**MONGO DB:**

**Mongo database is a no sql database.it is an open source.it is mainly used for big data applications.**

**Here, after installing the software there will be mongo and mongod where,**

**Mongod->server**

**Mongo->client**

**Mongo data base stores data inform of objects**

**Mongo db consists of 2 objects**

**1)JSON**

**2)BSON(binary JSON)**

**In Mongo db, the data is stored in form of collections .Inside collections there will be documents and documents consists of objects/fields.**

**COMMANDS:**

**To create /use a database:**

**Use databasename**

**Eg: use thub(here the data base will be created and automatically we will be switched into that data base)**

**To see the databases :**

**Show dbs( as along as collections are not there in data base our database does not exist and cant be seen in our databases)**

**Create collection:**

**db.createCollection(“collection name”)**

**Eg:db.createCollection(“students”)**

**Output:**

**{“ok”:1}**

**To see collections in our database:**

**show collections**

**eg:**

**show collections**

**output:**

**students**

**CRUD OPERATIONS:**

**Letter in mongodb injavascript**

**C -create insert post**

**R- read find get**

**U-update update(set) put**

**D-delete remove delete**

* **All CRUD operations of javascript are also called as Restful API’s**

**In mongo db the data base and collections are permenant so we will create them where as documents and objects are temporary so we will insert them in the collections.**

**To insert document:**

**db.collectionname.insert({objectname1:objectvalue1,objectname2:object value 2……})**

**Eg:db.students.insert({“name”:”sweety”,”college”:”ACOE”,”rno”:001,”branch”:”cse”})**

**Here, name, college,rno branch are called json objects**

* **We can directly declare our collection name in the insertion of document itself**
* **Creating a collection with its syntax is optional(db.createCollection(“collection name”))**

**To find the document:**

**db.student.find({})**

**toappearobjects in an consistent order we will use:**

**db.student.find({}).pretty()**

**To find a particular document:**

**db.student.find({“write any one object name and object value”}).pretty()**

**\*this will display the particular document which consists of object name and object value**

**Eg:db.students.find({“name”:”sweety”}).pretty()**

**Output:**

**\_id:**

**Name:”sweety”**

**College:”ACOE”**

**Rollno:001**

**Branch:”cse”**

**Updating the document:**

**Update can be done in 2 ways**

* **Adding new object**
* **Replacing new object with already existing objects**

**\*updates consists of 2 functions**

**Update({},{}) ->the 1st function is used to find if the document is exixting in the collection or not(match)**

**->the 2nd function is to modify the document with new objects (modify)**

**If the 1st function is satisfied then only it will execute the 2nd function.**

**SYNTAX:**

**db.collectionname.update({objectname},{$ set:{“new object”}})**

**Eg:**

**db.students.update({name:”sweety”},{$set:{gender:”F”}})**

**(or)**

**db.students.update({name:”sweety”},{$set:{name:”snoopy”}})**

**Output:**

**1)db.students.update({name:”sweety”},{$set:{gender:”F”}})**

**\_id:**

**Name:”sweety”**

**College:”ACOE”**

**Rollno:001**

**Branch:”cse”**

**Gender:”f”**

**2)db.students.update({name:”sweety”},{$set:{name:”snoopy”}})**

**\_id:**

**Name:”snoopy”**

**College:”ACOE”**

**Rollno:001**

**Branch:”cse”**

**Gender:”f”**

Remove the document:

db.collectionname.remove({“any object name in the document”})

Eg:

db.students.remove({rno:001})

To delete collection:

db.collectionname.drop()

Eg:db.students.drop()

To delete database:

db.dropDatabase()

To count number of documents in collection:

db.collectionname.find({}).count()

PROJECTIONS:

1)sort

2)skip

3)limit

Sort():

db.collectionname.find({}).sort({objectname:1})

* We will use name:1(to sort in ascending order)
* We will use name:-1(to sort in descending order)

Eg:it will sort the data using names if 1 is given it will sort in ascending order and -1 for descending order

skip():

db.students.find({}).skip(no of values to skip)

limit():

db.collectionname.find({}).limit(no of values to be there)

Eg:

\*db.collectionname({}).sort({name:1})

\*db.collectioname({}).skip(2)

\*db.collectionname({}).limit(3)

Indexing:(only receiving the necessary details without id and other)

db.collectioname.find({},{name:1,id:0})

it displays only name in the document

db.collectioname.find({},{name:0,id:0})

it displays all other objects except name and id

Eg:

1)db.students.find({},{name:1,id:0})

name:”snoopy”

2)db.students.find({},{name:0,id:0})

**College:”ACOE”**

**Rollno:001**

**Branch:”cse”**

**Gender:”f”**

**To remove one particular object from all documents:**

**db.collectioname.update({},{$unset:{college:1},{multi:true})**

**Eg:db.students.update({},{$unset:{college:1},{multi:true})**

**Output:**

**Name:”snoopy”**

**Rollno:001**

**Branch:”cse”**

**Gender:”f”**

**AGGREGATIONS:**

**These are used to create different groups depend upon the specifications**

**SYNTAX:**

**Sum:**

**db.collectioname.aggregate([{$group:{“\_id”:”$gender”,total:{$sum:1}}}])**

**Average:**

**db.collectioname.aggregate([{$group:{“\_id”:”$age”,total:{$avg:”$age”}}}])**

**Min and Max terms:**

* **db.collectioname.aggregate([{$group:{“\_id”:”$gender”,total:{$min:”$age”}}}])**
* **db.collectioname.aggregate([{$group:{“\_id”:”$gender”,total:{$max:”$age”}}}])**

**Studio3T:**

**\*Studio 3t is graphical user interface of mongodb.it is used to write the commands in this studio 3t instead of linux environment we can work in studio 3t .**

**\*It also displays the documents in json ,tabular and tree format**

**\*In linux environment it displays only in json format**

**\*Here there is no need of any commands to work with ,there will be avalibility of many icons to work with as it is a graphical user interface**