Name : Mayuri Shridatta Yerande
Class : D15B
Roll no. : 70

Experiment – Module 4: MongoDB

AIM: To study and implement Module 4: Mongodb and REST API

PROBLEM STATEMENT:

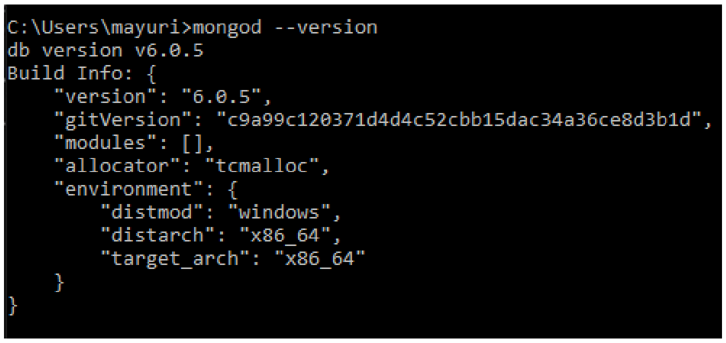
1. Build a RESTful API using MongoDB.
2. Create a full database in Mongodb
   a) create a bd: question-> code-> snapshot of result
   b) create collections
   c) create doc: insert one doc and insert many documents at once
   d) delta one or many documents
   e) search with conditions

IMPLEMENTATION:

★ **INSTALLATION:**

Link: https://www.mongodb.com/try/download/community
Link: https://www.mongodb.com/try/download/shell

- ➢ After Installation,Agree to the terms and conditions, click on next
- ➢ Choose "complete" package to install all the features of MongoDB
- ➢ Choose the default service name and click install
- ➢ Check the installation using cmd
- ➢ Add the environment variables for both MongoDB and MongoShell



```
C:\Users\mayuri>mongod --version
db version v6.0.5
Build Info: {
    "version": "6.0.5",
    "gitVersion": "c9a99c120371d4d4c52cbb15dac34a36ce8d3b1d",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

★ **REST API**

1. **In command prompt, Run the following commands**
- npm init
- npm install express
- npm install mongoose@6.10.2
- npm install body-parser
- npm install -g nodemon

2. **Create index.js, contactModel.js, contactcontroller.js, api-routes.js files**
3. **Mongodb --> create a db named as "phone" with collection name as "one" and update that name in index.js**

Index.js

```
// Import express
let express = require('express');
// Import Body parser
let bodyParser = require('body-parser');
// Import Mongoose
let mongoose = require('mongoose');
// Initialise the app
let app = express();


// Import routes
let apiRoutes = require("./api-routes");
// Configure bodyparser to handle post requests
app.use(bodyParser.urlencoded({
    extended: true
}));
app.use(bodyParser.json());
// Connect to Mongoose and set connection variable
mongoose.connect('mongodb://localhost:27017/phone', { useNewUrlParser: true });
var db = mongoose.connection;
```

```
// Added check for DB connection
if (!db)
    console.log("Error connecting db")
else
    console.log("Db connected successfully")


// Setup server port
var port = process.env.PORT || 8080;


// Send message for default URL
app.get('/', (req, res) => res.send('Hello World with Express'));


// Use Api routes in the App
app.use('/api', apiRoutes);
// Launch app to listen to specified port
app.listen(port, function() {
    console.log("Running RestHub on port " + port);
});
```

contactModel.js

```
var mongoose = require('mongoose'); // Setup schema
var contactSchema = mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    gender: String,
    phone: String,
    create_date: {
        type: Date,
        default: Date.now
    }
}); // Export Contact model
var Contact = module.exports = mongoose.model('contact', contactSchema);
module.exports.get = function(callback, limit) {
    Contact.find(callback).limit(limit);
}
```

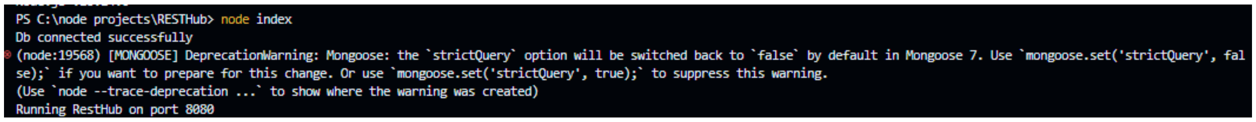
ContactController.js

