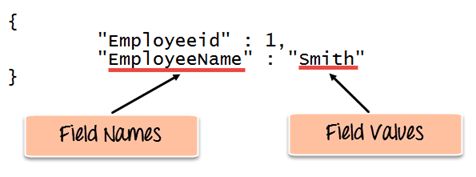
# **How to Create Database & Collection in MongoDB**

In MongoDB, the first basic step is to have a database and collection in place. The database is used to store all of the collections, and the collection in turn is used to store all of the documents. The documents in turn will contain the relevant Field Name and Field values.

The snapshot below shows a basic example of how a document would look like.



The Field Names of the document are "Employeeid" and "EmployeeName" and the Field values are "1" and "Smith' respectively. A bunch of documents would then make up a collection in MongoDB.

* **How to Creating a database using “use” command**
* **Creating a Collection/Table using insert()**
* **Adding documents using insert() command**

## Creating a database using “use” command

Creating a database in MongoDB is as simple as issuing the "**using**" command. The following example shows how this can be done.

The use Command

MongoDB **use DATABASE\_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of **use DATABASE** statement is as follows −

use DATABASE\_NAME

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

>use mydb

switched to db mydb

To check your currently selected database, use the command **db**

>db

mydb

If you want to check your databases list, use the command **show dbs**.

>show dbs

local 0.78125GB

test 0.23012GB

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

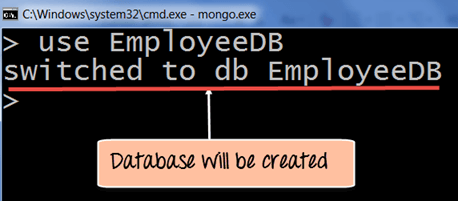
>db.movie.insert({"name":"Rai” not create any database, then collections will be stored in test database.

**Code Explanation:**

1. The **"use"** command is used to create a database in MongoDB. If the database does not exist a new one will be created.

If the command is executed successfully, the following Output will be shown:

**Output:**



MongoDB will automatically switch to the database once created.

**Creating a Collection/Table using insert()**

The easiest way to create a collection is to insert a record (which is nothing but a document consisting of Field names and Values) into a collection. If the collection does not exist a new one will be created.

The following example shows how this can be done.

db.Employee.insert

(

{

"Employeeid" : 1,

"EmployeeName" : "Martin"

}

)

**Code Explanation:**

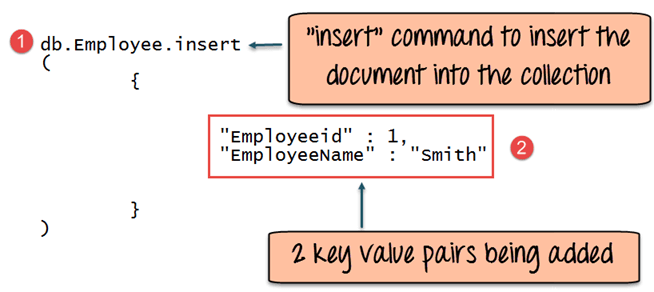
As seen above, by using the **"insert"** command the collection will be created.

**Adding documents using insert() command**

MongoDB provides the **insert () command** to insert documents into a collection. The following example shows how this can be done.

**Step 1)** Write the "insert" command

**Step 2)** Within the "insert" command, add the required Field Name and Field Value for the document which needs to be created.

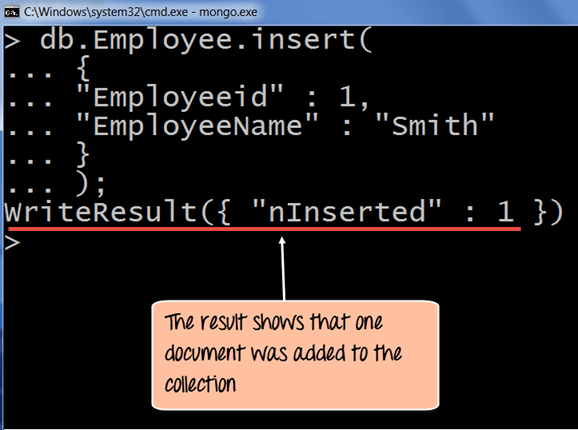


**Code Explanation:**

1. The first part of the command is the "**insert statement"** which is the statement used to insert a document into the collection.
2. The second part of the statement is to add the Field name and the Field value, in other words, what is the document in the collection going to contain.

If the command is executed successfully, the following Output will be shown

**Output:**



The output shows that the operation performed was an insert operation and that one record was inserted into the collection.

# Add MongoDB Array using insert() with Example

The "insert" command can also be used to insert multiple documents into a collection at one time. The below code example can be used to insert multiple documents at a time.

