**Bài 1:**

db.inventory.insertOne(  
... { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } ... }  
... )

* Chèn thêm một document vào trong collection “inventory” với các thông tin: item : canvas; qty : 100; tags: [“cotton”]; size: {h: 28, w: 35.5, uom: “cm”}.

db.inventory.insertMany([  
... { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm"} },  
... { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },  
... { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom:  
"cm" } }  
...])

* Chèn thêm nhiều document (3 document) vào trong collection “inventory” với các thông tin:

+ item : journal; qty : 25; tags: [“blank”, “red”]; size: {h: 14, w: 21, uom: “cm”}

+ item : mat; qty : 85; tags: [“gray”]; size: {h: 27.9, w: 35.5, uom: “cm”}

+ item : mousepad; qty : 25; tags: [“gel”, “blue”]; size: {h: 19, w: 22.85, uom: “cm”}

db.inventory.insertMany([  
... { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
... { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },  
... { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  
... { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  
... { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }  
...]);

* Chèn thêm nhiều document (5 document) vào trong collection “inventory” với các thông tin:

+ item : journal; qty : 25; size: {h: 14, w: 21, uom: “cm”}; status: “A”

+ item : notebook; qty : 50; size: {h: 8.5, w: 11, uom: “in”}; status: “A”

+ item : paper; qty : 100; size: {h: 8.5, w: 11, uom: “in”}; status: “A”

+ item : planner; qty : 75; size: {h: 22.85, w: 30, uom: “cm”}; status: “D”

+ item : postcard; qty : 45; size: {h: 10, w: 15.25, uom: “cm”}; status: “A”

db.inventory.insertMany( [

... { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },

... { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

... { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },

... { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },

... { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },

... { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },

... { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },

... { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },

... { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

... { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }

... ] )

* Chèn thêm nhiều document (10 document) vào trong collection “inventory” với các thông tin:

+ item: canvas; qty: 100; size: { h: 28, w: 35.5, uom: "cm" }; status: "A"

+ item: journal; qty: 25; size: { h: 14, w: 21, uom: "cm" }; status: "A"

+ item: mat; qty: 85; size: { h: 27.9, w: 35.5, uom: "cm" }; status: "A"

+ item: mousepad; qty: 25; size: { h: 19, w: 22.85, uom: "cm" }; status: "P"

+ item: notebook; qty: 50; size: { h: 8.5, w: 11, uom: "in" }; status: "P"

+ item: paper; qty: 100; size: { h: 8.5, w: 11, uom: "in" }; status: "D"

+ item: planner; qty: 75; size: { h: 22.85, w: 30, uom: "cm" }; status: "D"

+ item: postcard; qty: 45; size: { h: 10, w: 15.25, uom: "cm" }; status: "A"

+ item: sketchbook; qty: 80; size: { h: 14, w: 21, uom: "cm" }; status: "A"

+ item: sketch pad; qty: 95; size: { h: 22.85, w: 30.5, uom: "cm" }; status: "A"

db.inventory.updateOne(

... {item: "paper"},

... {

... $set: {"size.uom": "cm", status: "p"},

... $currentDate: {lastModified: true}

... }

... )

* Cập nhật một document có item là “paper” với các thông tin cập nhật là: uom: cm; status: p. Cập nhật lại ngày chỉnh sửa cuối cùng là ngày hiện tại.

db.inventory.updateMany(

... {"qty": {$lt: 50}},

... {

... $set: {"size.uom": "in", status: "p"},

... $currentDate: {lastModified: true}

... }

... )

* Cập nhật những document có “qty” nhỏ hơn 50, với các thông tin cập nhật là: uom: cm; status: p. Cập nhật lại ngày chỉnh sửa cuối cùng là ngày hiện tại.

db.inventory.replaceOne(

... {item: "paper"},

... {

... item: "paper", instock: [{warehouse: "A", qty: 60}, {warehouse: "B", qty: 40}]

... }

... )

* Thay thế document có item là “paper” bằng document có nội dung: item: "paper", instock: [{warehouse: "A", qty: 60}, {warehouse: "B", qty: 40}].

db.inventory.deleteMany({status: "A"})

* Xóa những document có status là “A”.

db.inventory.deleteOne({status: "D"})

* Xóa một document đầu tiên được tìm thấy với status: “A”.

db.inventory.find()

* Tìm kiếm trong collection những document với những diều kiện nhất định.

db.inventory.find({status: "D"})

* Tìm kiếm trong collection inventory, những document có status: “D”.

db.inventory.find({status: "A", qty: {$lt: 30}})

* Tìm kiếm những document có status: “A” và qty nhỏ hơn 30.

