MongoDB Data Types

MongoDB stores documents in BSON, which is the binary encoded format of JSON. Basically, the name BSON itself comes from Binary encoded JSON. The BSON data format provides various types, used when we store the JavaScript objects in the binary form. We can make remote procedure calls in MongoDB by using BSON. All the BSON data-types are supported in MongoDB. Below are the enlisted MongoDB data types. Each MongoDB datatypes corresponds a unique number which is used to identify them in $type method.

Each BSON type has both integer and string identifiers as listed in the following table:

| **Type** | **Number** | **Alias** | **Notes** |
| --- | --- | --- | --- |
| Double | 1 | “double” |  |
| String | 2 | “string” |  |
| Object | 3 | “object” |  |
| Array | 4 | “array” |  |
| Binary data | 5 | “binData” |  |
| Undefined | 6 | “undefined” | Deprecated. |
| ObjectId | 7 | “objectId” |  |
| Boolean | 8 | “bool” |  |
| Date | 9 | “date” |  |
| Null | 10 | “null” |  |
| Regular Expression | 11 | “regex” |  |
| DBPointer | 12 | “dbPointer” | Deprecated. |
| JavaScript | 13 | “javascript” |  |
| Symbol | 14 | “symbol” | Deprecated. |
| JavaScript (with scope) | 15 | “javascriptWithScope” |  |
| 32-bit integer | 16 | “int” |  |
| Timestamp | 17 | “timestamp” |  |
| 64-bit integer | 18 | “long” |  |
| Decimal128 | 19 | “decimal” | New in version 3.4. |
| Min key | -1 | “minKey” |  |
| Max key | 127 | “maxKey” |  |

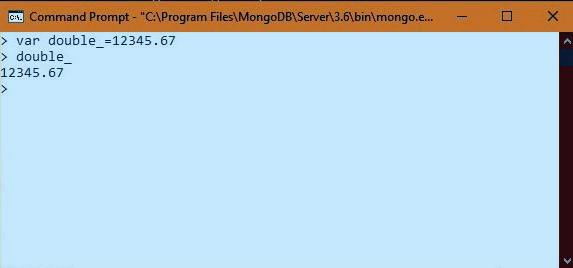
You can use these values with the [$type](https://docs.mongodb.com/manual/reference/operator/query/type/#op._S_type) operator to query documents by their BSON type.

## Different Data Types in MongoDB

Here, we will discuss 16 various MongoDB Data Types with the examples, let’s discuss them one by one:

### a. Double

The double data type is used to store floating point values. Let’s see with an example, how we can do it.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/double.jpg)

*Double – Data Types in MongoDB*

In the above example, we have inserted a floating value 12345.67 in double data type after creating a variable.

[**Read about Important Features of MongoDB**](https://data-flair.training/blogs/mongodb-features/)

### b. String

This is the most commonly used MongoDB data types, BSON strings are UTF-8. Drivers for each programming language convert from the string format of the language to UTF-8 while serializing and de-serializing BSON. This makes possible to easily store most international characters in BSON strings. The string must be valid to be saved.  
**Example-** We have inserted a string in a document. Here, we have used a database “dataflair”.

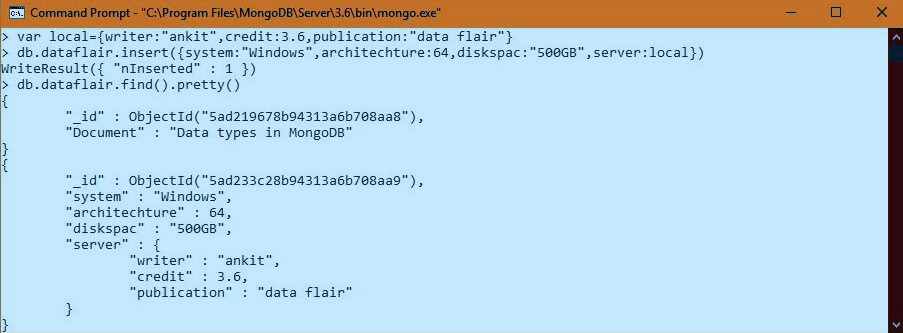
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/string.jpg)

*String – Data Types in MongoDB*

### c. Object

Object data type stores embedded documents. If a document contains another document in the form of the key-value pair then such type of document is known as an embedded document.

