## Practical 2A – Basic Queries

**Import code**

Mongoimport –db dbname –collection collectionname –file restaurant.json

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

**Code:-**

db.collection.find().pretty()

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

## Code:-

db.collection.find({},{"restaurant\_id":1,"name":1,"borough":1,"cuisine"

:1,"\_id":0}).pretty()

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.

**Code:-** db.collection.find({},{"restaurant\_id":1,"name":1,"borough":1,"cuisine"

:1,"\_id":0}).pretty()

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

**Code :-** db.collection.find({},{"restaurant\_id":1,"name":1,"borough":1,"address. zipcode":1}).pretty()

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

**Code:-**

db.collection.find({"borough":"Bronx"}).pretty()

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

**Code:-**

db.collection.find({"borough":"Bronx"}).limit(5).pretty()

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

**Code:-**

db.collection.find({"borough":"Bronx"}).limit(5).pretty()

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

**Code:-**

db.collection.find({grades:{$elemMatch:{"score":{$gt:90}}}}).pretty()

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

**Code:-** db.collection.find({grades:{$elemMatch:{"score":{$gt:80,$lt:100

}}}}). pretty()

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

**Code:-**

db.collection.find({"address.coord":{$lt: -95.754168}}).pretty()

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

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

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

**Code:-** db.collection.find({"cuisine":{$ne:"American"},"grades.score":{$gt:70}

,"address.coord":{$lt: -65.754168}}).pretty()

1. Write a MongoDB query to find the collection 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.

**Code:-** db.collection.find({"cuisine":{$ne:"American"},"grades.grade":"A","borough":"Brooklyn"}).sort({"cuisine": -1}).pretty()

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

**Code:-** db. collection.find({name:/^Wil/},{"restaurant\_id":1,"name":1,"borough ":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({name:/ces$/},{"restaurant\_id":1,"name":1,"borough"

:1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({"name":/.\*Reg.\*/},{"restaurant\_id":1,"name":1,"bor ough":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"c uisine":"Chinese"}]}).pretty()

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

**Code:-**

db. collection.find({"borough":{$in:["StatenIsland","Queens","Bronx"," Brooklyn"]}},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({"borough":{$nin:["StatenIsland","Queens","Bronx", "Brooklyn"]}},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({"grades.score":{$not:{$gt:10}}},{"restaurant\_id":1," name":1,"borough":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({$or:[{name:/^Wil/},{"$and":[{"cuisine":{$ne:"Amer ican"}},{"cuisine":{$ne:"Chinees"}}]}]},{"restaurant\_id":1,"name":1," b orough":1,"cuisine":1}).pretty()

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

**Code:-** db. collection.find({"grades.date":ISODate("201408- 11T00:00:00Z"),"grades.grade":"A","grades.score":11},{"restaurant\_id"

:1,"name":1,"grades":1}).pretty()

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".

**Code:-** db.collection.find({"grades.1.date":ISODate("2014-08-

11T00:00:00Z"),"grades.1.grade":"A","grades.1.score":9},{"restaurant\_i d":1,"name":1,"grades":1}).pretty()

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

**Code:-** dbcollection.find({"address.coord.1":{$gt:42,$lte:52}},{"restaurant\_id"

:1,"name":1,"address":1,"coord":1}).pretty()

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

**Code:-**

db. collection.find().sort({"name":1}).pretty()

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

**Code:-**

db.collection.find().sort({"name": -1}).pretty()

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

**Code:-**

db.collection.find().sort({"cuisine": 1,"borough": -1}).pretty()

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

**Code:-** db.collection.find({"address.street":{$exists:true}}).pretty()

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

**Code:-** db.collection.find({"address.coord":{$type:1}}).pretty()

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

**Code:-** db.collection.find({"grades.score":{$mod:[7,0]}},{"restaurant\_id":1,"na me":1,"grades":1}).pretty()

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

**Code:-** db.collection.find({name:{$regex:"mon,\*",$options:"i"}},{"name":1," b orough":1,"address.coord":1,"cuisine":1}).pretty()

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

**Code:-** db.collection.find({name:{$regex:/^Mad/i}},{"name":1,"borough":1,"ad dress.coord":1,"cuisine":1}).pretty()

**Practical 3 Aggregate functions**

**Insert collection in the database and display all the records.**

**Example …..**

**db.collection.insert([{id:1,name:"Satish",age:77,city:"Ahmednagar",s al:76000}])**

1. **Group by function to get count.**

**Code :-**

db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$sum:1}}}]

)

1. **Sum function.**

**Code:-**

db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$sum:"$sal"

}}}])

1. **Avg function.**

Code:- db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$avg:"$sal"

}}}])

1. **Min function**

**Code:-** db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$min:"$sal"

}}}])

1. **Max function.**

**Code:-** db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$max:"$sa l "}}}])

1. **Push function**

**Code:-** db.collection.aggregate([{$group:{\_id:"$city",num\_tutorial:{$push:"$sal "}}}])

1. **addToSet function**

**Code:-** db.collection.aggregate([{$group:{\_id:"$city",url:{$addToSet:"$sal"}}}]

)

1. **First function**

Code:- db. collection.aggregate([{$group:{\_id:"$city",url:{$first:"$sal"}}}])

1. **Last function**

**Code:-**

db. collection.aggregate([{$group:{\_id:"$city",url:{$last:"$sal"}}}])