db.inventory.find({$or: [{status: "A"}, {qty: {$lt: 30}}]})

* Tìm những document hoặc là có status: “A”, hoặc là có qty nhỏ hơn 30.

db.inventory.find({

... status: "A",

... $or: [{qty: {$lt: 30}}, {item: /^p/}]

... })

* Tìm những document có status: “A” và hoặc có qty nhỏ hơn 30 hoặc có item bắt đầu bằng ký tự “p”.

db.inventory.find({size: {h: 14, w: 21, uom: "in"}})

* Tìm những document có size: h: 14, w: 21, uom: "in”. Kết quả sẽ trả về những document có đúng điệu kiện lọc ở trên (thứ tự các điều kiện lọc).

db.inventory.find({size: {w: 21, h: 14, uom: "cm"}})

* Tìm những document có size: w: 21, h: 14, uom: "cm”. Nếu điều kiện lọc bị thay đổi thứ tự với thứ tự các thuộc tính trong document thì câu truy vấn không trả về kết quả nào.

db.inventory.find({"size.uom": "in"})

* Tìm những document có size.uom là “in”.

db.inventory.find({"size.h" : {$lt: 15}})

* Tìm những document có size.h nhỏ hơn 15.

db.inventory.find({"size.h" : {$lt: 15}, "size.uom": "in", status: "D"})

* Tìm những docmment có size.h nhỏ hơn 15, size.uom là “in” và status là “D”.

db.inventory.insertMany([

... { item: "journal", qty: 25, tags: ["blank", "red"], dim\_cm: [ 14, 21 ] },

... { item: "notebook", qty: 50, tags: ["red", "blank"], dim\_cm: [ 14, 21 ] },

... { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim\_cm: [ 14, 21 ] },

... { item: "planner", qty: 75, tags: ["blank", "red"], dim\_cm: [ 22.85, 30 ] },

... { item: "postcard", qty: 45, tags: ["blue"], dim\_cm: [ 10, 15.25 ] }

... ])

* Chèn vào collection những document.

db.inventory.find( { tags: ["red", "blank"] } )

* Tìm những document có tags là mảng: “red”, “blank”.

db.inventory.find( { tags: { $all: ["red", "blank"] } } )

* Tìm những document có tags có những thuộc tính “red” và “blank”.

db.inventory.find({tags: "red"})

* Tìm những document có tags có chứa “red”.

db.inventory.find({dim\_cm: {$gt: 25}})

* Tìm những document có dim\_cm lớn hơn 25.

db.inventory.find({dim\_cm: {$gt: 15, $lt: 20}})

* Tìm những document có dim\_cm lớn hơn 15 và nhỏ hơn 20.

db.inventory.find({dim\_cm: {$elemMatch: {$gt: 22, $lt: 30}}})

* Tìm những document nằm trong mảng có dim\_cm lớn hơn 22 và nhỏ hơn 30.

db.inventory.find({"dim\_cm.1": {$gt: 25}})

* Tìm những document có dim\_cm lớn hơn 25 nằm ở chỉ mục thứ 2 trong mảng.

db.inventory.find({"tags": {$size: 3}})

* Tìm những document có tags có số lượng phần tử là 3.

db.inventory.insertMany( [

... { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15

... } ] },

... { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },

... { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 }

... ] },

... { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5

... } ] },

... { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty:

... 35 } ] }

... ])

* Chèn những document với các thông tin trên vào trong collection.

db.inventory.find({"instock": {warehouse: "A", qty: 5}})

* Tìm những document với điều kiện là: "instock": {warehouse: "A", qty: 5}. Kết quả sẽ phụ thuộc vào thứ tự các điều kiện trong câu truy vấn.

db.inventory.find({"instock": {qty: 5, warehouse: "A"}})

* Tìm những document với điều kiện là: "instock": qty: 5, warehouse: "A". Sai thứ tự với thứ tự các thuộc tính có trong các document nên kết quả sẽ trả về null.

db.inventory.find({"instock.qty": {$lte: 20}})

* Tìm những document có instock.qty nhỏ hơn 20.

db.inventory.find({"instock.0.qty": {$lte: 20}})

* Tìm những document có instock.qty nhỏ hơn 20 nằm ở vị trí đầu tiên trong mảng kết quả tìm thấy.

db.inventory.find({"instock": {$elemMatch: {qty: 5, warehouse: "A"}}})

* Tìm những document có instock là một mảng có qty: 5 và warehouse là “A”, không cần phân biệt thứ tự vì nó là một mảng.