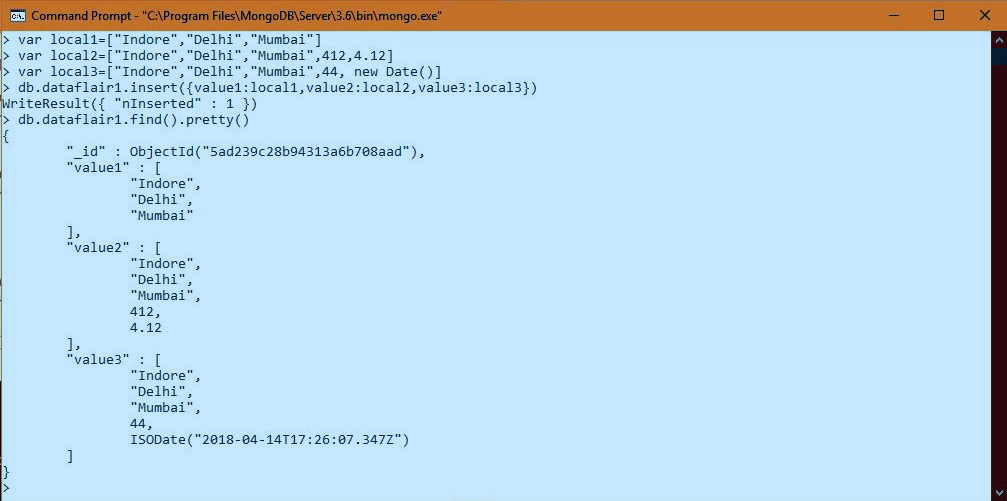
[**Let’s Discuss Advantages and Disadvantages of MongoDB**](https://data-flair.training/blogs/advantages-of-mongodb/)  
**Example-** We have inserted a document named local in another document of database “dataflair”. This is stored in the object data type

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/embedded_document.jpg)

*Object – Data Types in MongoDB*

### d. Array

These MongoDB data types stores array. A set of values are represented as an array. This data type can store multiples of values and data types.  
**Example-** Let’s store some values in variable arrays, local1, local2, local3. We have stored a string, Integer, float, and date data type in these array data type.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/array.jpg)

*Array – Data Types in MongoDB*

[**Do you know What is MongoDB Indexing?**](https://data-flair.training/blogs/mongodb-index/)

### e. Binary Data

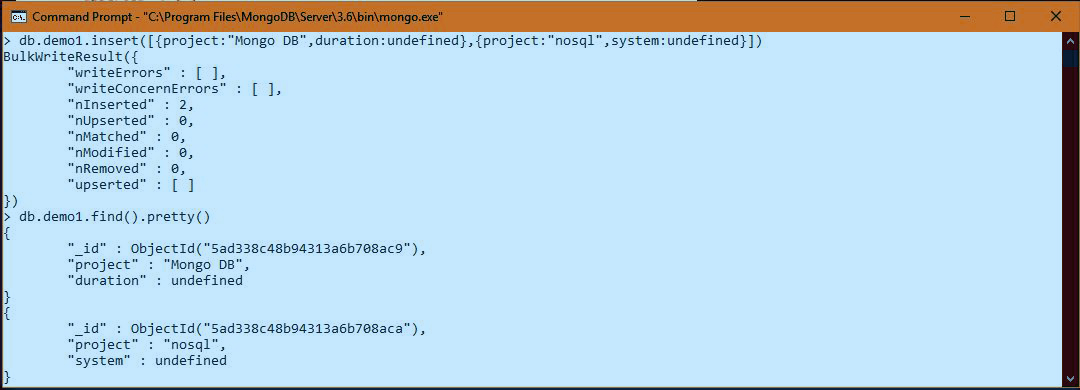
These MongoDB data types store the binary data in it.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/Binary.jpg)

