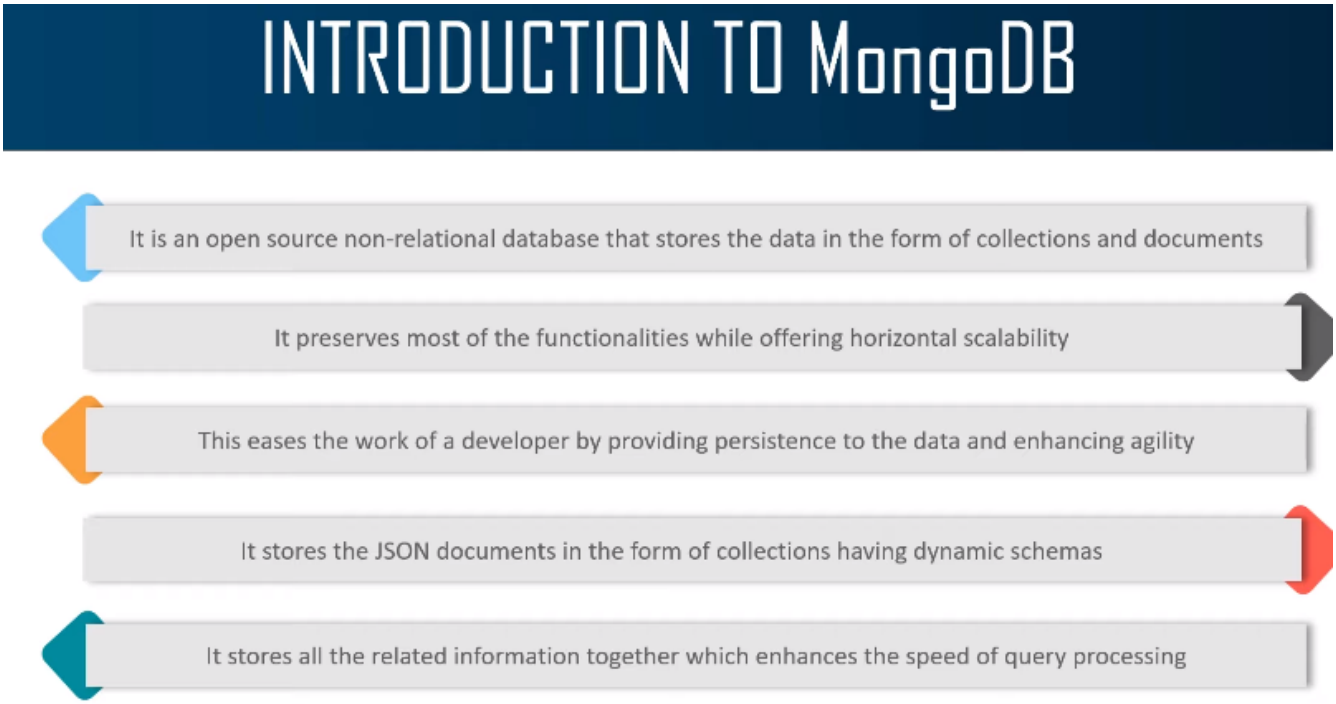
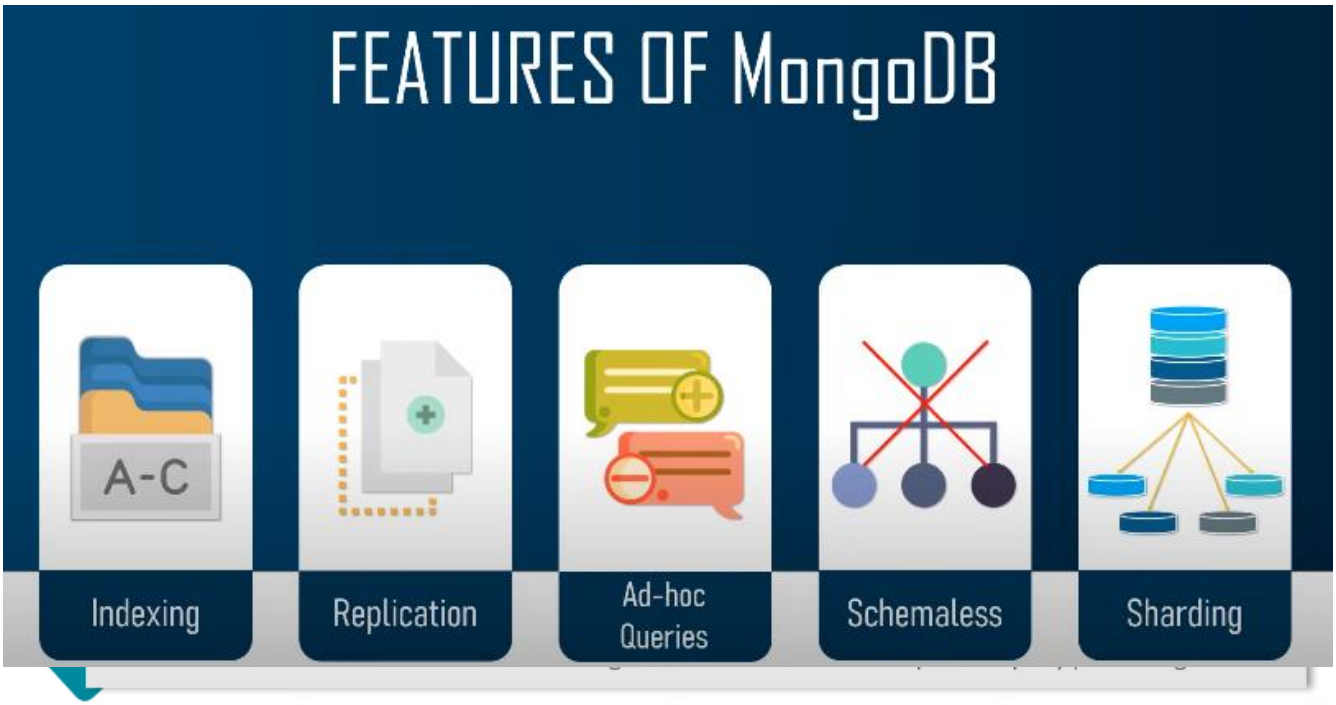
# MONGO DB

