{

"address": {

"building": "1007",

"coord": [ ­73.856077, 40.848447 ],

"street": "Morris Park Ave",

"zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [

{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },

{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },

{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },

{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },

{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }

],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

}

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

- db.Restaurant.find();

2. Write a MongoDB query to display the fields restaurant\_id, borough and cuisine, but

exclude the field \_id and name for all the documents in the collection restaurant.

- db.Restaurant.find({}, {\_id: 0, restaurant\_id: 1, borough: 1, cuisine: 1});

3.Write a MongoDB query to find the number of restaurant which is in the borough Abc.

- db.Restaurant.find({borough: ‘Abc’}).count();

4. Write a MongoDB query to display the first 4 restaurant which is in the borough

Bronx.

- db.Restaurant.find({borough: ‘Bronx’}).limit(4);

5. Write a MongoDB query to display the next 3 restaurants after skipping first 4 which

are in the borough Bronx.

- db.Restaurant.find({borough: ‘Bronx’}).skip(4).limit(3);

6. Write a MongoDB query to find the restaurants that achieved a score, less than 40

but more than 10.

- db.Restaurant.find({‘grades.score’: {$gt: 10, $lt: 40}});

7.Write a MongoDB query to find the restaurants which locates in latitude value greater

than -95.754168

- db.Restaurant.find({‘address.coord.0’: {$gt: -95.754168}});

8. Write a MongoDB query to find the restaurants that does not prepare any cuisine of

'American' and their grade score more than 70 and longitude less than -65.754168.

- db.Restaurant.find(

{

cuisine: {$ne: ‘American’},

grades: {$elemMatch: {score: {$gt: 70}}},

‘address.coord.1’: {$lt: -65.754168}

}

);

9. Write a MongoDB query to find the restaurants which does 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.Restaurant.find(

{

cuisine: {$ne: 'American'},

grades: {$elemMatch: {grade: 'A'}},

borough: {$ne: 'Brooklyn'}

}

).sort({cuisine: -1});

10.Write a MongoDB query to find the restaurants which does not prepare any cuisine

of 'American' and achieved a score more than 70 and not located in the longitude less

than ­65.754168. Note : Do this query without using $and operator.

- db.Restaurant.find(

{

cuisine: {$ne: 'American'},

grades: {$elemMatch: {score: {$gt: 70}}},

'address.coord.1': {$gt: -65.754168}

}

);

11. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for

those restaurants which contains 'Wil' as first three letters for its name.

- db.Restaurant.find(

{

name: {$regex: /Wil/}

},{

restaurant\_id: 1,

name: 1,

borough: 1,

cuisine: 1

}

);

12. Write a MongoDB query to find the restaurant Id, name and grades for those

restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014­08­

11T00:00:00Z" among many of survey dates.

- db.Restaurant.find(

{

grades: {

$elemMatch: {

grade: 'A',

score: '11',

date: {$eq:new Date("2014-­08­-11T00:00:000Z")}

}

}

},{

restaurant\_id: 1,

name: 1,

grades: 1

}

);

13. Write a MongoDB query to arranged the name of the cuisine in descending order

and for those same cuisine borough should be in ascending order.

- db.Restaurant.find(

{},

{

cuisine: 1,

borough: 1

}

).sort({cuisine: -1, borough: 1});

14. Write a MongoDB query to know whether all the addresses contains the street or

not.

- db.Restaurant.find(

{

'address.street': {$exists: true}

}

);

15.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 5.

- db.Restaurant.find(

{

'grades.score': {$mod: [5, 0]}

},

{

restaurant\_id: 1,

name: 1,

grades: 1

}

);

16. Write a mongo db query that returns cuisine wimulti: true,se restaurants.

- db.Restaurant.aggregate([

{

$group: {

\_id: '$cuisine',

count: {$sum: 1}

}

}

]);

17. Write a mongoDb query for calculating cuisine wise total and average budget(2

queries..one for total and one for average).

18. Write a mongodb query to get all the restaurants that has maximum revenue in their

cuisine.

19. Write a mongodb query to update to change the score to 60 where the score is 6.

-db.Restaurant.update(

{},

{

$set: {'grades.$[i].score': 60}

},

{

multi: true,

arrayFilters: [{'i.score': 6}]

}

);

20. Create a seperate collection which includes restaurant id and revenue .Update

revenue from that collection into this.

21. Copy this collection to a different database named test and collection name copy

and drop the current database.

- db.Restaurant.find().forEach(function(col) {

db.getSiblingDB('test')['copy'].insert(col)

});

db.dropDatabase();

22. Backup the earlier database and restore it in another one.

- mongodump --db Exercise --collection Restaurant

- mongorestore --collection Restaurant --db Exercise dump/