db.inventory.find({"instock": {$elemMatch: {qty: {$gt: 10, $lte: 20}}}})

* Tìm những document có instock là một mảng có qty lớn hơn 10 và nhỏ hơn bằng 20.

db.inventory.find({"instock.qty": {$gt: 10, $lte: 20}})

* Tìm những document có instock.qty lớn hơn 10 và nhỏ hơn hoặc bằng 20.

db.inventory.find({"instock.qty": 5, "instock.warehouse": "A"})

* Tìm những document ccos instock.qty là 5 và instock.warehouse là “A”.

db.inventory.insertMany( [

... { item: "journal", status: "A", size: { h: 14, w: 21, uom: "cm" }, instock: [ {

... warehouse: "A", qty: 5 } ] },

... { item: "notebook", status: "A", size: { h: 8.5, w: 11, uom: "in" }, instock: [ {

... warehouse: "C", qty: 5 } ] },

... { item: "paper", status: "D", size: { h: 8.5, w: 11, uom: "in" }, instock: [ {

... warehouse: "A", qty: 60 } ] },

... { item: "planner", status: "D", size: { h: 22.85, w: 30, uom: "cm" }, instock: [ {

... warehouse: "A", qty: 40 } ] },

... { item: "postcard", status: "A", size: { h: 10, w: 15.25, uom: "cm" }, instock: [ {

... warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }

... ])

* Chèn thêm các document vào trong collection với các thông tin.

db.inventory.find( { status: "A" } )

* Tìm những document có status: “A”.

db.inventory.find( { status: "A" }, { item: 1, status: 1 } )

* Tìm những document có status là “A”, hiển thị tất cả các field của item và status.

db.inventory.find( { status: "A" }, { item: 1, status: 1, \_id: 0 } )

* Tìm những document có status là “A”, hiển thị tất cả các fiels của item và status nhưng không hiển thị \_id.

db.inventory.find( { status: "A" }, { status: 0, instock: 0 } )

* Tìm những document có status là “A”, thông tin hiển thị không bao gồm các field: status và instock.

db.inventory.find({status: "A"}, {item: 1, status: 1, "size.uom": 1})

* Tìm những document có status là “A”, thông tin hiển thị gồm các field: item, status, size.uom.

db.inventory.find({status: "A"}, {"size.uom": 0})

* Tìm những document có status là “A”, thông tin hiển thị gồm tất cả các field trừ field size.uom.

db.inventory.find({status: "A"}, {item: 1, status: 1, "instock.qty: 1"})

* Tìm những document có status là “A”, thông tin hiển thị gồm có các field: item, status, instock.qty.

db.inventory.find({status: "A"}, {item: 1, status: 1, instock: {$slice: -1}})

* Tìm những document có status “A”, thông tin hiển thị gồm có: item, status, phần tử cuối cùng trong mảng instock.

db.inventory.insertMany([

... { \_id: 1, item: null },

... { \_id: 2 }

... ])

* Chèn thêm vào trong collection 2 document với các thông tin như trên.

db.inventory.find({item: null})

* Tìm những document có item là giá trị null.

db.inventory.find({item: {$type: 10}})

* Tìm những document có item là giá trị null. $type: 10 ở đây là kiểu dữ trong BSON Type, $type: 10 tương ứng với giá trị null.

db.inventory.find({item: {$exists: false}})

* Tìm những document có không bao gồm thuộc tính item trong các thuộc tính của document này.

db.inventory.find().sort({KEY: 1})

* Sắp xếp tất cả các document trong collection inventory theo chiều tăng dần của \_id.

db.inventory.find().sort({KEY: -1})

* Sắp xếp tất cả các document trong collection inventory theo chiều giảm dần của **\_id.**

**Bài 2:**

db.zips.find({})

* Hiển thị tất cả các document có trong collection zips.

db.zips.insert({city: "Ho Chi Minh City", loc: [-12.12, 23.23], pop: 90000, state: "HCM"})

* Chèn thêm một document vào trong collection với các thông tin tương ứng.

db.zips.find({city: "Ho Chi Minh City"})

* Tìm document có city là “Ho Chi Minh City”.

db.zips.find({city: "PALMER"})

* Tìm document có city là “PALMER”.

db.zips.find({city: "FISHER ISLAND"}, {pop: 1, city: 1, \_id: 0})

* Tìm các document có city là “FISHER ISLAND”, thông tin hiển thị gồm có các field: pop, city.

db.zips.find({pop: {$gt: 100000}})

* Tìm các document có dân số lớn hơn 100000.

db.zips.find({pop: {$gt: 10, $lt: 50}})

* Tìm các document có dân số lớn hơn 10 và nhỏ hơn 50.