## MongoDB Atlas

**Initial release:**
February 11, 2009
**Stable release :**
5.0[2] Edit this on Wikidata / 13 July 2021
**Repository:**
github.com/mongodb/mongo
**Written in:**
C++, JavaScript, Python
**Operating system:**
Windows Vista and later, Linux,
OS X 10.7 and later, Solaris,[3] FreeBSD[4]
**DB Type:**
Type Document-oriented database
**License:**
Server Side Public License
**Website:**
www.mongodb.com




**Official Site Doc Link:**
**https://docs.mongodb.com/manual/crud/**
**https://docs.mongodb.com/manual/tutorial/query-documents/**

## MongoDB

**It is a cross-platform, document oriented database that provides, high performance,
high availability, and easy scalability. MongoDB works on concept of collection and document.**

## Database

Database is a physical container for collections. Each database gets its own set of files on the
file system. A single MongoDB server typically has multiple databases.

## Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A
collection exists within a single database. Collections do not enforce a schema. Documents
within a collection can have different fields. Typically, all documents in a collection are of
similar or related purpose.

## Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema
means that documents in the same collection do not need to have the same set of fields or
structure, and common fields in a collection's documents may hold different types of data.
The following table shows the relationship of RDBMS terminology with MongoDB

**Any relational database has a typical schema design that shows number of tables and the
relationship between these tables. While in MongoDB, there is no concept of relationship.**

## Advantages of MongoDB over RDBMS

 Schema less: MongoDB is a document database in which one collection holds different
documents. Number of fields, content and size of the document can differ from one

document to another.
 Structure of a single object is clear.
 No complex joins.
 Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
 Tuning.
 Ease of scale-out: MongoDB is easy to scale.
 Conversion/mapping of application objects to database objects not needed.
 Uses internal memory for storing the (windowed) working set, enabling faster access
of data.

## Why Use MongoDB?

 Document Oriented Storage: Data is stored in the form of JSON style documents.
 Index on any attribute
 Replication and high availability
 Auto-sharding
 Rich queries
 Fast in-place updates
 Professional support by MongoDB

## Where to Use MongoDB?

 Big Data
 Content Management and Delivery
 Mobile and Social Infrastructure
 User Data Management
 Data Hub

## Installation:-

Mongo DB From Below URL (32 Bit)

https://www.mongodb.org/dl/win32/i386

Download File :- (mongodb-win32-i386-v3.2-latest-signed.msi) and install.

