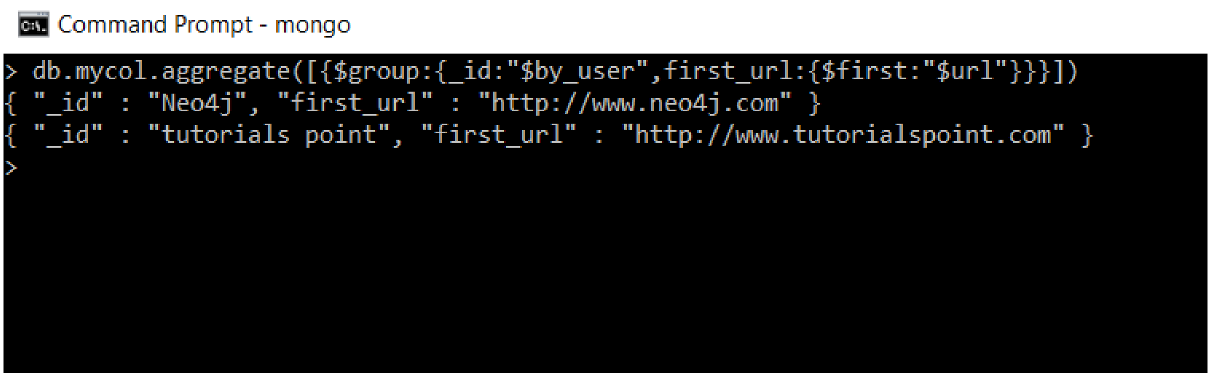


```
> db.mycol.aggregate([{$group:{_id:"$by_user",url:{$addToSet:"$url"}}}])
{ "_id" : "Neo4j", "url" : [ "http://www.neo4j.com" ] }
{ "_id" : "tutorials point", "url" : [ "http://www.tutorialspoint.com" ] }
>
```

Q.8 First function

$first

Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”- stage.

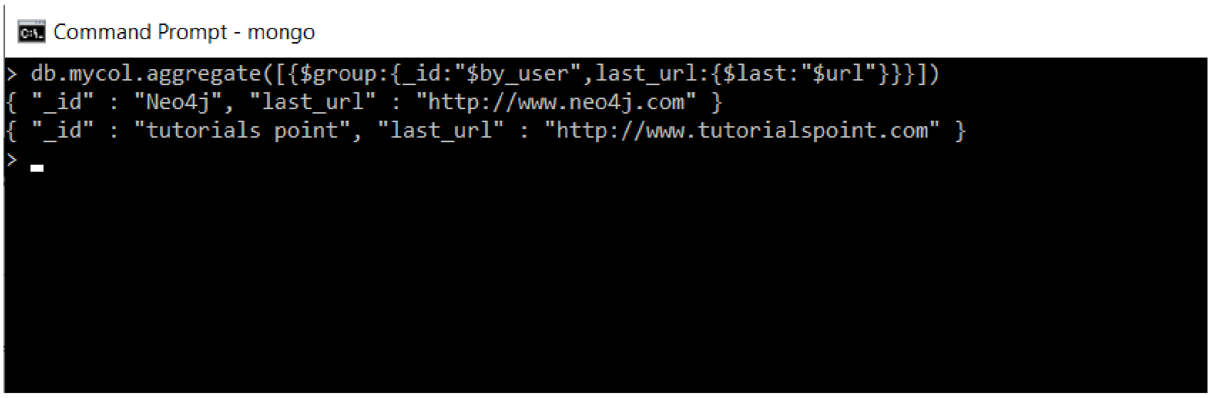

Command Prompt - mongo

```
> db.mycol.aggregate([{$group:{_id:"$by_user",first_url:{$first:"$url"}}}])
{ "_id" : "Neo4j", "first_url" : "http://www.neo4j.com" }
{ "_id" : "tutorials point", "first_url" : "http://www.tutorialspoint.com" }
>
```

Q.9 Last function

$last

Gets the last document from the source documents according to the grouping. Typically, this makes only sense together with some previously applied “$sort”- stage.


Command Prompt - mongo

```
> db.mycol.aggregate([{$group:{_id:"$by_user",last_url:{$last:"$url"}}}])
{ "_id" : "Neo4j", "last_url" : "http://www.neo4j.com" }
{ "_id" : "tutorials point", "last_url" : "http://www.tutorialspoint.com" }
>
```