```
// contactController.js// Import contact model
Contact = require('./contactModel'); // Handle index actions
exports.index = function(req, res) {
    Contact.get(function(err, contacts) {
        if (err) {
            res.json({
                status: "error",
                message: err,
            });
        }
        res.json({
            status: "success",
            message: "Contacts retrieved successfully",
            data: contacts
        });
    });
}; // Handle create contact actions
exports.new = function(req, res) {
    var contact = new Contact();
    contact.name = req.body.name ? req.body.name : contact.name;
    contact.gender = req.body.gender;
    contact.email = req.body.email;
    contact.phone = req.body.phone; // save the contact and check for errors
    contact.save(function(err) {
        if (err)
            res.json(err);
        res.json({
            message: 'New contact created!',
            data: contact
        });
    });
}; // Handle view contact info
exports.view = function(req, res) {
    Contact.findById(req.params.contact_id, function(err, contact) {
        if (err)
            res.send(err);
        res.json({
            message: 'Contact details loading..',
            data: contact
        });
    });
}; // Handle update contact info
exports.update = function(req, res) {
    Contact.findById(req.params.contact_id, function(err, contact) {
        if (err)
            res.send(err);
        contact.name = req.body.name ? req.body.name : contact.name;
        contact.gender = req.body.gender;
        contact.email = req.body.email;
        contact.phone = req.body.phone; // save the contact and check for
errors
        contact.save(function(err) {
            if (err)
                res.json(err);
            res.json({
                message: 'Contact Info updated',
                data: contact
            });
        });
    });
}; // Handle delete contact
exports.delete = function(req, res) {
    Contact.remove({
        _id: req.params.contact_id
    }, function(err, contact) {
        if (err)
            res.send(err);
        res.json({
            status: "success",
            message: 'Contact deleted'
        });
    });
};
```

api-routes.js

```
// api-routes.js// Initialize express router
let router = require('express').Router(); // Set default API response
router.get('/', function(req, res) {
    res.json({
        status: 'API Its Working',
        message: 'Welcome to RESTHub crafted with love!',
    });
}); // Import contact controller
var contactController = require('./contactController'); // Contact routes
router.route('/contacts')
    .get(contactController.index)
    .post(contactController.new);
router.route('/contacts/:contact_id')
    .get(contactController.view)
    .patch(contactController.update)
    .put(contactController.update)
    .delete(contactController.delete); // Export API routes
module.exports = router;
```

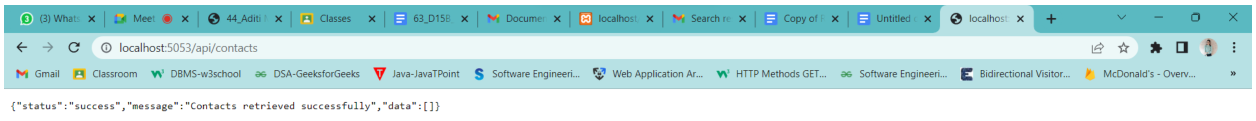
4. Run node index in vscode terminal and check if db is connected successfully.