Now create a Project Like (D:\MERN\NodeExpressMongo_Project)

open CMD
> npm init
replace index.js to app.js

you will get your package.json file. Now install some dependencies.

```
> npm install express
>npm install mongodb
>npm install mongoose
>npm install -g nodemon –save-dev
```

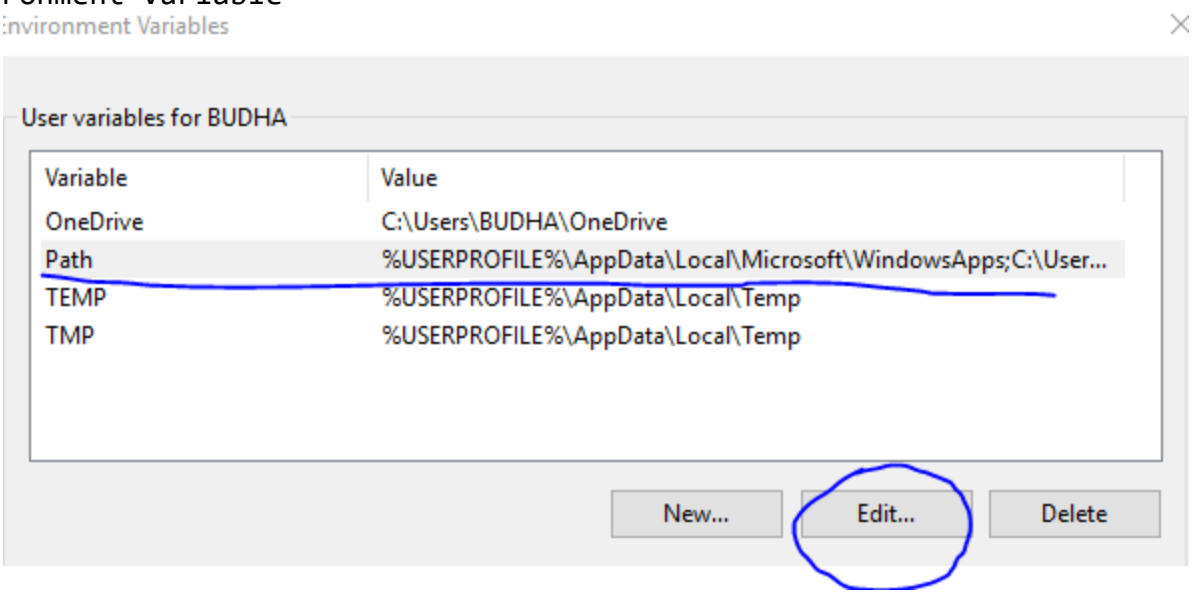
Package.json -

```
{
  "name": "nodeexpressmongo_project",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodemon app.js"  // replace test with start.
  },
  "author": "MEN",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "mongodb": "^4.0.0",
    "mongoose": "^5.13.3"
  },
  "devDependencies": {
    "nodemon": "^2.0.12"
  }
}
```

