5. db.restaurant.find({'grades.score':{$gt:80,$lt:100}})

6. db.restaurant.find({'address.coord.0':{$lt:-95.754168}})

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.

cuisine:{$ne:'American'}

'grade.score':{$gt:70}

'address.coord.0':{$lt:-65.754168}

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'.

cuisine:{$nin:['American','Chinees']}

name:/^Wil/

$or:[{cuisine:{$nin:['American','Chinees']}},{name:/^Wil/}]

db.restaurant.find({$or:[{cuisine:{$nin:['American','Chinees']}},{name:/^Wil/}]})

Write a MongoDB query to display the first 5 restaurant

which is in the borough Bronx.

db.restaurant.find({borough:'Bronx'}).limit(5)

display first 5 movies arrange in sorted order of rating

---sort({rating:1}) ---it will arrange data in the ascending order of rating

---sort({rating:-1}) ---it will arrange data in the descending order of rating

---limit(n) ---- to display n documnets

---skip(n)-----to skip first few documents

---- if you use all 3 functions in any sequence, always

----it will et executed in sequence sort, skip, limit

----we cannot define the sequence

db.movie.find().sort({rating:1}).limit(5)

----------------------------------------------------------------------

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.restaurant.find({grades:{$elemMatch:{grade:'A',score:11,

date:ISODate("2014-08-11T00:00:00Z")}}})

db.restaurant.find({'grades.0.grade':'A','grades.0.score':11,

'grades.0.date':ISODate("2014-08-11T00:00:00Z")}}})

---------------------------------------------------------------------

**Types of indexes**

1. single field index --- the index based on single key

db.movie.createIndex({rating:1})

2. compound index --- the index based on more than one key

db.movie.createIndex({rating:1,price:-1})

3. multikey indexes --- the index based on multikey

db.restaurant.createIndex({'adress.zipcode':1})

4. if the indexes are based on gps locations then use geospatial indexes

5. if the size of text is huge, then use text indexes

To create ascending indexes

db.movie.createIndex({rating:1})

To create descending indexes

db.movie.createIndex({rating:-1,price:1})

db.restaurent.createIndex({'address.zipcode':-1,price:1})

To find all indexes on a collection

db.movie.getIndexes();

To remove index from a collection

db.movie.dropIndex(index name);

db.movie.dropIndex('rating\_1)

=======================================================================

**------to update data in mongodb**

**update**: this function is used to to update one or more documents

**updateOne** : this is used to update only one matching document

**updateMany**:this is used to update all matching documnet

updateMany({query},{update action})

updateOne({query},{update action})

update({query},

{update action},

{multi:true,upsert:true,arrayFilters:[]})

**what update operation**

1. add new key-value pair--- **$set, $min, $max**

2. delete existing key------ **$unset**

3. overwrite the value of the key --- **$set,$min, $max**

4. increase or decrease the value of key --- **$inc, $mul**

5. to assign current date ---**$currentDate**

6. to rename the existing key ---**$rename**

If there is a array of values

1. add a new value array **$push---> $each, $position**

2. delete a value from array **$pop,$pull**

3. overwrrite the value in the array ---- **$,$[],$[identifier]**

-----increase price by 100 for all movies

db.movie.updateMany({},{$inc:{price:100}})

-----decrease price by 100 for all movies

db.movie.updateMany({},{$inc:{price:-100}})

---increase price by 2%

price=1\*price+price\*0.02

=price(1+0.02)

=price\*1.02

db.movie.updateMany({},{$mul:{price:1.02}})

---decrease price by 2%

price=1\*price-price\*0.02

=price(1-0.02)

=price\*0.98

db.movie.updateMany({},{$mul:{price:0.98}})

----overwrite value of price by 250

---**$ set** will overwrite the value of price if key exists

---otherwise it will add a new key value pair

db.movie.updateMany({},{$set:{price:250}})

db.movie.updateMany({price:{$exists:true}},{$set:{price:250}})

----overwrite value of price by 300 if the current value < 300

price :400 don't overwrite

price :250 then overwrite 300

db.movie.updateMany({name:/^k/i},{$max:{price:300}})

----overwrite value of price by 300 if the current value>300

price :400 then overwrite 300

price :250 don't overwrite

db.movie.updateMany({name:/^k/i},{$min:{price:300},$set:{}})

----increment rating by one and overwirte price by 300 for all movies with name starts with p

db.movie.updateMany({name:/^p/i},

{$inc:{rating:1},$set:{price:300}})

----increment rating by one and overwirte price by 300 for all movies with name starts with p, assign current date to lastmodified key

db.movie.updateMany({name:/^p/i},

{$inc:{rating:1},$set:{price:300},

$currentDate:{lastmodified:true}})