```
PS C:\node projects\RESTHub> node index
Db connected successfully
(node:19568) [MONGOOSE] DeprecationWarning: Mongoose: the `strictQuery` option will be switched back to `false` by default in Mongoose 7. Use `mongoose.set('strictQuery', fal
se);` if you want to prepare for this change. Or use `mongoose.set('strictQuery', true);` to suppress this warning.
(Use `node --trace-deprecation ...` to show where the warning was created)
Running RestHub on port 8080
```

5. Install thunder client

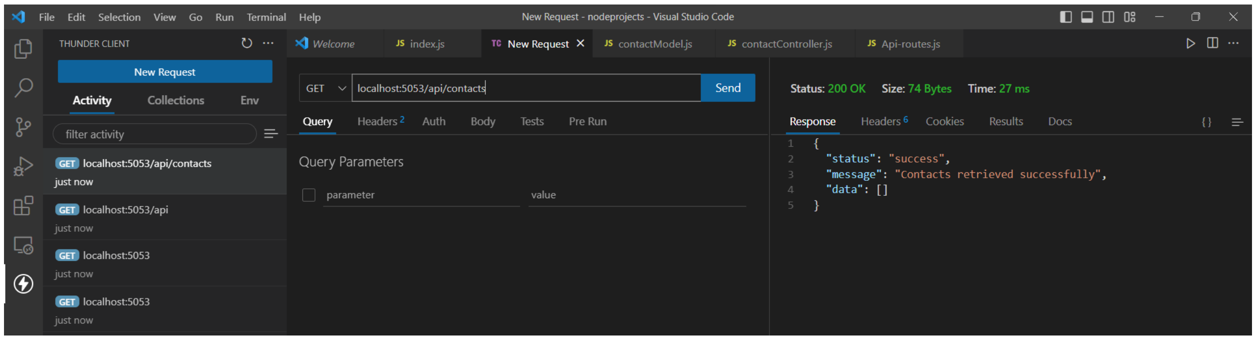
6. Go to localhost:5053/api/contacts



localhost:5053/api/contacts

```
{"status":"success","message":"Contacts retrieved successfully","data":[]}
```

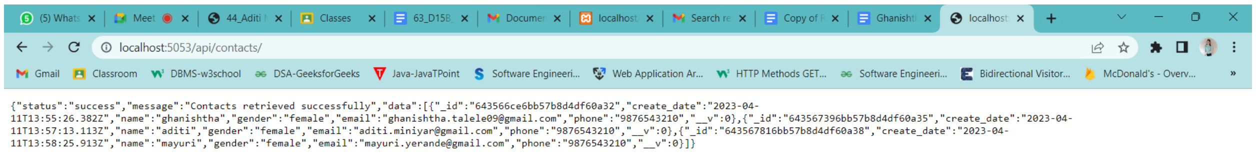
7. GET Contacts



GET localhost:5053/api/contacts

Status: 200 OK Size: 74 Bytes Time: 27 ms

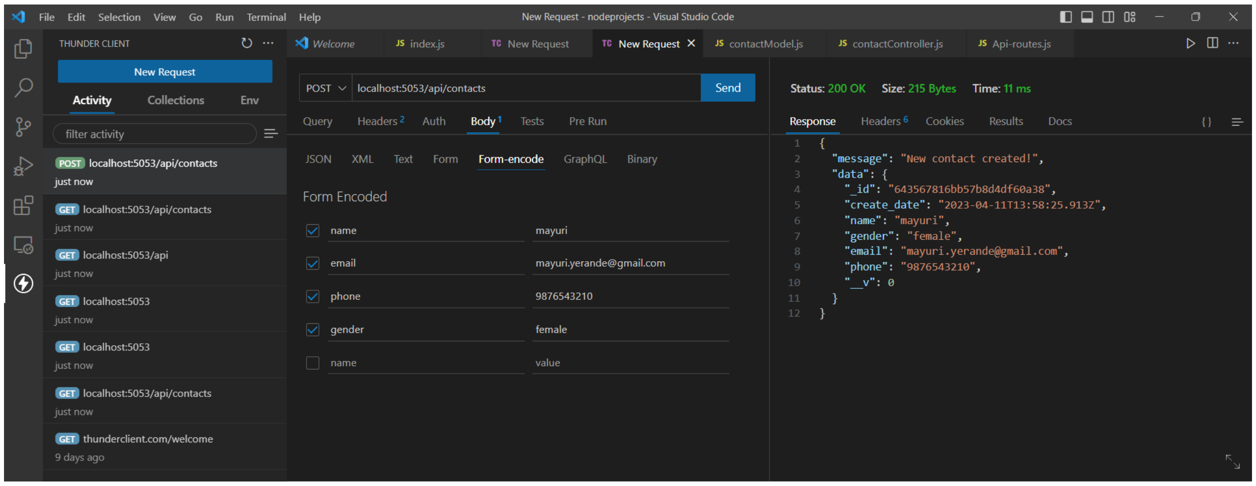
```
{
  "status": "success",
  "message": "Contacts retrieved successfully",
  "data": []
}
```



localhost:5053/api/contacts/

