**Installation step:**

Download mongo db from following link : <https://www.mongodb.org/dl/win32/i386>

Install in your computer

Put all files under C:\Mangodb

Run command prompt and run below command

C:\MongoDB\bin>dir

C:\MongoDB\bin>mongod.exe

It will give error

Now create two folder in C: drive

C:\MongoDB\bin>cd\

C:\>md data

C:\>cd data

C:\data>md db

C:\data>cd \

C:\>cd MongoDB

C:\MongoDB>cd bin

C:\MongoDB\bin>mongod.exe

It will give storage error

C:\MongoDB\bin>mongod.exe --storageEngine=mmapv1

Open second command prompt

C:\Users\WEBSERVER>cd\

C:\>cd MongoDB

C:\MongoDB>cd bin

C:\MongoDB\bin>mongo.exe

> db;

Test/local

> use Ankit <- student name;

switched to db Ankit

> db.dept.insert({"name":"CE/IT"});

> Show dbs;

(now, your database is created)

Note: Don’t close the command promote

Download robomongo Editor for mongodb.

Link: https://robomongo.org/download (download portable version)

Click on connect: ( if any database on left hand side then not create connection again )

Click on create

Give the name of connection and click ok

**Practical 1 : Create database and collection.**

Table name: students

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Student name | Grade | Hobbies |
| 1 | Michelle Jacintha | VII | Internet Surfing |
| 2 | Mabel Mathews | VII | Baseball |

Table name: mycol

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Title | Description | By | url | Tages | Likes |
| ObjectId(7df78ad8902c) | 'MongoDB Overview | MongoDB is no sql database | CGPIT | www.cgpit-bardoli.edu | mongodb, database, NoSQL | 100 |

Table Name: post

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Title | Description | By | url | Tages | Likes | Comment |
| ObjectId  (7df78ad8902c) | 'MongoDB Overview | MongoDB is no sql database | CGPIT | www.cgpit-bardoli.edu | mongodb, database, NoSQL | 100 |  |
|  | NoSQL Databas | NoSQL database does not have tables | UTU | www.utu.ac.in | mongodb, database, NoSQL | 20 | |  |  |  |  | | --- | --- | --- | --- | | user | message | Date Created | Like | | User1 | New Pic | 1/7/2017 | 0 | |

Table: Company

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Name | title | Birth | Death | Contribs | award |
| 1 | first : John  last" : Backus |  | ISODate("1924-12-03T05:00:00Z"), | ISODate("2007-03-17T04:00:00Z"), | Fortran,  ALGOL,  Backus-Naur Form, FP | award : W.W. McDowell Award,  year : 1967,  by : IEEE Computer Society  award : National Medal of Science,  year : 1975,  by : National Science Foundation  award : Turing Award,  year : 1977,  by : ACM  award : Draper Prize,  year : 1993,  by : National Academy of Engineering |
| ObjectId("51df07b094c6acd67e492f41"), | first : John,  last McCarthy |  | ISODate("1927-09-04T04:00:00Z") | ISODate("2011-12-24T05:00:00Z") | Lisp,  Artificial Intelligence,  ALGOL | award : Turing Award,  year : 1971,  by : ACM  award : Kyoto Prize,  year : 1988,  by : Inamori Foundation  award : National Medal of Science,  year : 1990,  by : National Science Foundation |
| 3 | first: Grace,  last : Hopper | Rear Admiral | ISODate("1906-12-09T05:00:00Z") | ISODate("1992-01-01T05:00:00Z"), | UNIVAC,  compiler,  FLOW-MATIC,  COBOL | award" : Computer Sciences Man of the Year,  year : 1969,  by : Data Processing Management Association  award : Distinguished Fellow,  year : 1973,  by" : British Computer Society  award" : W. W. McDowell Award,  year" : 1976,  by : IEEE Computer Society  award : National Medal of Technology,  year : 1991,  by : United States |
| 4 | first : Kristen,  last : Nygaard |  | ISODate("1926-08-27T04:00:00Z") | ISODate("2002-08-10T04:00:00Z") | OOP,  Simula | award : Rosing Prize,  year : 1999,  by : Norwegian Data Association  award : Turing Award,  year : 2001,  by : ACM  award : IEEE John von Neumann Medal,  year : 2001,  by : IEEE |
| 5 | first : Ole-Johan,  last : Dahl |  | ISODate("1931-10-12T04:00:00Z") | ISODate("2002-06-29T04:00:00Z") | OOP,  Simula | award : Rosing Prize,  year : 1999,  by" : Norwegian Data Association  award : Turing Award,  year : 2001,  by : ACM  award : "IEEE John von Neumann Medal,  year : 2001,  by : "IEEE |
| 6 | first: James,  last : Gosling |  | ISODate("1955-05-19T04:00:00Z") |  | java | award : The Economist Innovation Award,  year : 2002,  by : The Economist  award : Officer of the Order of Canada,  year : 2007,  by : Canada |
| 7 | first : Yukihiro,  middle : Matz,  last : Matsumoto |  | ISODate("1965-04-14T04:00:00Z"), |  | Ruby | award" : Award for the Advancement of Free Software,  year : 2011,  by : Free Software Foundation |
| 8 | first : Martin,  last : Odersky |  |  |  | Scala |  |

1. Display all collection.
2. Remove one of collection from database.

**Practical 2**

1. Find the students id 3 and whose name is Aryan david and whose grade is VII. Update his hobbies with skating.
2. Try upsert option in update query.
3. Try save method behalf of update. Find the difference between save method and update method.
4. Save your record in students with id 4 and student name: vasmi bapat and grade VI
5. Save above data with extra attribute hobbies with cricket.
6. Add location with value Bangalore where student’s id is 4.
7. Set location with value Mumbai where student’s grade is VII. (update only single record)
8. How to alter all record of above query.
9. Remove the id 4 from students.
10. Remove the id 4 and student name is vasi bapat and grade is VI
11. Unset the value of location Bangalore where students is 4

Practical 3:

To search for documents from the “Students” collection based on certain search criteria.