*Binary Data – Data Types in MongoDB*

### f. Undefined

This MongoDB data type stores the undefined values.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/Undefined.jpg)

*Undefined – Data Types in MongoDB*

### g. ObjectId

This data type in MongoDB stores the unique key Id of documents stored. There is an\_id field in MongoDB for each document. The data which is stored in Id is in hexadecimal format. The size of ObjectId is 12 bytes which are divided into four parts as follows.

|  |  |
| --- | --- |
| **Part name** | **Size(bytes)** |
| **Timestamp** | 4 |
| **Machine Id** | 3 |
| **Process Id** | 2 |
| **Counter** | 3 |

**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/ObjectId.jpg)

*ObjectId – Data Types in MongoDB*

[**Read about What is MongoDB DropDatabase() Method**](https://data-flair.training/blogs/mongodb-drop-database/)

### h. Boolean

Boolean data type stores Boolean values i.e. true/false.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/boolean.jpg)

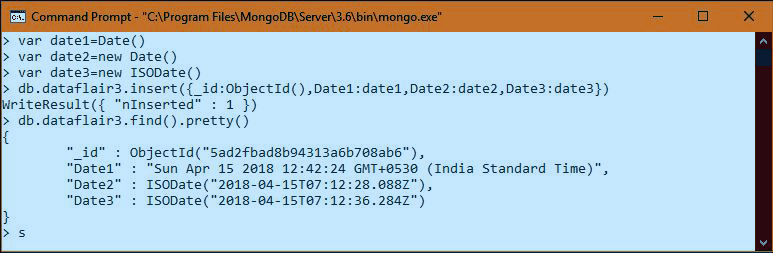
*Boolean – Data Types in MongoDB*

### i. Date

Date data type stores current date or time. There are various methods to return date. It can be either as a string or as a date object. In the below table, we have discussed the methods for the date.

|  |  |
| --- | --- |
| **Date Method** | **Description** |
| **Date()** | It returns the current date in string format. |
| **New Date()** | Returns a date object. Uses the ISODate() wrapper. |
| **ISODate()** | It also returns a date object. Uses the ISODate() wrapper. |

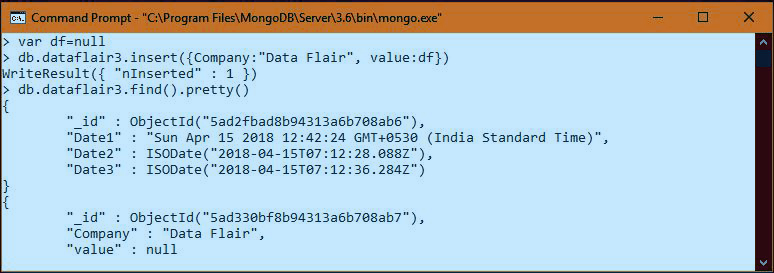
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/Date.jpg)

*Date- Data Types in MongoDB*

### j. Null

This MongoDB data types stores a null value in it.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/null.jpg)

*Null – Data Types in MongoDB*

### k. Regular Expression

These MongoDB data types stores regular expressions in MongoDB. It maps directly to JavaScript RegExp.  
**Example-**

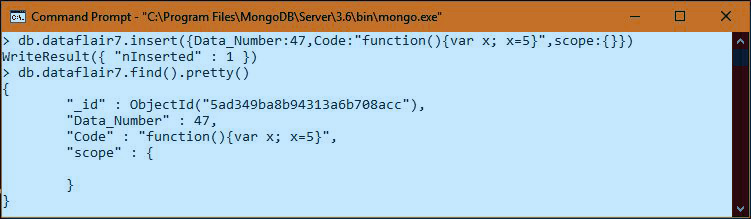
[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/Expression.jpg)

*Regular Expression – Data Types in MongoDB*

[**Follow this link to know about MongoDB Data Modelling with Document Structure**](https://data-flair.training/blogs/mongodb-data-modeling/)