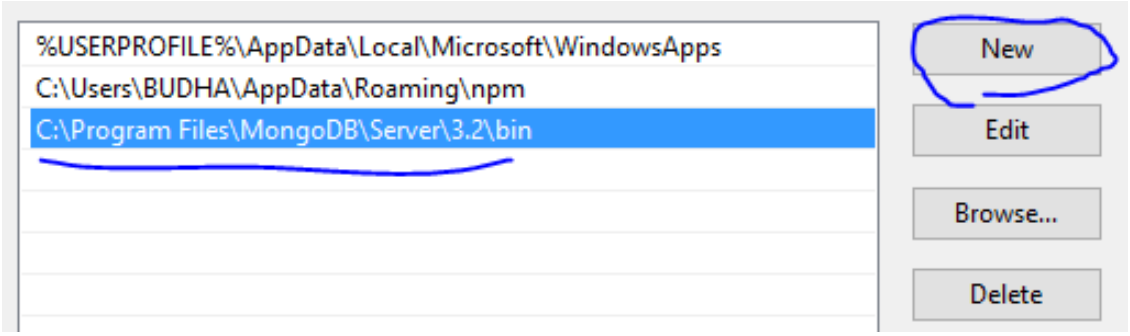
## Configuring:-

environment Variable

C:\Program File



s\MongoDB\Server\3.2\bin



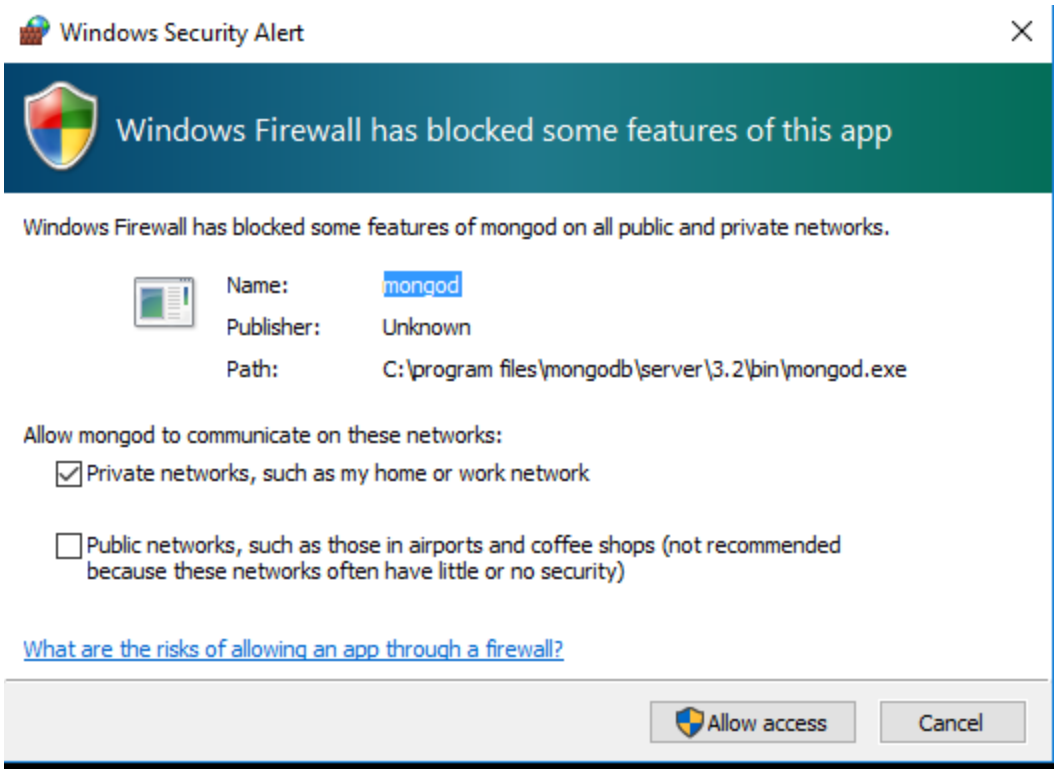

**Create  a folder as below (To save data in local Mongo DB): -**

C:\data\db

**Go to your Project path and open cmd and run below command  to start server (only for 32 bit PC.)**

D:\MERN\NodeExpressMongo_Project>**mongod --storageEngine=mmapv1 --dbpath "C:\data\db"**

**you will get below screen**




**for 64 bit PC**

D:\MERN\NodeExpressMongo_Project>**mongod**

**IF your Server started successfully you will see below screen.**

```
2021-07-18T19:22:51.126+0530 I CONTROL  [main]
2021-07-18T19:22:51.154+0530 I CONTROL  [initandlisten] MongoDB starting : pid=620 port=27017 dbpath=C:\data\db 32-bit h
ost=ExploreMe-PC
2021-07-18T19:22:51.154+0530 I CONTROL  [initandlisten] targetMinOS: Windows Vista/Windows Server 2008
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] db version v3.2.21-2-g105acca0d4
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] git version: 105acca0d443f9a47c1a5bd608fd7133840a58dd
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] modules: none
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] build environment:
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten]     distarch: i386
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten]     target_arch: i386
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] options: { storage: { dbPath: "C:\data\db", engine: "mmapv1" } }

2021-07-18T19:22:52.470+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.470+0530 I CONTROL  [initandlisten] ** WARNING: This 32-bit MongoDB binary is deprecated
2021-07-18T19:22:52.471+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.471+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.472+0530 I CONTROL  [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2021-07-18T19:22:52.472+0530 I CONTROL  [initandlisten] **       32 bit builds are limited to less than 2GB of data (or
less with --journal).
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten] **       Note that journaling defaults to off for 32 bit and is
currently off.
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten] **       See http://dochub.mongodb.org/core/32bit
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:53.062+0530 I NETWORK  [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2021-07-18T19:22:56.736+0530 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C
:/data/db/diagnostic.data'
2021-07-18T19:22:56.744+0530 I NETWORK  [initandlisten] waiting for connections on port 27017
```

