## 1. Querying Restaurants Collection

**3.1 How many “Chinese” (cuisine) restaurants are in “Queens” (borough)?**

**Query:**

db.restaurants.find ({cuisine: "Chinese", borough: "Queens"}).count ()

**Result:**

728

**3.2 What is the \_id of the restaurant which has the grade with the highest ever score?**

**Query**:

db.restaurants.aggregate([

{$project: {"grades.score": 1}},

{$unwind: "$grades"},

{$sort: {"grades.score": -1}},

{$limit: 1}

])

**Result:**

{ "\_id" : ObjectId("5a58d35221cf419367c8a30d"), "grades" : { "score" : 131 } }

**3.3 Add a grade { grade: "A", score: 7, date: ISODate() } to every restaurant in “Manhattan” (borough).**

**Query**:

db.restaurants.update(

{borough: "Manhattan"},

{$push: {

grades: {

grade: "A",

score: 7,

date: ISODate()

}

}},

{multi: true})

**Result:**

WriteResult({ "nMatched" : 10258, "nUpserted" : 0, "nModified" : 10258 })

**3.4 What are the names of the restaurants which have a grade at index 8 with score less then 7? Use projection to include only names without \_id**

**Query:**

db.restaurants.find({"grades.7.score" : {$lt: 7}}, {name: 1, \_id: 0})

**Result**

{ "name" : "El Castillo De Madison" }

{ "name" : "Vee'S Restaurant" }

{ "name" : "Don Alex Restaurant" }

{ "name" : "Gahm Mi Oak Restaurant" }

{ "name" : "Au Za'Atar" }

{ "name" : "Sunshine 27 Seafood Restaurant" }

{ "name" : "New Chung Mee Restaurant" }

{ "name" : "Lucky 11 Bakery" }

{ "name" : "La Cueva Deli & Grocery" }

{ "name" : "Kennedy Fried Chicken" }

{ "name" : "New China Star" }

{ "name" : "Hoy Wong Restaurant" }

{ "name" : "Elena'S Restuarant" }

**3.5 What are \_id and borough of “Seafood” (cuisine) restaurants which received at least one “B” grade in period from 2014-02-01 to 2014-03-01? Use projection to include only \_id and borough**

**Query:**

db.restaurants.find({

cuisine: "Seafood",

grades: {

$elemMatch: {

grade: "B",

date: {

$gt: ISODate("2014-02-01"),

$lt: ISODate("2014-03-01")

}

}

}},

{\_id: 1, borough: 1})

**Result:**

{ "\_id" : ObjectId("5a58d35221cf419367c8d710"), "borough" : "Bronx" }

{ "\_id" : ObjectId("5a58d35221cf419367c8d996"), "borough" : "Manhattan" }

## 2. Indexing Restaurants Collection

**1. Create an index which will be used by this query and provide proof (from explain() or Compass UI) that the index is indeed used by the winning plan: db.restaurants.find({ name: "Glorious Food" })**

**Query:**

db.restaurants.createIndex({ name: 1 })

**Result:**

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"name" : 1

},

"indexName" : "name\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"name" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"name" : [

"[\"Glorious Food\", \"Glorious Food\"]"

]

}

}

}

**2. Drop index from task 4.1**

**Query:**

db.restaurants.dropIndexes()

**Result:**

{

"nIndexesWas" : 3,

"msg" : "non-\_id indexes dropped for collection",

"ok" : 1

}

**3. Create an index to make this query covered and provide proof (from explain()** **or Compass UI) that it is indeed covered:** db.restaurants.find({ restaurant\_id: "41098650" }, { \_id: 0, borough: 1 })

**Query:** db.restaurants.createIndex({ restaurant\_id : 1, borough: 1})

**Result:**

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 15,

"totalKeysExamined" : 1,

"totalDocsExamined" : 0,

"executionStages" : {

"stage" : "PROJECTION",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"transformBy" : {

"\_id" : 0,

"borough" : 1

},

"inputStage" : {

"stage" : "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 0,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"keyPattern" : {

………………………….

**4. Create a partial index on cuisine field which will be used only when filtering on borough equal to “Staten Island”:**

db.restaurants.find({ borough: "Staten Island", cuisine: "American" }) – uses index

db.restaurants.find({ borough: "Staten Island", name: "Bagel Land" }) – does not use index

db.restaurants.find({ borough: "Queens", cuisine: "Pizza" }) – does not use index

**Query:**

db.restaurants.createIndex({ cuisine: 1}, {partialFilterExpression: {"borough": "Staten Island"}})

**Result:**

"winningPlan" : {

"stage" : "FETCH",

"filter" : {

"borough" : {

"$eq" : "Staten Island"

}

},

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"cuisine" : 1

},

"indexName" : "cuisine\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"cuisine" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : true,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"cuisine" : [

"[\"American\", \"American\"]"

]

}

}

}

**5. Create an index to make query from task 3.4 covered and provide proof (from explain() or Compass UI) that it is indeed covered:** db.restaurants.find({"grades.7.score" : {$lt: 7}}, {name: 1, \_id: 0})

**Query:**

db.restaurants.createIndex({"grades.7.score": 1, name: 1})

**Result:**

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"name" : 1,

"\_id" : 0

},

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"grades.7.score" : 1,

"name" : 1

},

"indexName" : "grades.7.score\_1\_name\_1",

"isMultiKey" : true,

"multiKeyPaths" : {

"grades.7.score" : [

"grades"

],

"name" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"grades.7.score" : [

"[-inf.0, 7.0)"

],

"name" : [

"[MinKey, MaxKey]"

]

}

}

},

……………………………………….

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 13,

"executionTimeMillis" : 2,

"totalKeysExamined" : 13