EJEMPLOS extraídos del Libro:
Sams Teach Yourself NoSQL with MongoDB in 24 Hours de Brad Dayley

--------- consultas simples

```
> coll.findOne ({zona: "Bronx" }));  // da solo el primer documento que cumple la condición
> sale= coll.find ( {zona: "Bronx" }); // ejemplo sobre restaurantes
```


```
{ "_id" : ObjectId("555cbf1cf3bef5d39a9d3f8b"), "direccion" : { "edificio" : "10
07", "coordenadas" : [ -73.856077, 40.848447 ], "calle" : "Morris Park Ave", "di
strito" : "10462" }, "zona" : "Bronx", "tipococina" : "Bakery", "puntuaciones" :
 [ { "fecha" : { "date" : 1393804800000 }, "nivel" : "A", " valor" : 2 }, { "fec
ha" : { "date" : 1378857600000 }, "nivel" : "A", " valor" : 6 }, { "fecha" : { "
date" : 1358985600000 }, "nivel" : "A", " valor" : 10 }, { "fecha" : { "date" :
1322006400000 }, "nivel" : "A", " valor" : 9 }, { "fecha" : { "date" : 129971520
0000 }, "nivel" : "B", " valor" : 14 } ], "nombreRes" : "Morris Park Bake Shop",
 "restaurante_id" : "30075445" }

{ "_id" : ObjectId("555cbf32f3bef5d39a9d3f8c"), "direccion" : { "edificio" : "10
07", "coordenadas" : [ -73.856077, 40.848447 ], "calle" : "Morris Park Ave", "di
strito" : "10462" }, "zona" : "Bronx", "tipococina" : "Bakery", "puntuaciones" :
 [ { "fecha" : { "date" : 1393804800000 }, "nivel" : "A", " valor" : 2 }, { "fec
ha" : { "date" : 1378857600000 }, "nivel" : "A", " valor" : 6 }, { "fecha" : { "
date" : 1358985600000 }, "nivel" : "A", " valor" : 10 }, { "fecha" : { "date" :
1322006400000 }, "nivel" : "A", " valor" : 9 }, { "fecha" : { "date" : 129971520
0000 }, "nivel" : "B", " valor" : 14 } ], "nombreRes" : "Morris Park Bake Shop",
 "restaurante_id" : "30075445" }
```


```
// comprobar si se ha ejecutado bien
> print(sale);
DBQuery: test.palabras -> { }
> if (sale) {print("fue bien")} ;
fue bien
```

```
> sale= coll.find ( { "tamaño" : 3, "ultima" :e },{}); // los strings deben ir entre comillas :"e"
2015-05-20T20:18:36.067+0200 ReferenceError: e is not defined

> sale= coll.find ( { "tamaño" : 3, "ultima" :"e" },{});
```


```
{ "_id" : { "str" : "52d87454483398c8f2429277" }, "palabra" : "the", "primera" :
 "t", "ultima" : "e", "tamaño" : 3, "letras" : [ "t", "h", "e" ], "estadis" : {
"vocales" : 1, "consonantes" : 2 }, "caractsets" : [ { "tipo" : "consonantes", "
caracts" : [ "t", "h" ] }, { "tipo" : "vocales", "caracts" : [ "e" ] } ] }
```

```
> sale= coll.find ( { "tamaño" : { $gt: 0, $lt: 4 }, "ultima" :'e' },{});
```


```
{ "_id" : { "str" : "52d87454483398c8f2429277" }, "palabra" : "the", "primera" :
 "t", "ultima" : "e", "tamaño" : 3, "letras" : [ "t", "h", "e" ], "estadis" : {
"vocales" : 1, "consonantes" : 2 }, "caractsets" : [ { "tipo" : "consonantes", "
caracts" : [ "t", "h" ] }, { "tipo" : "vocales", "caracts" : [ "e" ] } ] }
```

>

> sale= coll.find ({ "tamaño" : { $gt: 0, $lt: 4 }, "ultima" :'e' },{palabra: 1 }); // solo atrbi palabra
{ "_id" : { "str" : "52d87454483398c8f2429277" }, "palabra" : "the" }
>