```
{"status":"success","message":"Contacts retrieved successfully","data":[{"_id":"643566ce6bb57b8d4df60a32","create_date":"2023-04-
11T13:55:26.382Z","name":"ghanishtha","gender":"female","email":"ghanishtha.talele09@gmail.com","phone":"9876543210","__v":0},{"_id":"643567396bb57b8d4df60a35","create_date":"2023-04-
11T13:57:13.113Z","name":"aditi","gender":"female","email":"aditi.miniyar@gmail.com","phone":"9876543210","__v":0},{"_id":"643567816bb57b8d4df60a38","create_date":"2023-04-
11T13:58:25.913Z","name":"mayuri","gender":"female","email":"mayuri.yerande@gmail.com","phone":"9876543210","__v":0}]}
```

8. POST A new Value



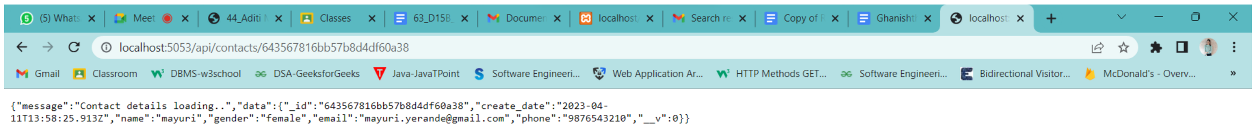
POST localhost:5053/api/contacts

Body: Form-encode

name	mayuri
email	mayuri.yerande@gmail.com
phone	9876543210
gender	female

Status: 200 OK Size: 215 Bytes Time: 11 ms

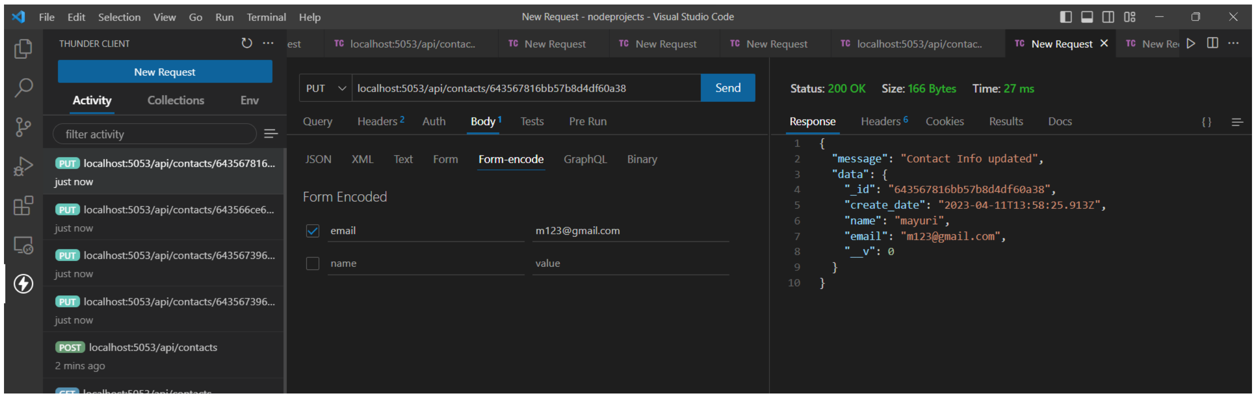
```
{
  "message": "New contact created!",
  "data": {
    "_id": "643567816bb57b8d4df60a38",
    "create_date": "2023-04-11T13:58:25.913Z",
    "name": "mayuri",
    "gender": "female",
    "email": "mayuri.yerande@gmail.com",
    "phone": "9876543210",
    "__v": 0
  }
}
```



localhost:5053/api/contacts/643567816bb57b8d4df60a38

