**MongoDB Notes :**

**Show Databases :** show dbs

**Switch to DB :** use db\_name

**Show Collections (tables):** show collections

**Run js file :** load(“myScript.js”);

**CRUD :**

**Create :** Note : coll means coll’n name.

db.coll.insertOne({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}})

db.coll.insertOne({date: ISODate()})

db.coll.insertMany([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert

db.coll.insertMany([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered bulk insert

**Read :**

db.coll.findOne() // returns a single document

db.coll.find()

db.coll.find({name: "Max", age: 32}) // implicit logical "AND".

// Count

db.coll.countDocuments({age: 32})

// Comparison

db.coll.find({"year": {$gt: 1970}})

db.coll.find({"year": {$in: [1958, 1959]}})

// Logical

db.coll.find({name:{$not: {$eq: "Max"}}})

db.coll.find({$or: [{"year" : 1958}, {"year" : 1959}]})

**Update :**

db.coll.updateOne({"\_id": 1}, {$set: {"year": 2016, name: "Max"}})

db.coll.updateOne({"\_id": 1}, {$unset: {"year": 1}})

db.coll.updateOne({"\_id": 1}, {$rename: {"year": "date"} })

db.coll.updateOne({"\_id": 1}, {$inc: {"year": 5}})

db.coll.updateOne({"\_id": 1}, {$mul: {price: NumberDecimal("1.25"), qty: 2}})

db.coll.updateOne({"\_id": 1}, {$min: {"imdb": 5}})

db.coll.updateOne({"\_id": 1}, {$max: {"imdb": 8}})

db.coll.updateOne({"\_id": 1}, {$currentDate: {"lastModified": true}})

db.coll.updateOne({"\_id": 1}, {$currentDate: {"lastModified": {$type: "timestamp"}}})

**Delete :**

db.coll.deleteOne({name: "Max"}) //delete first doc which matches with filter.

db.coll.deleteMany({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}}

db.coll.deleteMany({}) //delete all documents.

db.coll.findOneAndDelete({"name": "Max"}) //deletes the first doc and returns that doc, normal delete returns the no. of docs that has been deleted.

**Drop :**

db.coll.drop() // removes the collection and its index definitions

db.dropDatabase() //removes the db.

**Creating Schema :**

In MongoDB, creating a schema is not mandatory since it is a schema-less NoSQL database. However, you can enforce a schema at the collection level using JSON Schema validation. This allows you to define rules for the structure of documents in a collection.

db.createCollection("persons", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["email", "name", "fullname", "address"],

properties: {

email: {

bsonType: "string",

description: "must be a string and is required"

},

name: {

bsonType: "string",

description: "must be a string and is required"

},

}

}

}

});

**Insert Document :**

db.persons.insertOne({

email: "john.doe@example.com",

name: "John"

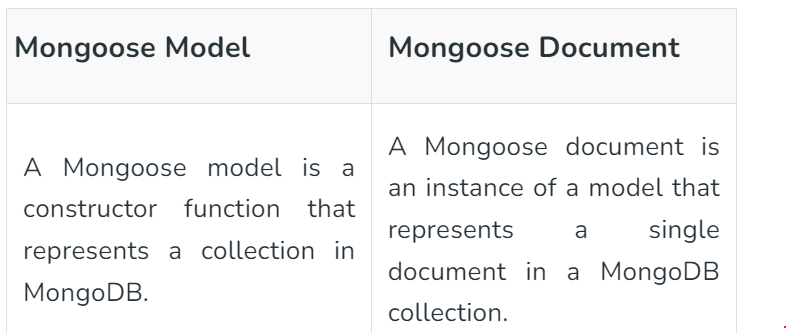
});

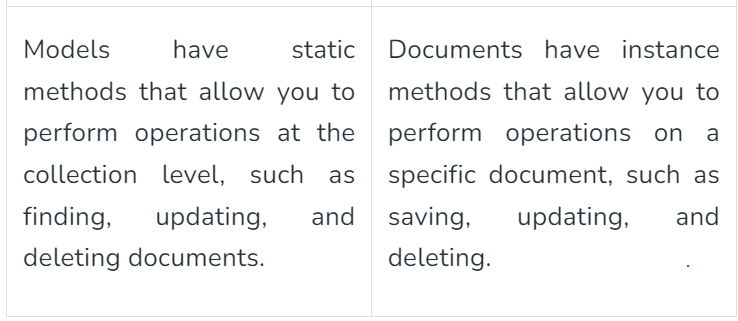
**Mongoose :**

**What is mongoose ?**

Mongoose is an Object Data Modeling (ODM) library for MongoDB. MongoDB is a NoSQL database and Mongoose is used to interact with MongoDB by providing a schema-based solution.

Uses of Mongoose :





If you are unable to understand just understand first difference and remember 2nd as it is.

**Advantages of Mongoose ?**

It provides many features like schema validation, middleware support, and easy query building.

**Basic Structure of How to build mongoose connection how to start server, how to connect Node with MongoDB and how to create schemas :**

**Server.js**

// const express = require('express');

import express from "express";

const app = express();

import dotenv from 'dotenv';

import databaseConnect from './config/database.js'

import authRouter from './routes/authRoute.js'

import bodyParser from 'body-parser';

import cookieParser from 'cookie-parser';

import messengerRoute from './routes/messengerRoute.js';

dotenv.config({

     path : 'backend/config/config.env'

})

app.use(bodyParser.json());

app.use(cookieParser());

app.use('/api/messenger',authRouter);  //here authRouter is file name where remaining url may present therefore using app.use here.

app.use('/api/messenger',messengerRoute);

const PORT = process.env.PORT || 4000;

app.get('/', (req, res)=>{

     res.send('This is from backend Sever')

})

databaseConnect();

app.listen(PORT, ()=>{

     console.log(`Server is running on port ${PORT}`)

})

**Database.js**

import mongoose from 'mongoose';

const databaseConnect = () => {

     mongoose.connect(process.env.DATABASE\_URL,{

          useNewUrlParser : true,

          useUnifiedTopology : true

     }).then(()=>{

          console.log('Mongodb Database Connected');

     }).catch(error=>{

          console.log(error)

     })

}

// module.exports = databaseConnect;

export default databaseConnect;

**messageModel.js**

import mongoose from 'mongoose';

const {model,Schema}=mongoose;

const messageSchema = new Schema({

     senderId : {

          type : String,

          required : true

     },

     senderName : {

          type: String,

          required : true

     },

     reseverId : {

          type: String,

          required : true

     },

     message : {

          text : {

               type: String,

               default : ''

          },

          image : {

               type : String,

               default : ''

          }

     },

     deleMsgMine:{

          type:String,

          default:''

     },

     status :{

          type : String,

          default : 'unseen'

     }

},{timestamps : true});

export default model('message',messageSchema);

**Now how to use this Schema ?**

We can directly import the model ex. messageSchema here and then use the mongoose operations on it.