Mongodb exrcise

<https://www.w3resource.com/mongodb-exercises/#MongoDB_restaurants>

# MongoDB query exercise

working with Chinh Le @ https://github.com/ldchinhcr

1. display all the documents in the collection restaurants.

```db.rest.find()```

2. display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.

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

3. display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

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

4. display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

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

5. display all the restaurant which is in the borough Bronx.

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

6. display the first 5 restaurant which is in the borough Bronx.

```db.rest.find({"borough" : "Bronx"}).limit(5)```

7. display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

```db.rest.find({"borough" : "Bronx"}).skip(5).limit(5)```

8. find the restaurants who achieved a score more than 90.

```db.rest.find({"grades.score" : {"$gt" : 90}})```

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

```db.rest.find({$and : [{"grades.score" : {"$gt" : 90}},{"grades.score" : {"$lt" : 100}}]})```

10. find the restaurants which locate in latitude value less than -95.754168.

```db.rest.find({"address.coord.0" : {$lt : -95.754168}})```

11. 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.rest.find({$and : [{"cuisine" : {$ne : "American "}}, {"address.coord.0" : {$lt : -65.754168}}, {"grades.score" : {$gt : 70}}]})```

12. find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

Note : Do this query without using $and operator.

```db.rest.find({$and : [{"cuisine" : {$ne : "American "}}, {"address.coord.1" : {$lt : -65.754168}}, {"grades.score" : {$gt : 70}}]})```

13. 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.rest.find({$and : [{"cuisine" : {$ne : "American "}}, {"grades.grade" : "A"}, {"borough" : {$ne : "Brooklyn "}}]}).sort({cuisine : -1})```

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

```db.rest.find({"name" : { $regex: /^Wil.\*/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

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

```db.rest.find({"name" : { $regex: /.\*ces$/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

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

```db.rest.find({"name" : { $regex: /Reg/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

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

```db.rest.find({borough: "Bronx", cuisine: {$in: ["American ","Chinese"]}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

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

```db.rest.find({$or: [{"borough": "Staten Island"}, {"borough": "Bronxor Brooklyn"}, {"borough": "Queens"}]}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

19. 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.rest.find( {borough: {$nin: ["Staten Island","Queens","Bronx","Brooklyn"]}} , {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

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

```db.rest.find({"grades.score": {$lte: 10}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

21. find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinese' or restaurant's name begins with letter 'Wil'.

```db.rest.find({$nor: [{cuisine: {$in: ["American ","Chinese"]}},{name: /^Wil.\*/}]},{\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})```

22. 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.rest.find({"grades" : {$elemMatch: {"date": ISODate("2014-08-11T00:00:00Z"), "grade":"A", "score":11}}}, {\_id:0, restaurant\_id:1, name:1, grades:1})```

23. 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.rest.find({$and: [{"grades.1.grade":"A"}, {"grades.1.score": 9}, {"grades.1.date": ISODate("2014-08-11T00:00:00Z")}]},{\_id:0, restaurant\_id:1, name:1, grades:1}).pretty()```

24. 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.rest.find({$and : [{"address.coord.1": {$gt : 42}},{"address.coord.1": {$lte : 52}}]}, {\_id:0, restaurant\_id:1, name:1, address:1})```

25. arrange the name of the restaurants in ascending order along with all the columns.

```db.rest.find({},{\_id:0, name:1}).sort( {name: 1})```

26. arrange the name of the restaurants in descending along with all the columns.

