Group B(2) :

Create Institute Database and Create Student collection with following keys

1. Student Id

2. Student Name

3. Branch

4. Address:{Area,City ,Pincode}

5. Subjects :[ max 5 subjects

{ subject name1:

score1:

}

{ subject name1:

score2:

} ]

6. Area of Interest : [“ DBMS”,”Networking”.....]

\* Enter Subject Names as : 1. DBMSA 2. TOC 3. DC & WSN 4. OSD 5. FC&A

1. Create database Institute.
2. Create collection Students.
3. Insert 10 document with above mentioned structure.
4. Display all students information.
5. Update student branch from IT to Computer of studentid 3.
6. Add interest Python in studentid 5.
7. Add one subject name and its score for Student Id 8.
8. Change City name from Pune to Delhi.
9. Remove record with student id 3.
10. Drop collection.

Group B(17) : For Institute Database ,execute following queries

1. Apply Index on student id in ascending order
2. Apply unique index on studentid.
3. Display students staying in Pune city.
4. Display students staying in Pune or Mumbai City.
5. Display students with area of interest Python and MongoDB.
6. Display students with branch IT and area of interest.
7. Display students scored more than 60 marks in DBMSA.
8. Display students scored more than 60 marks in All subjects.
9. Display students having score greater than 50 and less than 70 in TOC.
10. Display score of students for FC&A in ascending order.
11. Find out name of student starting with 'A'.
12. Find out name of student consists of 'Kumar'.
13. Display Address of all students with studentid.

**Group B Assignments Problem statements**

2 Below given example shows the document structure of a blog site and solve following queries.

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2011,1,20,2,15),

like: 0

},

{

user:'user2',

message: 'My second comments',

dateCreated: new Date(2011,1,25,7,45),

like: 5

}

]

}

1. Create database BlogSite.
2. Create collection post.
3. Insert 10 documents with above mentioned structure.
4. Display all posts information.
5. Update different records by satisfying following condition:
   * + 1. Increment value of likes in one record($inc)
       2. Push some more tags ($push)
       3. Pop some comments($pop)
       4. Replace existing record with new record(update)
       5. set the new title 'New Mong oDB T utorial' of the documents whose title is 'Mong oDB Overview'.($set)
       6. Add one record if doesn’t exist else update($upsert)
6. Remove all the documents whose title is 'Mong oDB Overview'
7. Use $gte,$gt,$lt operators and fire queries.
8. Find records having tags ‘mongodb’.
9. Find records having more than 3 likes.
10. Drop collection.
11. Find all records starting with title ‘M’.
12. Find all records having Mongo in their title.
13. Find total count of records in your collection.
14. All the documents where the number of tags is four.($size)
15. Display the title of the document while querying the document.
16. Display the documents sorted by title in descending order.
17. Show all the tutorials written by 'tutorials point' and whose title is 'Mong oDB Overview'.
18. show all the tutorials written by 'tutorials point' or whose title is 'Mong oDB Overview'
19. Display only 2 documents while querying the document.
20. Locate all the documents that contain the tags “mongodb”,”nosql”.($all)

15 Aggregation and indexing with suitable example using MongoDB.

* 1. Display a list that how many tutorials are written by each user
  2. Compute the top five urls .($sum,$sort)
  3. Return url having likes above 1000.($match,$sum)
  4. Return all 'comments' from user 'sam' .($unwind)
  5. Find total likes for “'http: //www.tutorialspoint.com'” only.
  6. Get all comments created after '2012-11-22'.
  7. Return the Five Most Common “tags”.
  8. Get the total count of the number of comments among all users.

17 Indexing and querying with MongoDB using suitable example.

Import zip.json.

A description...

Solve following queries

1. Display all states having population above 1600.
2. Display all cities having population above 1600.
3. Display all cities in state “KS”.
4. Display location of city "TIMKEN"
5. Uses explain command with all above queries before and after creating index.

18 Connectivity with MongoDB using any Java application.(Insert,update,remove and find)

16 Map reduce operation with suitable example using MongoDB.

The following example assumes we have an events collection with objects of the form:

{

time : <time>,

user\_id : <userid>,

type : <type>, ...

}

use MapReduce to extract all users who have had at least one event of type "sale":

A description...

### Get the count of Males and Females

var mapper = function () {

    emit(this.gender, 1);

};

var reducer = function(gender, count){

    return Array.sum(count);

};

apper = function () {

    emit(this.gender, 1);

};

reducer = function(gender, count){

    return Array.sum(count);

};

db.sourceData.mapReduce(

    mapper,

    reducer,

    {

        out : "example1\_results"

    }

);

db.example1\_results.find()

<http://thejackalofjavascript.com/mapreduce-in-mongodb/>

### Get the Eldest and Youngest Person in each gender

### Example 3 : Count the number of users in each hobby

var mapper = function () {

     var hobbys = this.hobbies.split(',');

      for (i in hobbys) {

        emit(hobbys[i], 1);

    }

};

var reducer = function (key, values) {

    var count = 0;

    for (index in values) {

        count += values[index];

    }

    return count;

};

db.sourceData.mapReduce(

    mapper,

    reducer,

    {

        out : "example3\_results"

    }

);

> m = function() { emit(this.user\_id, 1); }

> r = function(k,vals) { return 1; }

> res = db.events.mapReduce(m, r, { query : {type:'sale'}, out : 'example1' });

> db[res.result].find().limit(2)

{ "\_id" : 8321073716060 , "value" : 1 }

{ "\_id" : 7921232311289 , "value" : 1 }

If we also wanted to output the **number of times the user had experienced the event** in question, we could modify the reduce function like so:

> r = function(k,vals) {

... var sum=0;

... for(var i in vals) sum += vals[i];

... return sum;

... }

db.blogs.aggregate( [

{ $project : { 'comments' : 1 } },

{ $unwind: "$comments" },

{

$match: {

'comments.create' : {

$gt: ISODate("2012-12-21T00:00:00Z")

}

}

}

])

Return all 'comments' from user 'sam' :

db.blogs.aggregate( [

{ $unwind: "$comments" },

{ $match: { 'comments.author' : "sam" } },

{ $group: { \_id: "$comments" } }

] )

db.users.aggregate(

[

{ $unwind : "$likes" },

{ $group : { \_id : null, number : { $sum : 1 } } }

]

);