**Practical 7 :**

**Create a collection by the name “food” and then insert documents into the food collection**

db.food.insert({\_id:1, fruits:['banana', 'apple', 'cherry']});

db.food.insert({\_id:2, fruits:['orange', 'butterfruit', 'mango']});

db.food.insert({\_id:3, fruits:['peneapple', 'strawberry', 'grapes']});

**Find those collections from the “food” collection which the fruits array constituted of ‘banana’, ‘apple’ and ‘cherry’.**

db.food.find({fruits:['banana','apple','cherry']});

**Find those collections from the “food” collection which the fruits array having ‘banana’ as an element.**

db.food.find({fruits: 'banana'});

**Find those collections from the “food” collection which the fruits array having ‘banana’ in the first index position**

db.food.find({'fruits.0': 'banana'});

**Find those collections from the “food” collection where size of the array is two.**

db.food.find({fruits: {$size: 2}});

**Find collections from the “food” collection and display first two elements**

db.food.find({},{fruits: {$slice: 2}});

**Find collections from the “food” collection and display two elements starting with the element at 1st index position**

db.food.find({},{fruits: {$slice: [1,2]}});

**Update the element at 0th index position of document with \_id:3 by ‘apple’**

db.food.update({\_id:3},{$set: {'fruits.0': 'apple'}});

**Update the document with \_id:1 and push new key value pairs in the fruits array**

db.food.update({\_id:1},{$push: {price:{banane: 50, apple:150, cherry:100}}});

**Update document with \_id:3 by adding an element an ‘Orange’**

db.food.update({\_id:3},{$addToSet:{fruits:'Orange'}});

**Update document with \_id:3 by popping an element**

db.food.update({\_id:3},{$pop:{fruits:1}});

**Update document with \_id:3 by popping an element from the beginning of the array**

db.food.update({\_id:3},{$pop:{fruits:-1}});

**Update document with \_id:2 by popping two elements from the list : ‘orange’ and ‘mango’.**

db.food.update({\_id:2},{$pullAll:{fruits:['orange','mango']}});

**To pull out an array element based on index position’.**

db.food.update({\_id:1},{$pull:{"fruits":null}});

**Cursors**

To create a collection of “a”, “b”,….”z”.by the name “alphabets”

db.alphabets.insert({\_id:1, alphabet: “a”});

…….

db.alphabets.insert({\_id:26, alphabet: “z”})

1) Display A to Z from above table using cursor.

2) Make a table just like below and perform addition of two number and insert that number in last column of the table.

|  |  |  |
| --- | --- | --- |
| No1 | No2 | Answer |
| 1 | 1 | 2 ( do with cursor) |

**Pratical 8:**

1. **Create one simple function whose return the same value which you have passed.**

function ret(n)

{

return n;

}

print("value return: "+ ret(10)); //function call

1. **Create a function to display addition of two number;**

function sum(a,b)

{

var c=a+b;

print("Sum of "+a +" and "+ b +" is: "+c);

}

sum(2,3); //function call

1. **Create a function to display factorial number using recursion.**

function rfact(n)

{

if (n ==1 || n==0)

return 1;

else

return n\*rfact(n-1);

}

var n=rfact(5); //function call

print("factorial : "+n)

function fact(num)

{

var i,f=1;

for(i=1; i<=num;i++)

{

f=f\*i;

}

print("factorial of number:"+f);

}

fact(5)//function call

1. **Create a function to display an Armstrong number.**

function arms(n)

{

var p,ct=0,m,s=0,temp;

p=n;

temp=String(n);

ct=temp.length;

while(p>0)

{

m=p%10;

p=parseInt(p/10);

s=s+Math.pow(m,ct)

}

if(n==s)

{

print(n+" number is armstrong number")

}

else

{

print(n+" number is not an armstrong number")

}

}

arms(153) //function call

1. **Create a function to display an Fibonacci series less than 100.**

function fib(num)

{

var a=0,b=1,c,i;

print(a);

print(b);

for(i=2;i<num;i++)

{

c=a+b;

if(c<=100)

{

print(c);

a=b;

b=c;

}

}

}

fib(15)//function call

**Pratical 9:**

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

db.rest.find()

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

db.rest.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.**

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

**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.**

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

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

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

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

db.rest.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.**

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

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

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

db.rest.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.**

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

**10. Write a MongoDB query to find the restaurants which locate in latitude**

**value less than -95.754168.**

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

**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.rest.find({"cuisine" : {$ne : "American "},"grades.score" :{$gt: 70},"address.coord" : {$lt : -65.754168}});

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

**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.rest.find({name: /^Wil/},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

**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.rest.find({name: /Reg/},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

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

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

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

**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.rest.find({"grades.score":{$lt:10}},{restaurant\_id:1,name:1,borough:1,cuisine:1,"grades.score":1});

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

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

**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.rest.find({"grades.2.grade":"A","grades.score":9,"grades.date":ISODate("2014-08-11T00:00:00Z")},{restaurant\_id:1,name:1,grades:1});

**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.rest.find({'address.coord.1':{$gt :42,$lte :52}},{restaurant\_id:1,name:1,address:1,"address.coord":1 })

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

db.rest.find().sort({name:1})

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

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

**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.rest.find().sort({cuisine:1,borough:-1})

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

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

**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.rest.find({name: /mon/},{"name":1,"borough":1,"address.coord":1,"cuisine" :1});

**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.rest.find({name: /^Mad/},{"name":1,"borough":1,"address.coord":1,"cuisine" :1});