**Insert a resturants data from resturants file and perform the following query:**

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

ANS: db.restaurant.find();

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

ANS: db.restaurant.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.

ANS: db.restaurant.find({}, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 })

1. Write a MongoDB query to display the fields zip code, but exclude the field \_id for all the documents in the collection restaurant.

ANS: db.restaurant.find({}, { "address.zipcode": 1, \_id: 0 })

1. Write a MongoDB query to display all the restaurants which are in the borough **Brooklyn**.

ANS: db.restaurant.find({ borough: "Brooklyn" })

1. Write a MongoDB query to display the **first 5 restaurants**which are in the borough **Brooklyn**.

ANS: db.restaurant.find({ borough: "Brooklyn" }).limit(5)

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

ANS: db.restaurant.find({ borough: "Brooklyn" }).skip(5).limit(5)

1. Write a MongoDB query to find the restaurants that achieved a score of more than 70.

ANS: db.restaurant.find({ "grades.score": { $gt: 70 } })

1. Write a MongoDB query to find the restaurants that achieved a score, of more than 70 but less than 100.

ANS: db.restaurant.find({ "grades.score": { $gt: 70, $lt: 100 } })

1. Write a MongoDB query to find the restaurants that do **not prepare any cuisine of ‘American’** and their grade score of **more than 70**.

ANS: db.restaurant.find({ cuisine: { $ne: 'American' }, 'grades.score': { $gt: 70 } })

1. Write a MongoDB query to find the restaurants which **do not prepare any cuisine of ‘American’**and achieved a grade point **‘A’** not belonging to the borough **Brooklyn**. The document must be displayed according to the **cuisine** in **descending order.**

**ANS: db.restaurant.find({ cuisine: { $ne: 'American' }, borough: { $nin: ['Brooklyn']}, 'grades.grade': 'A' }).sort({ cuisine: -1 })**

1. Write a MongoDB query to find the **restaurant Id, name, borough, and cuisine** for those restaurants which contain ‘**Wil**’ as the **first three letters** of their name.

ANS: db.restaurant.find({ name: { $regex: /^Wil/i } }, { \_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 ‘**Food**’ as the **last three letters** of their name.

ANS: db.resturants.find({ name: /Food$/i }, { 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 ‘**Seafood**’ as three letters somewhere in their name.

ANS: db.resturants.find({ name: /Seafood/i }, { 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** dishes.

ANS: db.restaurant.find({ borough: 'Bronx', cuisine: { $in: ['American', 'Chinese'] } })

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

ANS: db.restaurant.find({ borough: { $in: ['Staten Island', 'Queens', 'Bronx', 'Brooklyn'] }}, {\_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 do not belong to the borough **Staten Island or Queens, or Bronxor Brooklyn**.

ANS: db.restaurant.find({ borough: { $nin: ['Staten Island', 'Queens', 'Bronx', 'Brooklyn'] }}, { \_id: 1, name: 1, borough: 1, cuisine: 1 })

1. Write a MongoDB query to find the r**estaurant Id, name, borough, and cuisine** for those restaurants which achieved a score that is **not more than 10**.

ANS: db.restaurant.find({ 'grades.score': { $lte: 10 }}, { \_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 prepared dishes except ‘**American**’ and ‘**Chinese**’ or the restaurant’s name begins with the letter ‘**Sea**’.

ANS: db.restaurant.find({ $or: [{ cuisine: { $nin: ['American', 'Chinese'] } }, { name: { $regex: /^Sea/i } } ]}, { \_id: 1, name: 1, borough: 1, cuisine: 1 })

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

ANS: db.restaurant.find({}, { \_id: 1, name: 1, borough: 1, cuisine: 1}).sort({ name: 1 })

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

ANS: db.restaurant.find({}, { \_id: 1, name: 1, borough: 1, cuisine: 1 }).sort({ name: -1 })

1. Write a MongoDB query to arrange the name of the **cuisine** in **ascending** order and for that same **cuisine**, the **borough** should be in **descending** order.

ANS: db.restaurant.find({}, { \_id: 1, name: 1, borough: 1, cuisine: 1 }).sort({ cuisine: 1, borough: -1 })

1. Write a MongoDB query to know whether all the addresses **contain** the **building** or not.

ANS: db.restaurant.find({ address: { $not: /building/i } })

1. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants where the 2nd element of the grades array contains a grade of “A” and score 9 on an ISODate “2013–09–11T00:00:00Z”.

ANS: db.restaurant.find({ 'grades.1.grade': 'A', 'grades.1.score': 9, 'grades.1.date': ISODate("2013-09-11T00:00:00Z")}, { \_id: 1, name: 1, grades: 1 })

1. 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 “2013–09–11T00:00:00Z” among many of the survey dates.

ANS: db.restaurant.find({ 'grades.grade': 'A', 'grades.score': 11, 'grades.date': ISODate("2013-09-11T00:00:00Z")}, { \_id: 1, name: 1, grades: 1 })

1. Write a MongoDB query that will select all documents in the restaurants' collection where the **coord** field value is **double**.

ANS: db.restaurant.find({'address.coord': { $type: 'double' } })

1. Write a MongoDB query that 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.restaurant.find({ 'grades.score': { $mod: [7, 0]}}, { \_id: 1, name: 1, grades: 1 })

1. Write a query to find the restaurants with more than three grade surveys (‘grades’ Array contains more than three elements) and display only the name and the number of grades.

ANS: db.restaurant.aggregate([ { $project: {name: 1, numGrades: { $size: "$grades" } }}, { $match: { numGrades: { $gt: 3 } }} ])

1. Write an aggregation pipeline to count the number of restaurants in the borough “Bronx” for each cuisine type. Display the number of restaurants that prepare Caribbean cuisine. A single query is expected that satisfies all the above requirements.

ANS: db.restaurant.aggregate([

{ $match: { borough: "Bronx" }},

{ $group: { \_id: "$cuisine", count: { $sum: 1 } }},

{ $project: { cuisine: "$\_id", count: 1, \_id: 0 }},

{$match: { cuisine: "Caribbean" }}

])