**You need to open two terminal**

C:\Windows\System32\cmd.exe - mongod --storageEngine=mmapv1 --dbpath "C:\data\db"

```
D:\MERN\NodeExpressMongo_Project>mongod --storageEngine=mmapv1 --dbpath "C:\data\db"
```

C:\Windows\System32\cmd.exe - nodemon run start

```
D:\MERN\NodeExpressMongo_Project>nodemon run start
[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node run start app.js`
(node:4328) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine i
in a future version. To use the new Server Discover and Monitoring engine, pass option
he MongoClient constructor.
(Use `node --trace-warnings ...` to show where the warning was created)
connected..
```

1. D:\MERN\NodeExpressMongo_Project>mongod --storageEngine=mmapv1 --dbpath "C:\data\db"
(to keep server running)

2. D:\MERN\NodeExpressMongo_Project>nodemon run start

to run your application.

**Adding Code to get response using PostMan**

Now create a app.js file and add below code.

```
const express = require('express');
const mongoose = require('mongoose');
const url = 'mongodb://localhost/MENApp';
const app = express();

mongoose.connect(url, {useNewUrlParser: true});
const con = mongoose.connection;
con.on('open', () => {
    console.log('connected..')
});

const customRoute = require('./Router/menappRoute');
app.use('/details', customRoute); // this is a middleware.
app.listen(9000, () => {
    console.log('Server Started.')
});
```

IF your server work correctly you will get connected… message in CMD.

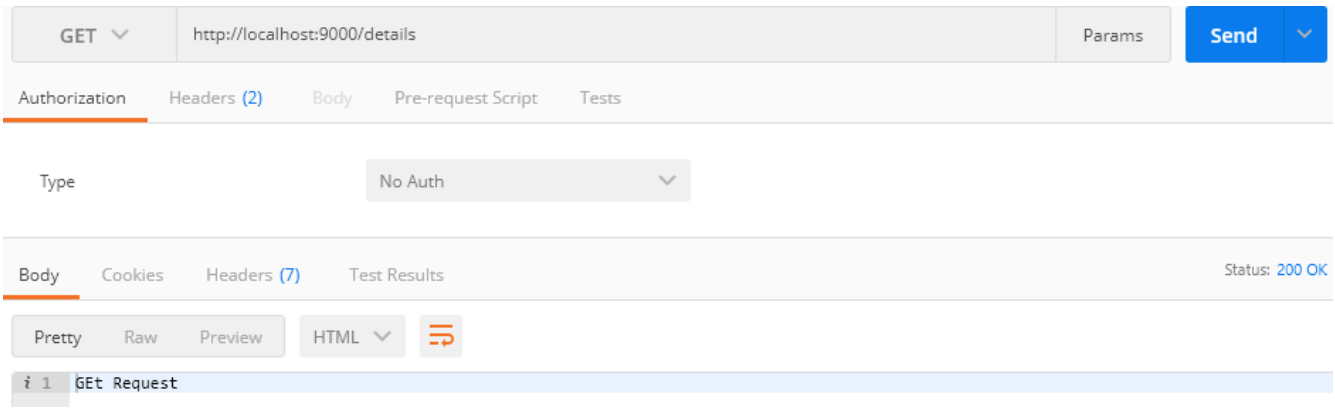
Now we need to create Router to use (GET/POST/PATCH/DELETE Operations)

so create a folder let say Router → menAppRoute.js and add below code

```
const express = require('express');
const router = express.Router();
router.get('/', (req, res) => {
    res.send('GEt Request');
});

module.exports = router;
```

Now press send button of your Postman. (URL : http://localhost:9000/details)




# MONGO DB COMMANDS

**To run mongo commands first you have to open a new command prompt and run below command (for 32 bit) to keep database active and in running state.**

```
> mongod --storageEngine=mmapv1 --dbpath "C:\data\db"(mongod for  64  bit)
```