--------- Pretty : tabula los resultados para verlos mejor

> db.palabras.find().pretty()


```
{
	"_id" : {
		"str" : "52d87454483398c8f2429277"
	},
	"palabra" : "the",
	"primera" : "t",
	"ultima" : "e",
	"tamaño" : 3,
	"letras" : [
		"t",
		"h",
		"e"
	],
	"estadis" : {
		"vocales" : 1,
		"consonantes" : 2
	},
	"caractsets" : [
		{
			"tipo" : "consonantes",
			"caracts" : [
				"t",
				"h"
			]
		},
		{
			"tipo" : "vocales",
			"caracts" : [
				"e"
			]
		}
	]
}
```

---------- CON OTRAS DBs ----------

db.students.find({ score: { $gt: 0, $lt: 2 } })

Matches the following documents:

{ "_id" : 1, "score" : [-1, 3] }
{ "_id" : 2, "score" : [1, 5] }

'awards' array contains an embedded document element
that contains the 'award' field equal to "Turing Award"
and the 'year' field greater than 1980:

```
db.bios.find(
  {
    awards: {
          $elemMatch: {
             award: "Turing Award",
             year: { $gt: 1980 }
          }
    }
  }
)
```

-- the embedded document name is
exactly { first: "Yukihiro", last: "Matsumoto" },
including the order:

```
db.bios.find(
   {
     name: {
           first: "Yukihiro",
           last: "Matsumoto"
         }
   }
)
```

-

**** ------- usando find_all.js : encuentra todos los documentos (uso de CURSOR)**

---------- con colección palabras

```
      // find devuelve un “cursor” , no el contenido
db.palabras.find().forEach(function(word){
  print("una palabra: " + word.palabra);
});

   // -- da lo mismo haciéndolo por separado
  //      “var” es para ámvito global
var cursor = db.palabras.find();
cursor.forEach(function(word){
  print("una palabra: " + word.palabra);
});
//
// ---------- el resultado es un array
todas = db.palabras.find().map(function(word){
               return word.palabra;
 });
printjson(todas);

  // colocar el resultado del find en un array
var todas = db.palabras.find().toArray();
 // muestra todos los documentos del array [ …]
todas
```

```
01 mongo = new Mongo("localhost");
02 wordsDB = mongo.getDB("words");
03 wordsColl = wordsDB.getCollection("word_stats");
04 print("\nFor Each List: ");
//  --- recorre usando el método forEach del objeto Cursor
05 cursor = wordsColl.find();
06 cursor.forEach(function(word){
07     print("word: " + word.word);
08   });

09 print("\nMapped Array: ");
// ---- la función map crea una array del campo word de todos los
//     documentos
10 cursor = wordsColl.find();
11 words = cursor.map(function(word){
12   return word.word;
13 });

14 printjson(words);

15 print("\nIndexed Document in Array: ");
// ---- transforma el cursor en un array e imprime elemento 55
16 cursor = wordsColl.find();
17 words = cursor.toArray();
18 print(JSON.stringify(words[55]));
```

```
19 print("\nNext Document in Cursor: ");
// ---- obtiene el siguiente elemento en la lista y lo imprime
20 cursor = wordsColl.find();
21 word = cursor.next();
22 print(JSON.stringify(word));
```

** Resultado --------

```
------- 05,06,07 -----
For Each List:
word: the
word: be
word: and
...
word: apology
word: till
---- 09 -> 14 --
Mapped Array:
[
    "the",
    "be",
    "and",
...
    "apology",
    "till"
]
-------- 15 --> 18 --------
Indexed Document in Array:
{"_id":{"str":"52d87454483398c8f24292ae"},"word":"there","first":"t","last":"e",
 "size":5,"letters":["t","h","e","r"],"stats":{"vowels":2,"consonants":3},
 "charsets":[{"type":"consonants","chars":["t","h","r"]},
          {"type":"vowels","chars":["e"]}]}

----------- 19 --> 22 -------
Next Document in Cursor:
{"_id":{"str":"52d87454483398c8f2429277"},"word":"the","first":"t","last":"e",
 "size":3,"letters":["t","h","e"],"stats":{"vowels":1,"consonants":2},
 "charsets":[{"type":"consonants","chars":["t","h"]},{"type":"vowels","chars":[
"e"]}]}
```