db.zips.find({state: "MA", pop: {$gt: 500}}, {city: 1, pop: 1, \_id: 0})

* Tìm các thành phố của bang “MA” có số dân lớn hơn 500, thông tin gồm có thành phố, dân số.

db.zips.distinct("state").sort()

* Tìm tất cả các bang, không được trùng lặp.

db.zips.distinct("state", {pop: {$gt: 100000}})

* Tìm tất cả các bang có chứa thành phố có số dân lớn hơn 100000.

db.zips.aggregate([{$group: {\_id: "$state", avgpop: {$avg: "$pop"}}}])

* Tính số dân trung bình của mỗi bang, thông tin gồm có: \_id, trung bình số dân.

db.zips.aggregate([{$match: {state: "WA"}}, {$group: {\_id: "$state", noCity: {$sum: 1}}}])

* Tính tổng số thành phố của bang “WA”, thông tin gồm có: \_id, tổng số thành phố.

db.zips.aggregate([{$group: {\_id: "$state", noCity: {$sum: 1}}}])

* Tính số thành phố của mỗi bang.

db.zips.aggregate({$group: {\_id: "$state", totalPop: {$sum: "$pop"}}}, {$match: {totalPop: {$gt: 10000000}}})

* Tìm ra các bang có tổng dân số lớn hơn 10000000.

**Bài 3:**  
1. Viết câu truy vấn hiển thị tất cả các document trong collection restaurants.

db.restaurants.find().pretty()

2. Viết câu truy vấn hiển thị các fields restaurant\_id, name, borough, cuisine cho các documnent trong collection restaurant.

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

3. Viết câu truy vấn hiển thị các fields restaurant\_id, name, borough, cuisine, không hiển thị field \_id cho các document trong collection restaurant.

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

4. Viết câu truy vấn hiển thị các fields restaurant\_id, name, borough, zip code, không hiển thị field \_id cho các document trong collection restaurant.

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

5. Viết câu truy vấn hiển thị tất cả các restaurant có quận (borough) là Bronx.

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

6. Viết câu truy vấn hiển thị 5 restaurant đầu tiên có quận (borough) là Bronx.

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

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

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

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

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

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

db.restaurants.find({grades: {$elemMatch:{"score": {$gt: 80, $lt: 100}}}})

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

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

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

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

12. Write a MongoDB query to find the restaurants which do 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.restaurants.find({"cuisine": {$ne: "American"}, "grades.score": {$gt: 70}, "address.coord": {$lt: -65.754168}}).pretty()

13. Write a MongoDB query to find the restaurants 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.

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

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

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

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

db.restaurants.find({name: /ces$/}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1}).pretty()

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

db.restaurants.find({name: /.\*Reg\*./}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1}).pretty()

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

db.restaurants.find({"borough": "Bronx", $or: [{"cuisine": "American"}, {"cuisine": "Chinese"}]}).pretty()

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

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

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

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

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

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

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

db.restaurants.find({$or: [{name: /^Wil/}, {"$and": [{"cuisine": {$ne: "American"}}, {"cuisine" : {$ne :"Chinees"}}]}]}, {"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1}).pretty()

22. 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 "2014-08-11T00:00:00Z" among many of survey dates.

db.restaurants.find({ "grades.date": ISODate("2014-08-11T00:00:00Z"), "grades.grade": "A" ,"grades.score" : 11}, {"restaurant\_id" : 1,"name":1, "grades": 1}).pretty()

23. 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".

db.restaurants.find({"grades.1.date": ISODate("2014-08-11T00:00:00Z"), "grades.1.grade": "A" , "grades.1.score" : 9}, {"restaurant\_id" : 1,"name":1, "grades": 1}).pretty()

24. 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.

db.restaurants.find({"address.coord.1": {$gt : 42, $lte : 52}}, {"restaurant\_id" : 1,"name":1,"address":1,"coord":1}).pretty()

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

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

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

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

27. 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.

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

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

db.restaurants.find({"address.street": {$exists: true}})

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

db.restaurants.find({"address.coord": {$type: 1}})

30. 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.

db.restaurants.find({ "grades.score": {$mod: [7,0]}}, {"restaurant\_id" : 1,"name":1, "grades": 1}).pretty()

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

db.restaurants.find({name: {$regex: "mon.\*", $options: "i"}}, {"name": 1, "borough": 1, "address.coord": 1, "cuisine": 1}).pretty()

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

db.restaurants.find({name: {$regex: /^Mad/i,}}, {"name": 1, "borough": 1, "address.coord": 1, "cuisine": 1}).pretty()