**Then run below command in a new command prompt**

```
> mongo
```

**you will see a shell like below with a cursor blinking.**

```
D:\MERN\MEN_CRUD_App\MEN_Project>mongo
MongoDB shell version: 3.2.22
connecting to: test
Server has startup warnings:
2021-10-09T19:20:16.377+0530 I CONTROL
2021-10-09T19:20:16.378+0530 I CONTROL
2021-10-09T19:20:16.380+0530 I CONTROL
2021-10-09T19:20:16.383+0530 I CONTROL
2021-10-09T19:20:16.385+0530 I CONTROL
2021-10-09T19:20:16.387+0530 I CONTROL
less with --journal).
2021-10-09T19:20:16.390+0530 I CONTROL
currently off.
2021-10-09T19:20:16.395+0530 I CONTROL
2021-10-09T19:20:16.397+0530 I CONTROL
> show dbs
local  0.078GB
> use local
switched to db local
> show collections
startup_log
system.indexes
> use mdb
switched to db mdb
>
```

**1. To  see all db**
**> show dbs**

**2. To create new db**
**> use   db_name**

**To see your newely created db you need to insert atleast one record.**

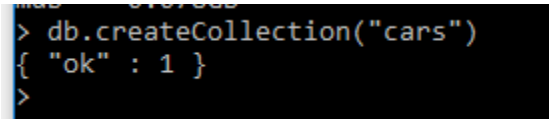
**3. db.dbs.insert({"mdb_userName": "Sandee"});**

**4. show dbs**

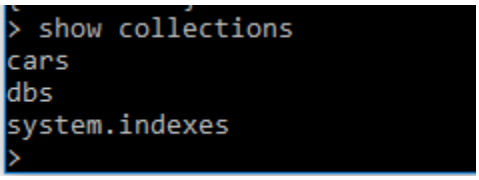
```
> show dbs
local  0.078GB
> use local
switched to db local
> show collections
startup_log
system.indexes
> use mdb
switched to db mdb
> db.dbs.insert({"mdb_userName": "Sandee"});
WriteResult({ "nInserted" : 1 })
> show dbs
local  0.078GB
mdb    0.078GB
>
```

**creating collections**

**> db.createCollection("cars")**




**> show collections**




**Insert document inside collection**

```
> db.cars.insert({name: 'TATA', manufacture: 2010, model: "tata indica",
price: 500000})
 db.cars.insert({name: 'TATA', manufacture: 2010, model: "tata indica",
price: 500000})
```

**show docs created inside collection**

```
> db.cars.find().pretty()
```

```
> db.cars.insert({name: 'TATA', manufacture: 2010, model: "tata indica", price: 500000})
WriteResult({ "nInserted" : 1 })
> db.cars.find().pretty()
{ "_id" : ObjectId("6161a53a164b3d58b89f1f60") }
{
        "_id" : ObjectId("6161a57b164b3d58b89f1f61"),
        "name" : "TATA",
        "manufacture" : 2010,
        "model" : "tata indica",
        "price" : 500000
}
> db.cars.insert({name:'honda',manufacture: 2020, model: 'fortuner', price: 220000})
WriteResult({ "nInserted" : 1 })
> db.cars.find().pretty()
{ "_id" : ObjectId("6161a53a164b3d58b89f1f60") }
{
        "_id" : ObjectId("6161a57b164b3d58b89f1f61"),
        "name" : "TATA",
        "manufacture" : 2010,
        "model" : "tata indica",
        "price" : 500000
}
{
        "_id" : ObjectId("6161a69f164b3d58b89f1f62"),
        "name" : "honda",
        "manufacture" : 2020,
        "model" : "fortuner",
        "price" : 220000
}
>
```

**Updating specific field inside any record.**

```
> db.cars.update({name: 'honda'}, {$set: {name: 'Ford'}})
```

```
> db.cars.update({name: 'honda'}, {$set: {name: 'Ford'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.cars.find().pretty()
{ "_id" : ObjectId("6161a53a164b3d58b89f1f60") }
{
        "_id" : ObjectId("6161a57b164b3d58b89f1f61"),
        "name" : "TATA",
        "manufacture" : 2010,
        "model" : "tata indica",
        "price" : 500000
}
{
        "_id" : ObjectId("6161a69f164b3d58b89f1f62"),
        "name" : "Ford",
        "manufacture" : 2020,
        "model" : "fortuner",
        "price" : 220000
}
>
```

**Adding new field to existing record.**

**> db.cars.update({name: 'Ford'}, {$set: {quantity: 400}}, {$upsert: true})**