res.sql

```
-- Establish a  connection with  the server :
mongosh string

-- show all databases:
show dbs

-- create a new database :
use databaseName

-- create a new collection :
db.createCollection(collectionName)

-- drop a database :
db.dropDatabase()  -- Drop the current used database :

--  insert a document into a collection :
db.collectionName.insertOne(object)

-- show collection documents :
db.collectionName.find()

-- insert many document in a collection :
db.collectionName.insertMany([object1,object2,...,objectN])

-- display documents of a collection sorted in a order :
db.collectionName.find().sort({filedNameToSortWith:[-1 : (desc) , 1 : (asc) ]})

-- limit the number of documents return  :
db.collectionName.find().limit(nbrDocumentToReturn)

-- using the find function  :
db.collection.find({query},{projection})
db.collection.find({filed1Name:valueToEqual},{filedNName:valueToEqual},
{filedName:apprearnesStat(false | true)} )

/*
-query : specfiy a condition at the document filed
-projection : select the filed thaat should be apprear [ the _id by default apprear you
should make it false ]
*/

-- example :  -- []
    -- sql query :
    select name from students
    where name = 'Ayoub'

    -- transformation to mongosh :
    db.students.find({name:'ayoub'},{name:true})

--  update a document :
db.collectionName.updateOne(filtre,update)
/*
$set : set or define a  new filed  value
$unset : remmove a filed
```

```
53
54  */
55  -- example :
56      -- update the document with the name ='ayoub' to be updatedName
57      db.students.updateOne({name:'ayoub'},{$set:{name:'updatedName'}})
58
59      db.students.updateOne(
60
61          {name:'ayoub',courses:[]},
62          {$set:{name:'updatedName'}}
63
64          )
65      -- remove the username filed of the document with the username = 'kk'
66      db.students.updateOne({username:'kk'},{$unset:{username:'d'}})
67
68  -- update multiple documents :
69  db.collectionName.updateMany(filtre,update)
70  -- examples :
71
72  -- example 1  : update all documents that do not have a name
73  db.students.updateMany({name:{$exists:false} },{$set:{name:'defaultName'}})
74
75  -- delete a document  :
76  db.collectionName.deleteOne({filtre})
77
78  --  example :
79  db.students.deleteOne({username:"kk"})
80
81  -- delete multiple documents :
82   db.collectionName.deleteMany({filtre})
83
84  -- example :
85  db.students.deleteMany({username:"kk"})
86
87  -- comparaison operators :
88  /*
89      $ne : not equal
90      $lt : less than
91      $gt : grater than
92      $lte : less than or equal
93      $gte : grater than or equal
94      $in : in range of values []
95      $nin : not in range of values []
96  */
97  -- examples :
98
99  -- find every lastname except Kamal : $ne
100 db.students.find({lastname:{$ne:'Kamal'}})
101
102 -- find every document where age grater than pr equal 20
103 db.students.find({age:{$gte:20}})
104
105 -- Retrieve all documents with names 'Ayoub' or 'Amina'
106 db.students.find({name:{$in:['Ayoub','Amina']}})
107
108 -- Retrieve all documents not in names 'Ayoub' or 'Amina'
```

```
db.students.find({name:{$nin:['Ayoub','Amina']}})

-- logical operators :
/*
    $and
    $not
    $nor : returns all documens that fail to match both clauses
    $or

*/
--   Retrieve all documents in names 'Ayoub' or 'kamal' and have an age grater than 20
db.students.find({ $and :[ {name: { $in:['Ayoub','kamal'] } },{age:{$gt:20}} ]})

--   Retrieve all documents in names 'Ayoub' or 'kamal' or have an age grater than 20
db.students.find({ $or :[ {name: { $in:['Ayoub','kamal'] } },{age:{$gt:20}} ]})

-- Retrieve all documents   names 'Ayoub' or 'kamal' and    have an age grater than 20
-- :(not include null ages)
 db.students.find({$and :[ {name:{$in :['Ayoub','kamal']}} ,{age :{$gt:20}} ]})

-- Retrieve all documents not in  names 'Ayoub' or 'kamal' and not  have an age grater than
20
-- :(include null ages)
db.students.find({$and :[ {name:{$in :['Ayoub','kamal']}} ,{age :{$not:{$lte:20}}} ]})


-- indexes :
/*
    Indexes support the efficient execution of queries in MongoDB. Without indexes,
    MongoDB mustperform a collection scan, i.e. scan every document in a collection,
    to select those documents that
    match the query statement. If an appropriate index exists for a query,
    MongoDB can use the index to limit the number of documents it must
    inspect.
*/

-- get the executionStat of a query :
db.collectionName.query().explain('executionStats')

-- add an index at a specific filed :
db.collectionName.createIndex({filedName:[1 | -1]})
/*
     1 : range filed values in ascending order
    -1 : range filed values in descending order
*/


-- examples :
    --  Add index to the 'name' field
    db.students.createIndex({ name: 1 })

    --  Add index to the 'age' field
    db.students.createIndex({ age: 1 })

    --  Add compound index to multiple fields
    db.students.createIndex({ name: 1, age: -1 })
```

```
164
165
166 -- get all indexes in your collection :
167 db.collectionName.getIndexes()
168
169 -- drop Indexes :
170
171     -- Drop all indexes except the default '_id' index
172     db.collection.dropIndexes()
173
174     -- Drop index by specifying the fields
175     db.collection.dropIndex({ field1: 1, field2: -1 })
176
177     -- Drop index by name
178     db.collection.dropIndex("index_name")
179
180 -- get all collection in your system
181 show collectios
182
183 -- create a collection :
184 db.createCollection('collectionName',{capped:true,size:valueInKb,max:maxNbrOfDocuments,
    autoIndexId: 1 | 0 })
185
186 /*
187 capped :
188     To create a capped collection, specify true. If you specify true,
189     you must also set a maximum size in the size option.
190 max :
191     The maximum number of documents allowed in the capped collection.
192     The size option takes precedence over this limit. If a capped collection
193     reaches the size limit before it reaches the maximum number of documents,
194     MongoDB removes old documents. If you prefer to use the max limit,
195     ensure that the size limit, which is required for a capped collection,
196     is sufficient to contain the maximum number of documents.
197
198 size :
199     Specify a maximum size in bytes for a capped collection. Once a capped
200     collection reaches its maximum size, MongoDB removes the older documents
201     to make space for the new documents. The size option is required for capped
202     collections and ignored for other collections.
203
204 autoIndexId:
205 Specify false to disable the automatic creation of an index on the _id field.
206 */
207
208 -- drop collection :
209 db.collectionName.drop()
210
```