```db.rest.find({},{\_id:0, name:1}).sort( {name: -1})```

27. arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```db.rest.find({}, {\_id:0, cuisine:1, borough:1}).sort({cuisine: 1, borough: -1})```

28. know whether all the addresses contains the street or not.

```

With Street:

db.rest.find({"address.street": {$regex: /Street/}}).pretty()

```

```

Not with street:

db.rest.find({"address.street": {$ne: {$regex: /Street/}}}).pretty()

```

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

```db.rest.find({"address.coord": {$type: "double"}}, {\_id:0, address: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.rest.find({"grades": {$elemMatch: {"score": {$mod: [7,0]}}}},{\_id:0, restaurant\_id:1, name:1, grades:1})```

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

```db.rest.find({name: {$regex: /mon/}},{\_id:0, name:1, borough:1, "address.coord":1, cuisine:1})```

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

```db.rest.find({name: {$regex: /^Mad.\*/}},{\_id:0, name:1, borough:1, "address.coord":1, cuisine:1})```

To work with an employee dataset in MongoDB and execute the queries you requested, I'll walk you through how to design the dataset and provide the queries for each scenario. The employee collection might look like this:

**Example Document Structure for Employees in MongoDB:**

json

Copy code

{

"\_id": ObjectId("..."), // MongoDB generated unique ID

"name": "John Doe",

"employee\_id": "E123",

"department": "Sales",

"joining\_date": ISODate("1999-05-20T00:00:00Z"),

"pay\_scale": 45000,

"promotions": 2,

"description": "Good performance, loyal employee"

}

**MongoDB Queries for Each Scenario:**

**1) Get all employee data who joined before the year 2000 or after 2010.**

To filter employees based on their joining date:

javascript

Copy code

db.employees.find({

$or: [

{ joining\_date: { $lt: ISODate("2000-01-01T00:00:00Z") } },

{ joining\_date: { $gt: ISODate("2010-01-01T00:00:00Z") } }

]

});

**2) Add an employee named "Sunil" to the employee "Anil".**

Assuming you want to add "Sunil" as a new employee and associate them with "Anil", you can insert a new document with "Anil" as a reference (or embedded document). Let's assume we are adding Sunil as a new employee to the database:

javascript

Copy code

db.employees.insertOne({

name: "Sunil",

employee\_id: "E124",

department: "Marketing",

joining\_date: ISODate("2022-06-15T00:00:00Z"),

pay\_scale: 55000,

promotions: 1,

description: "Good employee",

reference\_employee: "Anil"

});

Here, "reference\_employee": "Anil" represents that Sunil is associated with Anil.

**3) Find all employees that have a description that contains the word "Good" and not the word "bad".**

To query documents based on text in the description field:

javascript

Copy code

db.employees.find({

description: { $regex: "Good", $not: { $regex: "bad" } }

});

This query will find employees whose description contains the word "Good" but does **not** contain the word "bad".

**4) Delete the employee record of "Ramesh".**

To delete an employee with the name "Ramesh":

javascript

Copy code

db.employees.deleteOne({

name: "Ramesh"

});

This will delete the first document that matches the name "Ramesh". If there are multiple Ramesh records and you want to delete all, use deleteMany() instead.

**5) Find all employee details who get at least three (3) promotions till date.**

To find employees with three or more promotions:

javascript

Copy code

db.employees.find({

promotions: { $gte: 3 }

});

This will return employees who have received at least 3 promotions.

**6) Get all documents where department includes "Production & Control".**

To query departments containing the phrase "Production & Control":

javascript

Copy code

db.employees.find({

department: /Production & Control/

});

This uses a regular expression to match any department field that contains the phrase "Production & Control".

**7) Find all employee details whose pay scale is greater than 40k and joined in the last three years.**

To filter employees based on pay scale and joining date within the last 3 years:

javascript

Copy code

db.employees.find({

pay\_scale: { $gt: 40000 },

joining\_date: { $gte: new Date(new Date().setFullYear(new Date().getFullYear() - 3)) }

});

This will return employees whose pay scale is greater than 40,000 and who joined in the last 3 years.

**8) Sort all the documents according to employee joining date.**

To sort employees by their joining date in ascending order:

javascript

Copy code

db.employees.find().sort({ joining\_date: 1 });

To sort in descending order (most recent first):

javascript

Copy code

db.employees.find().sort({ joining\_date: -1 });

**Conclusion:**

These are the MongoDB queries to handle each of the scenarios you've described. The data manipulation tasks (such as adding, deleting, and updating) are fairly straightforward, and the querying examples show how to filter and sort employee data based on your specific requirements.