### l. JavaScript

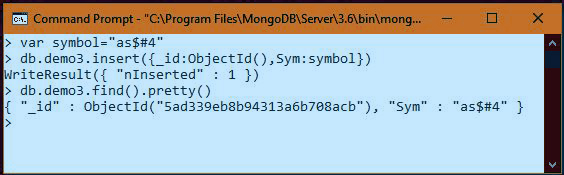
These MongoDB data types store the JavaScript data without a scope. There is another data type to store data with a scope.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/JavaScript.jpg)

*JavaScript – Data Types in MongoDB*

### m. Symbol

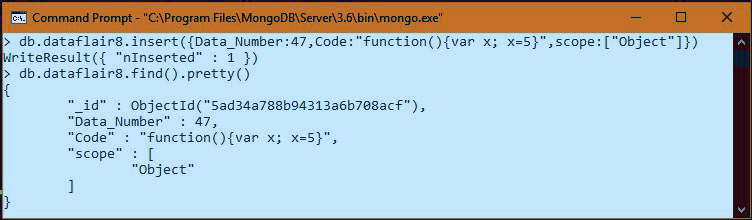
These MongoDB data types similar to the string data type. It is not supported by a shell. But if the shell gets a symbol from the database, it is converted into strings.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/symbols.jpg)

*Symbol – Data Types in MongoDB*

### n. JavaScript with Scope

These MongoDB data types store JavaScript data with a scope.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/JavaWithScope.jpg)

*Javascript with Scope – Data Types in MongoDB*

### o. Integer

These MongoDB data types used to store an integer value. There are two forms available for this data type, 32 bits, and 64 bits.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/integer.jpg)

*Integer – Data Types in MongoDB*

### p. Timestamp

This data type is used to store a timestamp. Also, this is useful when we modify our data to keep a record. This is 64-bit value data type.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/TimeStamp.jpg)

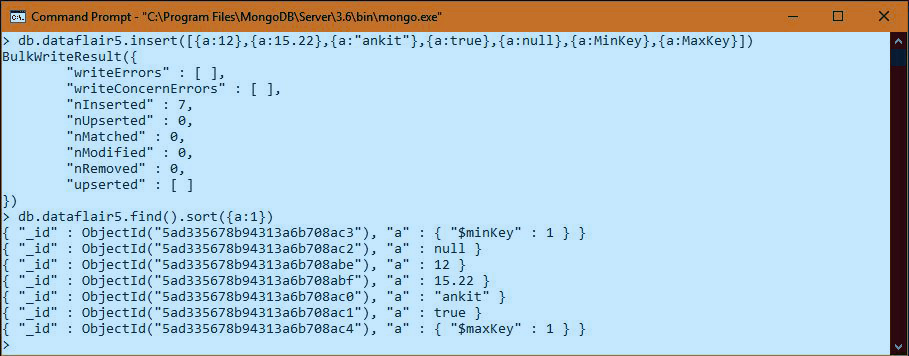
*Timestamp – Data Types in MongoDB*

In the given example the initial value of the timestamp is (0, 0), after that it is (1523790696, 1). Here the first value is the timestamp and another one is an order of operation.

[Do you know How to Install MongoDB on Windows](https://data-flair.training/blogs/mongodb-environment-setup/)

### q. Min & Max key

Min key compares the value of the lowest BSON element. Max key compares the value against the highest BSON element. Both are internal data types.  
**Example-**

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2018/04/MinMaxKey.jpg)

*Min & Min Key – Data Types in MongoDB*

This was all about MongoDB Data Types Tutorial. Hope you like our explanation of different Data Types in MongoDB.

[GridFS](https://docs.mongodb.com/manual/reference/glossary/#term-gridfs) is a specification for storing and retrieving files that exceed the [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson)-document [size limit](https://docs.mongodb.com/manual/reference/limits/#limit-bson-document-size) of 16 MB.

