**MongoDB- Complex Queries**

**MongoDB Exercises – With the Restaurants Data Set**

**Exercise Questions:**

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

db.adresses.find()

1. **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})

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

db.addresses.find({}, {“restaurant\_id” : 1, “name” : 1, “borough” : 1, “cuisine” : 1, “\_id” : 0})

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

db.addresses.find({}, {“restaurant\_id”:1, “name”:1, “borough”:1, “address.zipcode”:1, “\_id”:0})

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

db.addresses.find({“borough”: “Bronx”}).limit(5);

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

db.addresses.find({“borough”: “Bronx”})

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

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

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

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

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

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

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

db.restaurants.find({“address.coord” : {$lt : -95.754168} } )

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

db.addresses.find( { $and:

[

{ “cuisine” : {$ne : “American”} },

{“grades.score” : {$gt: 70} },

{“address.coord” : {$lt : -65.754168} }

]

}

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

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

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

db.addresses.find( {name: /^Wil/), {“restaurant\_id” : 1, “name”: 1, “borough”: 1, “cuisine”: 1 } );

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

db.addresses.find( {name: /ces$/}, { “restaurant\_id” : 1, “name” : 1, “borough” : 1, “cuisine” : 1} )

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

db.addresses.find( {“name” /.\*Reg.\*/}, { “restaurant\_id” : 1, “name” : 1, “borough” : 1, “cuisine” : 1 } )

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

db.addresses.find( { “borough”: “Bronx”, $or: [ {“cuisine” : “American”}, {“cuisine” : “Chinese” } ] } );

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

db.addresses.find( {“borough” :{$in :[“Staten Island”, “Queens”, “Bronx”, “Brooklyn”]}},

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

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

db.addresses.find( “borough” : {$nin :[“Staten Island”, “Queens”, “Bronx”, “Brooklyn”]} },

{ “restaurant\_id” : 1, “name” : 1, “borough” : 1, “cuisine” : 1 });

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

db.adresses.find( {“grades.score” : {$not: {$t: 10} } }, {“restaurant\_id”: 1, “name”: 1, “borough”: 1, “cuisine”: 1} )

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

db.addresses.find( {$or: [ {name: /^Wil/}, {“$and” : [ {“cuisine” : {$ne: “American” } },

{“cuisine” : {$ne : “Chinese”} } ] } ] }, {“restaurant\_id” : 1, “name” : 1, “borough” : 1, “cuisine” : 1} )

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

db.addresses.find( { “grades.date” : ISODate(“2014-08-11T00:00:00Z”), “grades.grade”: “A”, “grades.score” : 11 }, {“restaurant\_id”: 1, “name”: 1, “grades”: 1} )

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

db.addresses.find( { “address.coord.1”: {$gt: 42, $lt: 52} }, {“restaurant\_id” : 1, “name”: 1, “address”: 1, “coord”: 1} )

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

db.addresses.find().sort({“name”: 1})

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

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

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

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

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

db.addresses.find( {“addresses.street”: {$exists: true} } )

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

db.addresses.find( { “address.coord” : {$type: 1} } )

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

db.addresses.find({“grades.score”: {$mod: [7,0]} }, { “restaurant\_id” : 1, “name”: 1, “grades”: 1} )

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

db.addresses.find({name: {$regex: “mon. \*”, $options: “i” } }, { “name”: 1, “borough”: 1, “address.coord”: 1, “cuisine”: 1} )

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

db.addresses.find( {name: $regex: /^Mad/i, } }, { “name”: 1, “borough”: 1, “address.coord”: 1, “cuisine”: 1 } )