**MongoDB Complex Queries**

Name: Ch Rajesh

Gmail-ID: [chagantipati.rajesh09@gmail.com](mailto:chagantipati.rajesh09@gmail.com)

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

db.addresses.find()

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

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 restaurant.

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 restaurant which is in the borough Bronx.

db.addresses.aggregate([{$match:{borough:'Bronx'}},{$limit:5}])

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

db.addresses.aggregate([{$match:{borough:'Bronx'}}])

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

db.addresses.aggregate([{$match:{borough:'Bronx'}},{$skip:5},{$limit:5}])

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

db.addresses.aggregate({$match:{"grades.score":{$gt:90}}})

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

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

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

db.addresses.aggregate({$match:{'address.coord.0':{$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.aggregate({$match:{$and:[{cuisine:{$ne:"American "}},{"grades.score":{$gt:70}},{"address.coord.0":{$lt:-65.754168}}]}})

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

db.addresses.aggregate({$match:{$and:[{cuisine:{$ne:"American "}},{"grades.score":{$gt:70}},{"address.coord.1":{$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.aggregate({$match:{$and:[{cuisine:{$ne:"American "}},{"grades.grade":"A"},{borough:{$ne:"Brooklyn"}}]}},{$sort:{cuisine:-1}})

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

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

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

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

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

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

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