-

Encontrar Documentos a partir del valor de un campo

Base de datos "palabras". Cada palabra es un documento con estos campos

{"_id":{"str":"52d87454483398c8f2429277"},"palabra":"the","primera":"t","ultima":"e",
"tama o":3,"letras":["t","h","e"],"estadis":{"vocales":1,"consonantes":2},
"caractsets":[{"tipo":"consonants","caracts":["t","h"]},{"tipo":"vowels","caracts":[
"e"]}]}

- encontrar palabras de tamaño = 5

```
db.palabras.find({tamaño: 5});
```

- encontrar el documento (palabra) "there"

```
find({palabra: "there"});
```

- encontrar documentos (palabras) cuyo campo "primera" sea una de estas tres letras: a, b c.

```
find({primera:{$in: ['a', 'b', 'c']}});
```

- encontrar documentos con campo "tamaño" mayor de 12

```
find({tamano:{$gt: 12}});
```

- encontrar documentos con campo "tamaño" menor de 12

```
find({tamaño:{$lt: 12}});
```

- Encontrar docs con el campo "letras" que es un array tenga más de 10 elementos

```
find({letras:{$size: {$gt: 10}}});
```

- Encontrar docs con el campo "letras", que es un array tenga 14 elementos

```
find({letras:{$size: 14}});
```

- Encontrar docs. en un subdocumento usando ".": que tengan más de 6 vocales en sus estadísticas:

```
find({"estadis.vocales":{$gt:6}});
```

- Encontrar docs. según los contenidos de un campo array: que tengan en su campo "letras" todas los elementos de la consulta

```
find({letras:{$all: ['a','e','i','o','u']}});
```

- Encontrar docs. que tenga algún campo concreto (los campos son todos opcionales): que tengan campo "caracRaros"

```
find({caracRaros: {$exists:true}});
```

- Encontrar docs. que contengan un campo ("caractsets"), que es un array de subdocumentos. Se desea un valor o propiedad de un campo de los subdocumentos.

Cada subdocumento tiene dos campos, "tipo" y "caracts", con esta estructura:
[{"tipo":"xxxx",
"caracts":["1","2"]}

En este caso el array tiene dos elementos que son sets de caracteres:
"caractsets":[{"tipo":"consonants","caracts":["t","h"]},
{"tipo":"vowels", "caracts":["e"]}]

Queremos un set de caracteres que tenga el "tipo" = "otros" y el tamaño del campo "caracts", que es un array sea igual a 2

```
find(
{caractsets:{$elemMatch: {$and: [{tipo: 'other'},{caracts: {$size: 2}}]}}}
);
```

-

**** ---------- usando find_specific.js : encontrar docs específicos**

---------- con colección “palabras”

-- Hace un array con todos los valores de atrib. “palabra”

-- Si el total de la longitud del array > 65 caracteres, saca los primeros 50 y “. . .”

```
function displayWords(msg, cursor){
  print("\n"+msg);
  words = cursor.map(function(word){
    return word.palabra;
    });
  wordStr = JSON.stringify(words);
  if (wordStr.length > 65){
    wordStr = wordStr.slice(0, 50) + "...";
  }
  print(wordStr);
 }

var cur1 = db.palabras.find();
displayWords("alla va 1: ", cur1);
```

```
01 function displayWords(msg, cursor, pretty){
02   print("\n"+msg);
03   words = cursor.map(function(word){
04     return word.word;
05   });
06   wordStr = JSON.stringify(words);
07   if (wordStr.length > 65){
08     wordStr = wordStr.slice(0, 50) + "...";
09   }
10   print(wordStr);
11 }

12 mongo = new Mongo("localhost");
13 wordsDB = mongo.getDB("words");
14 wordsColl = wordsDB.getCollection("word_stats");
15 cursor = wordsColl.find({first: {$in: ['a', 'b', 'c']}});
16 displayWords("Words starting with a, b or c: ", cursor);
17 cursor = wordsColl.find({size:{$gt: 12}});
18 displayWords("Words longer than 12 characters: ", cursor);
19 cursor = wordsColl.find({size:{$mod: [2,0]}});
20 displayWords("Words with even Lengths: ", cursor);
21 cursor = wordsColl.find({letters:{$size: 12}});
22 displayWords("Words with 12 Distinct characters: ", cursor);
23 cursor = wordsColl.find({$and:
24                         [{first:{
25                           $in: ['a', 'e', 'i', 'o', 'o']}},
26                          {last:{
27                           $in: ['a', 'e', 'i', 'o', 'o']}}]});
28 displayWords("Words that start and end with a vowel: ", cursor);
29 cursor = wordsColl.find({"stats.vowels":{$gt: 6}});
```

```
30 displayWords("Words containing 7 or more vowels: ", cursor);
31 cursor = wordsColl.find({letters:{$all: ['a','e','i','o','u']}});
32 displayWords("Words with all 5 vowels: ", cursor);
33 cursor = wordsColl.find({otherChars: {$exists: true}});
34 displayWords("Words with non-alphabet characters: ", cursor);
35 cursor = wordsColl.find({charsets:{
36                     $elemMatch:{
37                       $and:[{type: 'other'},
38                           {chars: {$size: 2}}]}}});
39 displayWords("Words with 2 non-alphabet characters: ", cursor);
```

