**NOSQL - MONGODB**

**Post Test**

***Note: Import the records from the “restaurants.json” file into the database before solving the following questions.***

**Basics**

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

Ans) use orders

switched to db orders

db.restaurants.find()

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

Ans) db.restaurants.find(

{ },

{

restaurant\_id : 1,

name : 1,

borough :1,

cuisine: 1

}

) 

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 restaurant.    
Ans) db.restaurants.find(

{},

{

\_id : 0,

restaurant\_id :1,

name :1,

borough : 1,

cuisine :1

} )

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 collection restaurant.

Ans) db.restaurants.find(

{},

{

\_id : 0,

restaurant\_id :1,

name :1,

borough : 1,

“address.zipcode” :1

})

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

Ans) db.restaurants.find(  
 {

borough : “Bronx”

- }  
 )

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

Ans) db.restaurants.find(

{

borough : "Bronx" }

).limit(5)

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

Ans)   db.restaurants.find(

{

borough : "Bronx" })

.skip(5).limit(5)

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

Ans) db.restaurants.find(

{

"grades.score":{$gt:90}

})

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

Ans) db.restaurants.find(

{ "grades.score":{$gt:90, $lt : 100} }

)

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

Ans) db.restaurants.find(

{ "address.coord": { $lt: -95.754168 } }

)

**Intermediate**

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

Ans) db.restaurants.find(

{ "address.coord.1": { $gt: 42, $lte: 52 } },

{

restaurant\_id: 1,

name: 1,

address: 1,

"address": 1

}

)

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

Ans) db.restaurants.find().sort({name:1})

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

Ans) db.restaurants.find().sort({name:-1})

4. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

Ans) db.restaurants.find().sort({cuisine:1},{borough:-1})

5. 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.

Ans) db.restaurants.find(

{ "grades.score": { $mod: [7, 0] } },

{

restaurant\_id: 1,

name: 1,

grades: 1

}

)

**Advance**

Consider another example on a collection named examples that contain documents of the following type:

[

{

\_id: ObjectId("50a8240b927d5d8b5891743c"),

cust\_id: "a123",

ord\_date: new Date("Jan 04, 2019"),

status: 'A',

price: 25,

items: [ { sku: "m", qty: 5, price: 2.5 },

{ sku: "n", qty: 5, price: 2.5 } ]

},

{

\_id: ObjectId("50a8240b927d5d8b5891743d"),

cust\_id: "a124",

ord\_date: new Date("Jan 04, 2019"),

status: 'A',

price: 25,

items: [ { sku: "m", qty: 5, price: 2.5 },

{ sku: "n", qty: 5, price: 2.5 } ]

},

{

\_id: ObjectId("50a8240b927d5d8b5891743e"),

cust\_id: "a123",

ord\_date: new Date("Jan 04, 2019"),

status: 'A',

price: 25,

items: [ { sku: "m", qty: 5, price: 2.5 },

{ sku: "n", qty: 5, price: 2.5 } ]

},

{

\_id: ObjectId("50a8240b927d5d8b5891743c"),

cust\_id: "a124",

ord\_date: new Date("Jan 04, 2019"),

status: 'A',

price: 25,

items: [ { sku: "m", qty: 5, price: 2.5 },

{ sku: "n", qty: 5, price: 2.5 } ]

}

]

**Q1.** Calculate the total price for each customer

Ans) var a = db.examples.mapReduce(function(){

emit(this.cust\_id,this.price)},

function(k,v){

return Array.sum(v); },

{out: {inline:1}});

a.results.forEach(p=>{

print(p.\_id+"-->"+p.value);})

a123-->50

a124-->50

**Q2.** Calculate the total quantity of all sku’s for each customer

Ans) var a = db.examples.mapReduce(function(){

var total=0;

this.items.forEach(function(p)

{total+= p.qty;});

emit(this.cust\_id,this.total);},

function(k,v){

return Array.sum(v); },

{out: {inline:1}});

a.results.forEach(p=>{

print(p.\_id+"-->"+p.value);})

a124-->20

a123-->20