To display only the StudName and Grade from all the documents of the Students collection. The identifier \_id should be suppressed and NOT displayed.

To find those documents where the Grade is set to ‘VII’

To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

To find documents from the Students collection where the StudName begins with “M”.

To find documents from the Students collection where the StudName has an “e” in any position.

To find documents from the Students collection where the StudName ends in “a”.

To find documents from the Students collection where id is 3 or 4.

To find documents from the Students collection where id is 3 and 4.

To find documents from the Students collection where location is null.

To find distant name in student collection

Practical 4:

Insert the following documents into a movies collection.

title : Fight Club

writer : Chuck Palahniuk

year : 1999

actors : [

Brad Pitt

Edward Norton

]

title : Pulp Fiction

writer : Quentin Tarantino

year : 1994

actors : [

John Travolta

Uma Thurman

]

title : Inglorious Basterds

writer : Quentin Tarantino

year : 2009

actors : [

Brad Pitt

Diane Kruger

Eli Roth

]

title : The Hobbit: An Unexpected Journey

writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

title : The Hobbit: The Desolation of Smaug

writer : J.R.R. Tolkein

year : 2013

franchise : The Hobbit

title : The Hobbit: The Battle of the Five Armies

writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness.

Query the movies collection to

1. get all documents
2. get all documents with writer set to "Quentin Tarantino"
3. get all documents where actors include "Brad Pitt"
4. get all documents with franchise set to "The Hobbit"
5. get all movies released in the 90s
6. get all movies released before the year 2000 or after 2010

**Update Documents**

1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."
2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."
3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"

**Text Search**

1. find all movies that have a synopsis that contains the word "Bilbo"
2. find all movies that have a synopsis that contains the word "Gandalf"
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"
4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"
5. find all movies that have a synopsis that contains the word "gold" and "dragon"

**Delete Documents**

1. delete the movie "Pee Wee Herman's Big Adventure"
2. delete the movie "Avatar"

### **Insert the following documents into a posts collection**

username : GoodGuyGreg

title : Passes out at party

body : Wakes up early and cleans house

username : GoodGuyGreg

title : Steals your identity

body : Raises your credit score

username : GoodGuyGreg

title : Reports a bug in your code

body : Sends you a Pull Request

username : ScumbagSteve

title : Borrows something

body : Sells it

username : ScumbagSteve

title : Borrows everything

body : The end

username : ScumbagSteve

title : Forks your repo on github

body : Sets to private

### Insert the following documents into a comments collection

username : GoodGuyGreg

comment : Hope you got a good deal!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"

username : GoodGuyGreg

comment : What's mine is yours!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"

username : GoodGuyGreg

comment : Don't violate the licensing agreement!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github

username : ScumbagSteve

comment : It still isn't clean

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"

username : ScumbagSteve

comment : Denied your PR cause I found a hack

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"

**Querying related collections**

1. find all users
2. find all posts
3. find all posts that was authored by "GoodGuyGreg"
4. find all posts that was authored by "ScumbagSteve"
5. find all comments
6. find all comments that was authored by "GoodGuyGreg"
7. find all comments that was authored by "ScumbagSteve"
8. find all comments belonging to the post "Reports a bug in your code"

Practical 5:

First filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal” and then filter those documents wherein the “TotAccBal” is greater than 1200, use the below syntax:

// Insert customers documents

db.Customers.insert([{CustID:"C123",AccBal:500,AccType:"S"},

{CustID:"C123",AccBal:900,AccType:"S"},

{CustID:"C11",AccBal:1200,AccType:"S"},

{CustID:"C123",AccBal:1500,AccType:"C"}]);

First filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal”

First filter on “AccType:S” and then group it on “CustID” and then compute the sum of “AccBal” and then filter those documents wherein the “TotAccBal” is greater than 1200, use the below syntax:

Aggregate using Map reduce

To find the number of documents in the Students collection.

Consider the following document structure under **posts** collection containing the post text and its tags −

{

"post\_text": "enjoy the mongodb articles on website",

"tags": [

"mongodb",

"tutorial"

]

}

Perform following query form above table.

1. Use of regex Expression

2. Use of regex Expression with Case Insensitive

3. Use of regex for Array Elements

To sort the documents from the Students collection

Practical 7 :

Create a collection by the name “food” and then insert documents into the foold 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’.

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

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

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

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

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

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

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

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

Update document with \_id:3 by popping an element

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

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

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

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.

2) Create a function to display addition of two number;

3) Create a function to display factorial number using recursion.

4) Create a function to display an Armstrong number.

Pratical 9:

**Structure of 'restaurants' collection :**

{

"address": {

"building": "1007",

"coord": [ -73.856077, 40.848447 ],

"street": "Morris Park Ave",

"zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [

{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },

{ "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },

{ "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },

{ "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },

{ "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }

],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

}

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

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

**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.

**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. 

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

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

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

**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 100.

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

**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.

**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.

**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.

**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.

**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.

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

**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.

**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.

**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.

**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'.

**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..

**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".

**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.

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

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

**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. 

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

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

**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.

**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.

**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.