---------- resultado (traducido al español)

Palabras que empiezan por a, b ó c
["be","and","a","can't","at","but","by","as","can"...

Palabras más largas que 12 car.:
["international","administration","environmental",...

Palabras con longitudes pares:
["be","of","in","to","have","to","it","that","he",...

Palabras con 12 car. distintos:
["uncomfortable","accomplishment","considerably"]

Palabras que empiezan y terminan por vocal:
["a","i","one","into","also","one","area","eye","i...

Palabras que contienen 7 o más vocales:
["identification","questionnaire","organizational"...

Palabras con todas las 5 vocales:
["education","educational","regulation","evaluatio...

Palabras con caracteres que no son letras:
["don't","won't","can't","shouldn't","e-mail","lon...

Palabras con dos
caracteres que no son letras:
["two-third's","middle-class'"]

El comando getLastError o el método getLastError() del objeto Database devuelve el código de error de la última instrucción ejecutada.

Ejemplo con la BD wordsDB:

// el método da mucha más información : el estado, la operación que lo produjo , n mero de docs. modificados, el mensaje de error y otras propiedades que est n en la tabla

```
mongo = new Mongo('localhost');
```

```
wordsDB = mongo.getDB('words');
wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
wordsColl = wordsDB.getCollection('word_stats');
wordsColl.insert({word:"the"});
lastError = wordsDB.getLastError();  // <--- uso del m  todo de la DB
if(lastError){
  print("ERROR: " + lastError);
}
```

-

---

ejemplo ejecutando: mongo doc_add.js

```
01 selfie = {
02   word: 'selfie', first: 's', last: 'e',
03   size: 4, letters: ['s','e','l','f','i'],
04   stats: {vowels: 3, consonants: 3},
05   charsets: [ {type: 'consonants', chars: ['s','l','f']},
06             {type: 'vowels', chars: ['e','i']} ],
07   category: 'New' };
08 tweet = {
09    word: 'tweet', first: 't', last: 't',
10    size: 4, letters: ['t','w','e'],
11    stats: {vowels: 2, consonants: 3},
12    charsets: [ {type: 'consonants', chars: ['t','w']},
13              {type: 'vowels', chars: ['e']} ],
14    category: 'New' };
15 google = {
16    word: 'google', first: 'g', last: 'e',
17    size: 4, letters: ['g','o','l','e'],
18    stats: {vowels: 3, consonants: 3},
19    charsets : [ {type: 'consonants', chars: ['g','l']},
20               {type: 'vowels', chars: ['o','e']} ],
21    category: 'New' };
22
23 mongo = new Mongo('localhost');
24 wordsDB = mongo.getDB('words');
25 wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
26 wordsColl = wordsDB.getCollection('word_stats');
27 print('Before Inserting selfie: ');
28 cursor = wordsColl.find({word: {$in: ['tweet','google', 'selfie']}},
29                     {word:1});
30 printjson(cursor.toArray());
31 wordsColl.insert(selfie);
32 print('After Inserting selfie: ');
33 cursor = wordsColl.find({word: {$in: ['tweet','google', 'selfie']}},
34                     {word:1});
35 printjson(cursor.toArray());
36 print('After Inserring tweet and google');
37 wordsColl.insert([tweet, google]);
38 cursor = wordsColl.find({word: {$in: ['tweet','google', 'selfie']}},
39                     {word:1});
```

```
40 printjson(cursor.toArray());
```