### Querying by Data Type

The addressBook contains addresses and zipcodes, where zipCode has string, int, double, and long values:

copy

copied

db.addressBook.insertMany(

[

{ "\_id" : 1, address : "2030 Martian Way", zipCode : "90698345" },

{ "\_id" : 2, address: "156 Lunar Place", zipCode : 43339374 },

{ "\_id" : 3, address : "2324 Pluto Place", zipCode: NumberLong(3921412) },

{ "\_id" : 4, address : "55 Saturn Ring" , zipCode : NumberInt(88602117) },

{ "\_id" : 5, address : "104 Venus Drive", zipCode : ["834847278", "1893289032"]}

]

)

The following queries return all documents where zipCode is the [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) type string or is an array containing an element of the specified type:

copy

copied

db.addressBook.find( { "zipCode" : { $type : 2 } } );

db.addressBook.find( { "zipCode" : { $type : "string" } } );

These queries return:

copy

copied

{ "\_id" : 1, "address" : "2030 Martian Way", "zipCode" : "90698345" }

{ "\_id" : 5, address : "104 Venus Drive", zipCOde : ["834847278", "1893289032"]}

The following queries return all documents where zipCode is the [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) type double or is an array containing an element of the specified type:

copy

copied

db.addressBook.find( { "zipCode" : { $type : 1 } } )

db.addressBook.find( { "zipCode" : { $type : "double" } } )

These queries return:

copy

copied

{ "\_id" : 2, "address" : "156 Lunar Place", "zip" : 43339374 }

The following query uses the number alias to return documents where zipCode is the [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) type double, int, or long or is an array containing an element of the specified types:

copy

copied

db.addressBook.find( { "zipCode" : { $type : "number" } } )

These queries return:

copy

copied

{ "\_id" : 2, address : "156 Lunar Place", zipCode : 43339374 }

{ "\_id" : 3, address : "2324 Pluto Place", zipCode: NumberLong(3921412) }

{ "\_id" : 4, address : "55 Saturn Ring" , zipCode : 88602117 }

### Querying by Multiple Data Type

The grades collection contains names and averages, where classAverage has string, int, and double values:

copy

copied

db.grades.insertMany(

[

{ "\_id" : 1, name : "Alice King" , classAverage : 87.333333333333333 },

{ "\_id" : 2, name : "Bob Jenkins", classAverage : "83.52" },

{ "\_id" : 3, name : "Cathy Hart", classAverage: "94.06" },

{ "\_id" : 4, name : "Drew Williams" , classAverage : 93 }

]

)

The following queries return all documents where classAverage is the [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) type string or double or is an array containing an element of the specified types. The first query uses numeric aliases while the second query uses string aliases.

copy

copied

db.grades.find( { "classAverage" : { $type : [ 2 , 1 ] } } );

db.grades.find( { "classAverage" : { $type : [ "string" , "double" ] } } );

These queries return the following documents:

copy

copied

{ "\_id" : 1, name : "Alice King" , classAverage : 87.333333333333333 }

{ "\_id" : 2, name : "Bob Jenkins", classAverage : "83.52" }

{ "\_id" : 3, name : "Cathy Hart", classAverage: "94.06" }

### Querying by MinKey and MaxKey

The restaurants collection uses minKey for any grade that is a failing grade:

copy

copied

{

"\_id": 1,

"address": {

"building": "230",

"coord": [ -73.996089, 40.675018 ],

"street": "Huntington St",

"zipcode": "11231"

},

"borough": "Brooklyn",

"cuisine": "Bakery",

"grades": [

{ "date": **new** Date(1393804800000), "grade": "C", "score": 15 },

{ "date": **new** Date(1378857600000), "grade": "C", "score": 16 },

{ "date": **new** Date(1358985600000), "grade": MinKey(), "score": 30 },

{ "date": **new** Date(1322006400000), "grade": "C", "score": 15 }

],

"name": "Dirty Dan's Donuts",

"restaurant\_id": "30075445"

}

And maxKey for any grade that is the highest passing grade:

copy

copied

{

"\_id": 2,

"address": {

"building": "1166",

"coord": [ -73.955184, 40.738589 ],

"street": "Manhattan Ave",

"zipcode": "11222"