The following example shows how this can be done,

**Step 1)** Create a JavaScript variable called myEmployee to hold the array of documents

**Step 2)** Add the required documents with the Field Name and values to the variable

**Step 3)** Use the insert command to insert the array of documents into the collection

var myEmployee=

[

{

"Employeeid" : 1,

"EmployeeName" : "Smith"

},

{

"Employeeid" : 2,

"EmployeeName" : "Mohan"

},

{

"Employeeid" : 3,

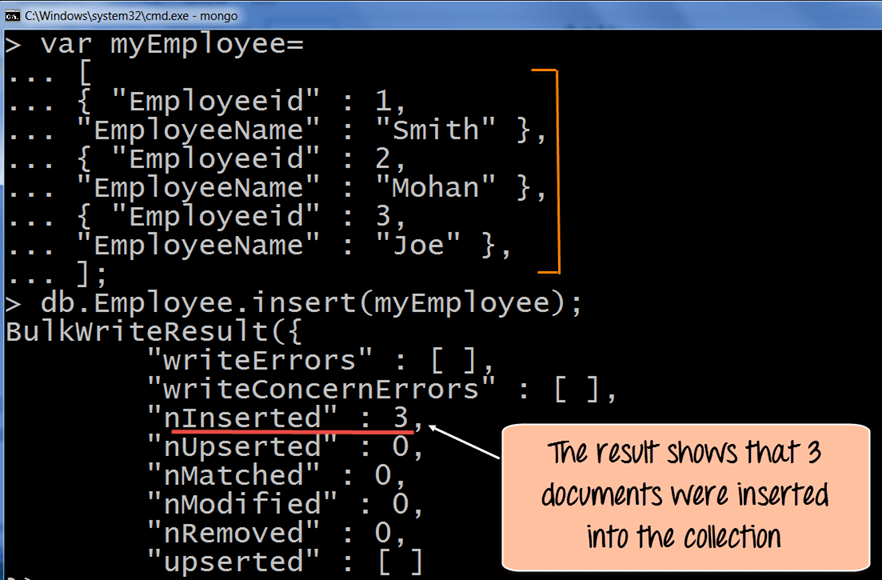
"EmployeeName" : "Joe"

},

];

db.Employee.insert(myEmployee);

If the command is executed successfully, the following Output will be shown



**Printing in JSON format**

JSON is a format called **JavaScript Object Notation**, and is just a way to store information in an organized, easy-to-read manner. In our further examples, we are going to use the JSON print functionality to see the output in a better format.

Let's look at an example of printing in JSON format

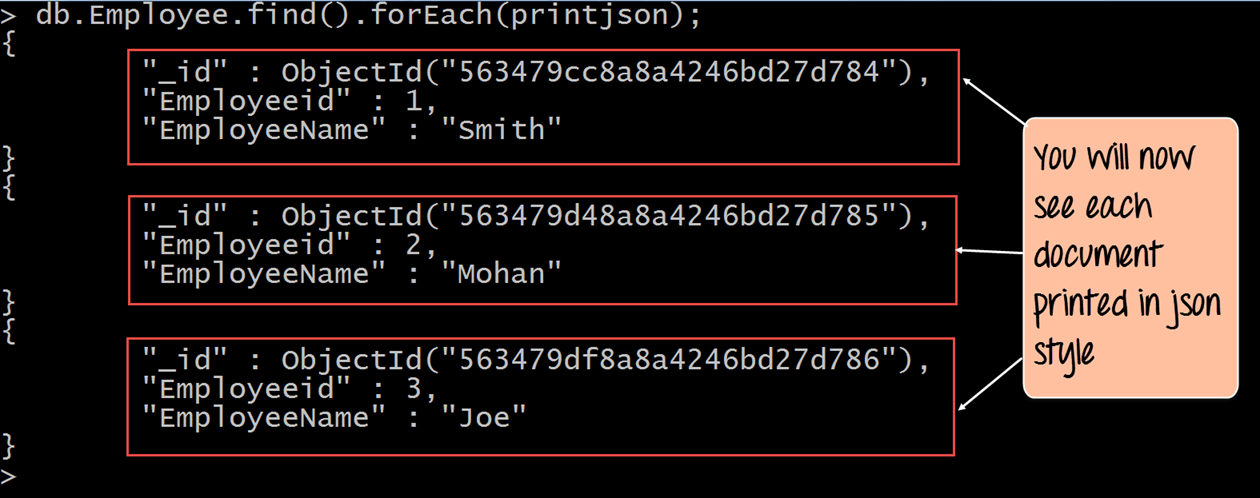
db.Employee.find().forEach(printjson)

**Code Explanation:**

1. The first change is to append the function called for Each() to the find() function. What this does is that it makes sure to explicitly go through each document in the collection. In this way, you have more control of what you can do with each of the documents in the collection.
2. The second change is to put the printjson command to the forEach statement. This will cause each document in the collection to be displayed in JSON format.