---

ejemplo ejecutando : mongo doc_update.js

```
01 function displayWords(cursor){
02   words = cursor.map(function(word){
03     return word.word + "(" + word.size + ")";
04   });
05   wordStr = JSON.stringify(words);
06   if (wordStr.length > 65){
07     wordStr = wordStr.slice(0, 50) + "...";
08   }
09   print(wordStr);
10 }
11 mongo = new Mongo('localhost');
12 wordsDB = mongo.getDB('words');
13 wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
14 wordsColl = wordsDB.getCollection('word_stats');
15 cursor = wordsColl.find({category:"QYwords"});
16 print("Before QYwords Update: ");
17 displayWords(cursor);
18 wordsColl.update( { $and:[{ first: "q"},{last:'y'}]},
19                 { $set: {category:'QYwords'}},
20                 false, true);
21 cursor = wordsColl.find({category:"QYwords"});
22 print("After QYwords Update: ");
23 displayWords(cursor);
24 print("Before Left Update: ");
25 word = wordsColl.findOne({word: 'left'},
26                       {word:1, size:1, stats:1, letters:1});
27 printjson(word);
28 wordsColl.update({ word: 'left'},
29                 { $set: {word:'lefty'},
30                   $inc: {size: 1, 'stats.consonants': 1},
31                   $push: {letters: "y"}},
32                 false, false);
33 word = wordsColl.findOne({word: 'lefty'},
34                       {word:1, size:1, stats:1, letters:1});
35 print("After Left Update: ");
36 printjson(word);
37 wordsColl.update({category:"QYwords"},
38                 {$set: {category:"none"}}, false, true);
39 wordsColl.update( { word: 'lefty'},
40                 { $set: {word:'left'},
41                   $inc: {size: -1, 'stats.consonants': -1},
42                   $pop: {letters: 1}});
43 word = wordsColl.findOne({word: 'left'},
44     {word:1, size:1, stats:1, letters:1});
45 print("After Lefty Update: ");
46 printjson(word);
```

---------- salida de la ejecución

```
Before QYwords Update:
[]
After QYwords Update:
["quickly(7)","quality(7)","quietly(7)"]
Before Left Update:
{
	"_id" : ObjectId("52e2992e138a073440e4663c"),
	"word" : "left",
	"size" : 4,
	"letters" : [
		"l",
		"e",
		"f",
		"t"
	],
	"stats" : {
		"vowels" : 1,
		"consonants" : 3
	}
}
After Left Update:
{
	"_id" : ObjectId("52e2992e138a073440e4663c"),
	"letters" : [
		"l",
		"e",
		"f",
		"t",
		"y"
	],
	"size" : 5,
	"stats" : {
		"consonants" : 4,
		"vowels" : 1
	},
	"word" : "lefty"
}
After Lefty Update:
{
	"_id" : ObjectId("52e2992e138a073440e4663c"),
	"letters" : [
		"l",
		"e",
		"f",
		"t"
	],
	"size" : 4,
	"stats" : {
		"consonants" : 3,
		"vowels" : 1
	},
	"word" : "left"
```

}

-

**** salvando -------- Saving docs in a collection (no usar)**

- Salvar doc que has creado nuevo
  → *No es tan eficiente como insert() o update()*

```
existingObject = myCollection.findOne({name:"existingObj"});
existingObject.name = "updatedObj";
myCollection.save(existingObj);
myCollection.save({name:"newObj"});
```