},

"borough": "Brooklyn",

"cuisine": "Bakery",

"grades": [

{ "date": **new** Date(1393804800000), "grade": MaxKey(), "score": 2 },

{ "date": **new** Date(1378857600000), "grade": "B", "score": 6 },

{ "date": **new** Date(1358985600000), "grade": MaxKey(), "score": 3 },

{ "date": **new** Date(1322006400000), "grade": "B", "score": 5 }

],

"name": "Dainty Daisey's Donuts",

"restaurant\_id": "30075449"

}

The following query returns any restaurant whose grades.grade field contains minKey or is an array containing an element of the specified type:

copy

copied

db.restaurants.find(

{ "grades.grade" : { $type : "minKey" } }

)

This returns

copy

copied

{

"\_id" : 1,

"address" : {

"building" : "230",

"coord" : [ -73.996089, 40.675018 ],

"street" : "Huntington St",

"zipcode" : "11231"

},

"borough" : "Brooklyn",

"cuisine" : "Bakery",

"grades" : [

{ "date" : ISODate("2014-03-03T00:00:00Z"), "grade" : "C", "score" : 15 },

{ "date" : ISODate("2013-09-11T00:00:00Z"), "grade" : "C", "score" : 16 },

{ "date" : ISODate("2013-01-24T00:00:00Z"), "grade" : { "$minKey" : 1 }, "score" : 30 },

{ "date" : ISODate("2011-11-23T00:00:00Z"), "grade" : "C", "score" : 15 }

],

"name" : "Dirty Dan's Donuts",

"restaurant\_id" : "30075445"

}

The following query returns any restaurant whose grades.grade field contains maxKey or is an array containing an element of the specified type:

copy

copied

db.restaurants.find(

{ "grades.grade" : { $type : "maxKey" } }

)

This returns

copy

copied

{

"\_id" : 2,

"address" : {

"building" : "1166",

"coord" : [ -73.955184, 40.738589 ],

"street" : "Manhattan Ave",

"zipcode" : "11222"

},

"borough" : "Brooklyn",

"cuisine" : "Bakery",

"grades" : [

{ "date" : ISODate("2014-03-03T00:00:00Z"), "grade" : { "$maxKey" : 1 }, "score" : 2 },

{ "date" : ISODate("2013-09-11T00:00:00Z"), "grade" : "B", "score" : 6 },

{ "date" : ISODate("2013-01-24T00:00:00Z"), "grade" : { "$maxKey" : 1 }, "score" : 3 },

{ "date" : ISODate("2011-11-23T00:00:00Z"), "grade" : "B", "score" : 5 }

],

"name" : "Dainty Daisey's Donuts",

"restaurant\_id" : "30075449"

}

## Querying by Array Type

A collection named SensorReading contains the following documents:

copy

copied

{

"\_id": 1,

"readings": [

25,

23,

[ "Warn: High Temp!", 55 ],

[ "ERROR: SYSTEM SHUTDOWN!", 66 ]

]

},

{

"\_id": 2,

"readings": [

25,

25,

24,

23

]

},

{

"\_id": 3,

"readings": [

22,

24,

[]

]

},

{

"\_id": 4,

"readings": []

},

{

"\_id": 5,

"readings": 24

}

The following query returns any document in which the readings field is an array, empty or non-empty.

copy

copied

db.SensorReading.find( { "readings" : { $type: "array" } } )

The above query returns the following documents:

copy

copied

{

"\_id": 1,

"readings": [

25,

23,

[ "Warn: High Temp!", 55 ],

[ "ERROR: SYSTEM SHUTDOWN!", 66 ]

]

},

{

"\_id": 2,

"readings": [

25,

25,

24,

23

]

},

{

"\_id": 3,

"readings": [

22,

24,

[]

]

},

{

"\_id": 4,

"readings": []

}

In the documents with \_id : 1, \_id : 2, \_id : 3, and \_id : 4, the readings field is an array.