**MongoDb Data Types**

Data type is an essential component of a language or script that is used to define the type of data being used in framing the database. It is important for you to know that MongoDB stores data in BSON format. In this chapter, you will learn about the different data types that exist in MongoDB, along with their implementation techniques.

What Are JSON and BSON?

JSON based databases usually return query results which can be effortlessly parsed, having modest or nix transformation, straightforwardly by the use of JavaScript along with most well-liked programming languages - dropping the quantity of logic one needs for building your application layer.

In the case of MongoDB, data representation is done in JSON document format, but here the JSON is binary-encoded, which is termed as BSON. BSON is the extended version of the JSON model, which is providing additional data types, makes performance to be competent to encode and decode in diverse languages and ordered fields.

**Different MongoDB Data Types**

Remote procedure calls in MongoDB can be made by using the BSON format. MongoDB has a unique way of representing data types in which each data type is associated with an alias as well as a number that is usually used to search or find any specific record within a MongoDB database. MongoDB allows its users to implement different variations of data types:

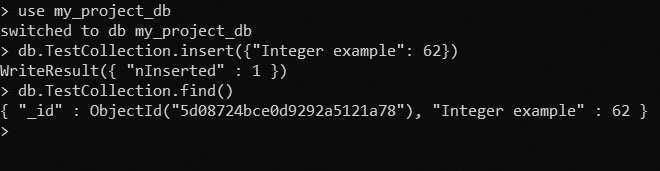
**Integer**

Integer is a data type that is used for storing a numerical value, i.e., integers as you can save in other programming languages. 32 bit or 64-bit integers are supported, which depends on the server.

Example:

db.TestCollection.insert({"Integer example": 62})

Output:



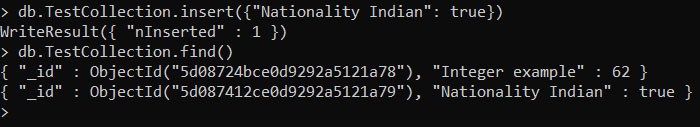
**Boolean**

Boolean is implemented for storing a Boolean (i.e., true or false) values.

Example:

db.TestCollection.insert({"Nationality Indian": true})

Output::



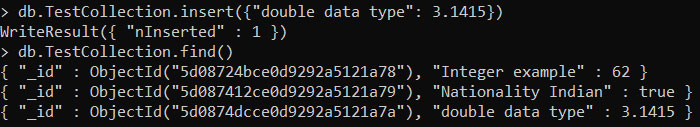
**Double**

Double is implemented for storing floating point data in MongoDB.

Example:

db.TestCollection.insert({"double data type": 3.1415})

Output:



**Min/Max keys**

Min / Max keys are implemented for comparing a value adjacent to the lowest as well as highest BSON elements.

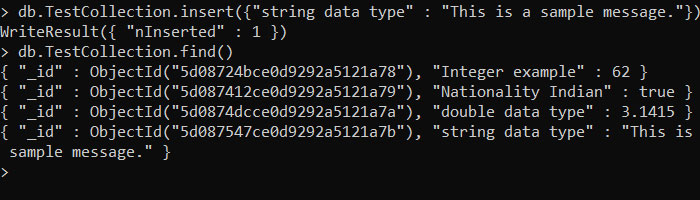
**String**

String is one of the most frequently implemented data type for storing the data.

Example:

db.TestCollection.insert({"string data type" : "This is a sample message."})

Output:



**Arrays**

Arrays are implemented for storing arrays or list type or several values under a single key.

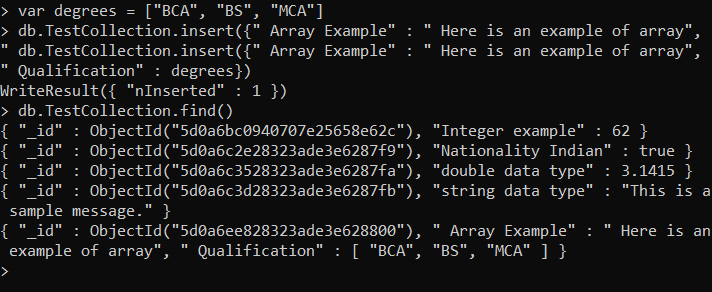
Example:

var degrees = ["BCA", "BS", "MCA"]

db.TestCollection.insert({" Array Example" : " Here is an example of array",

" Qualification" : degrees})

Output:



**Object**

Object is implemented for embedded documents.

Example:

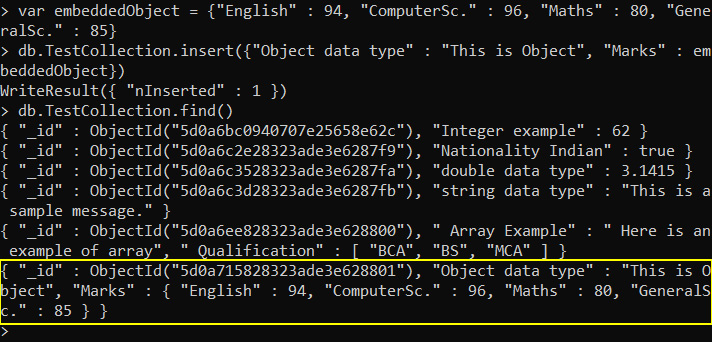
var embeddedObject={"English" : 94, "ComputerSc." : 96, "Maths" : 80,

"GeneralSc." : 85}

db.TestCollection.insert({"Object data type" : "This is Object",

"Marks" : embeddedObject})

Output:



**Symbol**

Symbol is implemented to a string and is usually kept reticent for languages having specific symbol type.

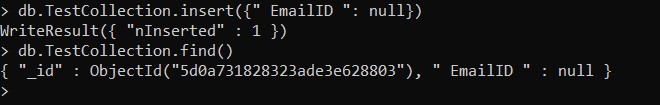
**Null**

Null is implemented for storing a Null value.

Example:

db.TestCollection.insert({" EmailID ": null})

Output:



**Date**

Date is implemented for storing the current date and time as UNIX-time format.

Example:

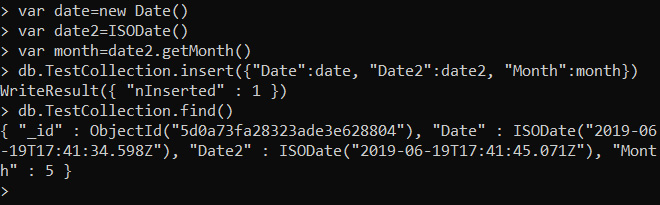
var date=new Date()

var date2=ISODate()

var month=date2.getMonth()

db.TestCollection.insert({"Date":date, "Date2":date2, "Month":month})

Output:



**Timestamp**

Timestamp stores 64-bit value, in which the first 32 bits are time\_t value (seconds epoch) and the other 32 bits are ordinal to operate within a given second.

**Binary data**

Binary data is implemented for storing binary data.

**Object ID**

Object ID is implemented for storing the ID of the document.

**Regular expression**

Regular expression is implemented for storing regular expression.

**Code**

Code is implemented for storing JavaScript code for your MongoDB document.