If the command is executed successfully, the following Output will be shown

**Output:**

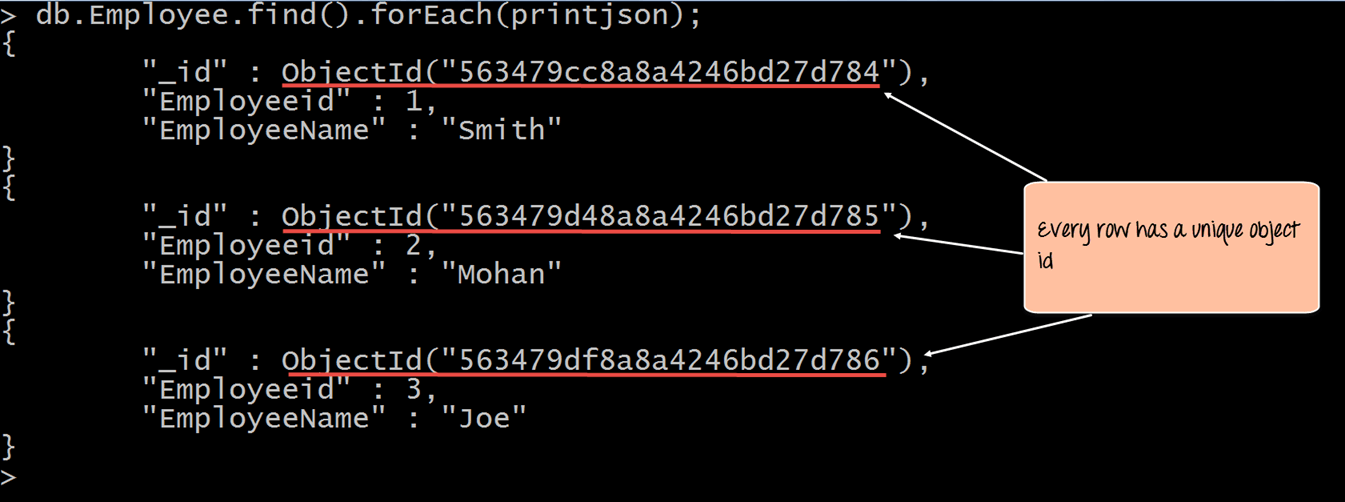


The output clearly shows that all of the documents are printed in JSON style.

### What is Primary Key in MongoDB?

In MongoDB, \_id field as the primary key for the collection so that each document can be uniquely identified in the collection. The \_id field contains a unique ObjectID value.

By default when inserting documents in the collection, if you don't add a field name with the \_id in the field name, then MongoDB will automatically add an Object id field as shown below



When you query the documents in a collection, you can see the ObjectId for each document in the collection.

If you want to ensure that MongoDB does not create the \_id Field when the collection is created and if you want to specify your own id as the \_id of the collection, then you need to explicitly define this while creating the collection.

When explicitly creating an id field, it needs to be created with \_id in its name.

Let's look at an example on how we can achieve this.

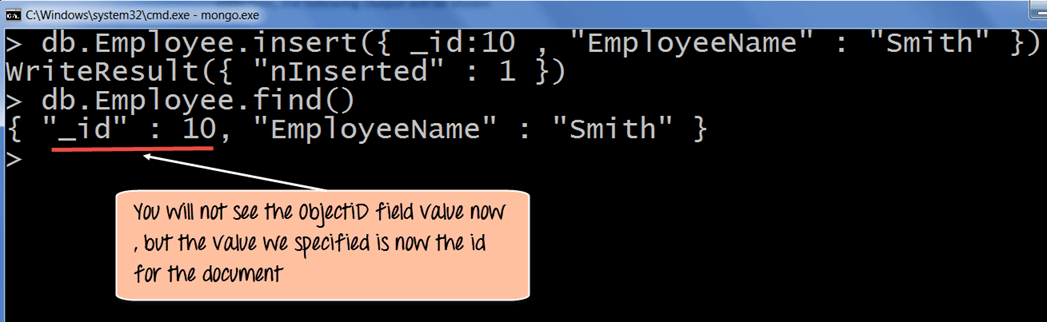
db.Employee.insert({\_id:10, "EmployeeName" : "Smith"})

**Code Explanation:**

1. We are assuming that we are creating the first document in the collection and hence in the above statement while creating the collection, we explicitly define the field \_id and define a value for it.

If the command is executed successfully and now use the find command to display the documents in the collection, the following Output will be shown

**Output:**



The output clearly shows that the \_id field we defined while creating the collection is now used as the primary key for the collection.

db.mycollection.update**(**

**{**

"salary"**:**12000"

}

,

{

$set: "Name" :"Tapan Kumar"

**}**

**)**