**** salvando -------- ejecutando: doc_save.js**

```
01 blog = {
02   word: 'blog', first: 'b', last: 'g',
03   size: 4, letters: ['b','l','o','g'],
04   stats: {vowels: 1, consonants: 3},
05   charsets: [ {type: 'consonants', chars: ['b','l','g']},
06             {type: 'vowels', chars: ['o']} ],
07   category: 'New' };
08 mongo = new Mongo('localhost');
09 wordsDB = mongo.getDB('words');
10 wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
11 wordsColl = wordsDB.getCollection('word_stats');
12 cursor = wordsColl.find({category:"blue"}, {word: 1, category:1});
13 print("Before Existing Save: ");
14 printjson(cursor.toArray());
15 word = wordsColl.findOne({word:"ocean"});
16 word.category="blue";
17 wordsColl.save(word);
18 word = wordsColl.findOne({word:"sky"});
19 word.category="blue";
20 wordsColl.save(word);
21 cursor = wordsColl.find({category:"blue"}, {word: 1, category:1});
22 print("After Existing Save: ");
23 printjson(cursor.toArray());
24 word = wordsColl.findOne({word:"blog"});
25 print("Before New Document Save: ");
26 printjson(word);
27 wordsColl.save(blog);
28 word = wordsColl.findOne({word:"blog"}, {word: 1, category:1});
29 print("After New Document Save: ");
30 printjson(word);
```

** resultado

```
Before Existing Save:
[ ]
After Existing Save:
[
```

```
    {
        "_id" : ObjectId("52e2992e138a073440e46784"),
        "word" : "sky",
        "category" : "blue"
    },
    {
        "_id" : ObjectId("52e2992e138a073440e469f2"),
        "word" : "ocean",
        "category" : "blue"
    }
]
Before New Document Save:
null
After New Document Save:
{
    "_id" : ObjectId("52e29c62073b7a59dcf89ee1"),
    "word" : "blog",
    "category" : "New"
}
```

**** Upserting Documents in Collection (nexts)**

```
update({color:"azure"}, {$set:{red:0, green:127, blue:255}}, true, false);
```

- es el true penúltimo parámetro

** upserting ejecutando: doc_upsert.js

```
01 mongo = new Mongo('localhost');
02 wordsDB = mongo.getDB('words');
03 wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
04 wordsColl = wordsDB.getCollection('word_stats');
05 cursor = wordsColl.find({word: 'righty'},
06                   {word:1, size:1, stats:1, letters:1});
07 print("Before Upsert: ");
08 printjson(cursor.toArray());
09 wordsColl.update({ word: 'righty'},
10               { $set: {word:'righty', size: 4,
11                 letters: ['r','i','g','h'],
12                 'stats.consonants': 3, 'stats.vowels': 1}},
13               true, true);
14 cursor = wordsColl.find({word: 'righty'},
15                   {word:1, size:1, stats:1, letters:1});
16 print("After Upsert: ");
17 printjson(cursor.toArray());
18 wordsColl.update({ word: 'righty'},
19    { $set: {word:'righty', size: 6,
20      letters: ['r','i','g','h','t','y'],
21      'stats.consonants': 5, 'stats.vowels': 1}}, true, true);
22 cursor = wordsColl.find({word: 'righty'},
23                   {word:1, size:1, stats:1, letters:1});
24 print("After Second Upsert: ");
25 printjson(cursor.toArray());
```

**** Deleting Documents from a Collection ----------**

- Borra docs que coinciden con la query
- justone: si quieres borrar solo el primero que encuentre

```
remove([query], [justOne])

collection = myDB.getCollection('word_stats');
collection.remove();
```

The following code deletes all words that start with a from the words_stats collection:

```
collection = myDB.getCollection('word_stats');
collection.remove({first:'a'}, false);
```

The following deletes only the first word that starts with a from the words_stats collection:

```
collection = myDB.getCollection('word_stats');
collection.remove({first:'a'}, true);
```

**** delecting ejecutando: doc_delete.js**

```
 mongo = new Mongo('localhost');
02 wordsDB = mongo.getDB('words');
```

```
03 wordsDB.runCommand( { getLastError: 1, w: 1, j: true, wtimeout: 1000 } );
04 wordsColl = wordsDB.getCollection('word_stats');
05 print("Before Delete One: ");
06 cursor = wordsColl.find({category: 'New'}, {word:1});
07 printjson(cursor.toArray());
08 wordsColl.remove({category: 'New'}, true);
09 cursor = wordsColl.find({category: 'New'}, {word:1});
10 print("After Delete One: ");
11 printjson(cursor.toArray());
12 wordsColl.remove({category: 'New'});
13 cursor = wordsColl.find({category: 'New'}, {word:1});
14 print("After Delete All: ");
15 printjson(cursor.toArray());
```