---to delete a key: value pair use $unset

----to delete price1 key from all movies

db.movie.updateMany({},{$unset:{price1:""}})

----change the name of key lastmodified to lastchange

db.movie.updateMany({name:/^p/},

{$rename:{lastmodified:"lastchange"}})

----add a actor xxx in movie padmavat

db.movie.updateMany({name:'padmavat'},

{$push:{actor:'xxx'}})

----add a actor yyy,zzz in movie padmavat

db.movie.updateMany({name:'padmavat'},

{$push:{actor:{$each:['yyy','zzz']}}})

----add a actor yyy,zzz in movie padmavat at 1st position

db.movie.updateMany({name:'padmavat'},

{$push:{actor:{$each:['yyy','zzz'],$position:1}}})

----add a actor sss in movie padmavat at 0 th position

db.movie.updateMany({name:'padmavat'},

{$push:{actor:{$each:['sss'],$position:0}}})

---to **delete** the value from actor array from the end

---**pop** will delete one value from the **end** if you use **1**

---if you use **-1** it will delete from the begining

db.movie.updateMany({name:'padmavat'},

{$pop:{actor:1}})

---to delete the value from actor array from the begining

db.movie.updateMany({name:'padmavat'},

{$pop:{actor:-1}})

---to delete the all occurences of the value from array use $pull

-----to delete zzz from movie padmavat

db.movie.updateOne({name:'padmavat'},

{$pull:{actor:'zzz'}})

-----to delete yyy and aaaaa from movie padmavat

db.movie.updateOne({name:'padmavat'},

{$pull:{actor:{$in:['yyy','aaaaa']}}})

------create student collection using following data

[

{

\_id: ObjectId("660e6c4da348c48876e3194b"),

no: 1,

name: 'Revati',

marks: [ 103, 110, 107 ]

},

{

\_id: ObjectId("660e6c65a348c48876e3194c"),

no: 2,

name: 'Rajan',

marks: [ 83, 110, 102 ]

},

{

\_id: ObjectId("660e6c87a348c48876e3194d"),

no: 3,

name: 'Isha',

marks: [ 78, 77, 82 ]

}

]

----to overwrite existing values of array we need to use either

**$[] --- to update all values**

**$---- to update first matching value**

**$[identifier] ----to update all matching values**

-----increase all marks of Revati and Rajan by 5

db.student.updateMany({name:/^R/}, {$inc:{'marks.$[]':5}})

----increase marks > 100 for Revati and Rajan

db.student.updateMany({name:/^R/,marks:{$gt:100}}, {$inc:{'marks.$':5}})

----increase all marks > 95 for all the students

db.student.updateMany({},

{$inc:{'marks.$[elem]':5}},

{arrayFilters:[{elem:{$gt:95}]})

**----create following collection**

[

{

\_id: 1,

grades: [

{ grade: 84, mean: 75, std: 6 },

{ grade: 85, mean: 100, std: 4 },

{ grade: 87, mean: 100, std: 6 }

]

},

{

\_id: 2,

grades: [

{ grade: 94, mean: 100, std: 6 },

{ grade: 87, mean: 100, std: 3 },

{ grade: 85, mean: 100, std: 4 }

]

}

]

----update all grade by 2 and std by 1, in students3 collection for all students

db.students3.updateMany({},

{$inc:{'grades.$[].grade':2,'grades.$[].std':1}})

----update first grade by 2 , in students3 collection for all students, if they are in std 6

db.students3.updateMany({'grades.std':6}, {$inc:{'grades.$.grade':2}})

----update all grade by 2 , in students3 collection for all students, if they are in std 6

db.students3.updateMany({},

{$inc:{'grades.$[elem].grade':2}},

{arrayFilters:[{'elem.std':{$eq:6}}]}

)

-----upsert ---- update if exists or insert if doesnot exists

db.movie.updateMany({name:'Kashmir files'},

{$inc:{price:200},$set:{rating:3,ticket\_no:300}},

{upsert:true})

All Employee’s with the desg as ‘CLERK’ are now called as (AO) Administrative Officers.

Update the Employee collection for this.

db.employee.updateMany({Desg:'CLERK'},{$set:{Desg:'AO'}})

Change designation of all employees to senior programmer if they are working on

name:”Project-1” for 4 hrs

db.employee.updateMany(

{project:{$elemMatch:{name:'project-1',Hrs:4}},

{$set:{Desg:'Senior programmer'}})

Decrease number of hrs by 2 for all employees who are working on project-2

db.employee.updateMany(

{'project.name':'project-2'},

{$inc:{'project.$.Hrs':-2}}

)