```
{"message":"Contact details loading..","data":{"_id":"643567816bb57b8d4df60a38","create_date":"2023-04-11T13:58:25.913Z","name":"mayuri","gender":"female","email":"mayuri.yerande@gmail.com","phone":"9876543210","__v":0}}
```

9. PUT - Updating email value



PUT localhost:5053/api/contacts/643567816bb57b8d4df60a38

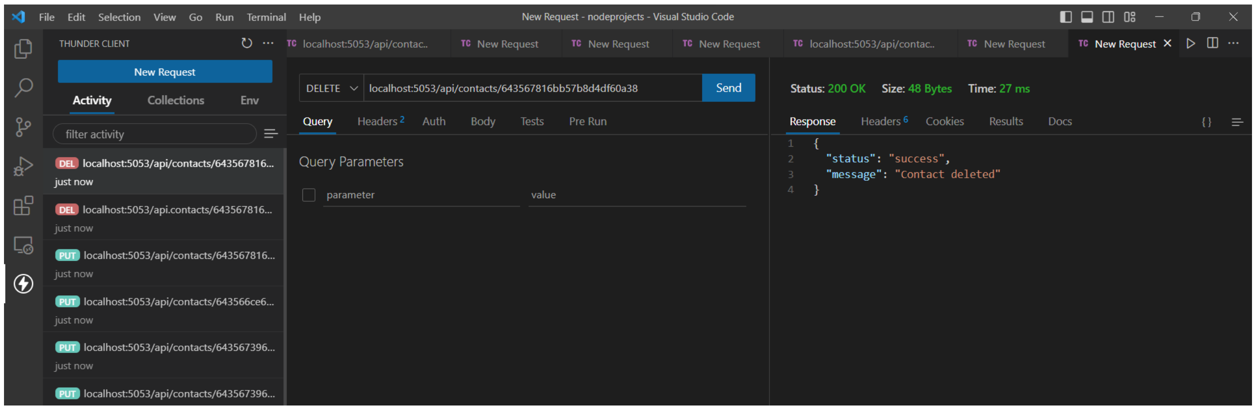
Body: Form-encode

email	m123@gmail.com

Status: 200 OK Size: 166 Bytes Time: 27 ms

```
{
  "message": "Contact Info updated",
  "data": {
    "_id": "643567816bb57b8d4df60a38",
    "create_date": "2023-04-11T13:58:25.913Z",
    "name": "mayuri",
    "email": "m123@gmail.com",
    "__v": 0
  }
}
```

10. DELETE contact



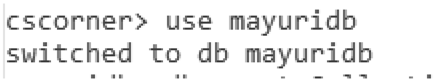
DELETE localhost:5053/api/contacts/643567816bb57b8d4df60a38

Status: 200 OK Size: 48 Bytes Time: 27 ms

```
{
  "status": "success",
  "message": "Contact deleted"
}
```

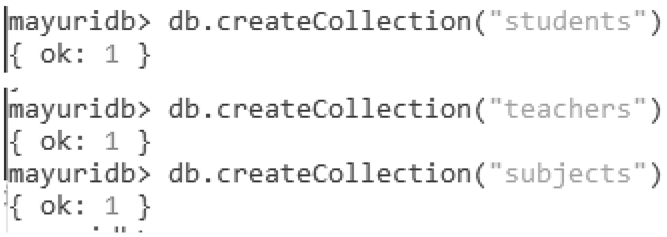
★ DATABASE

- **Creating database**



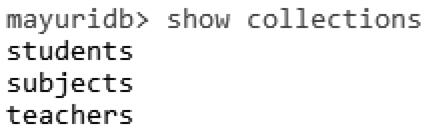
```
cscorner> use mayuridb
switched to db mayuridb
```

- **Creating collections**



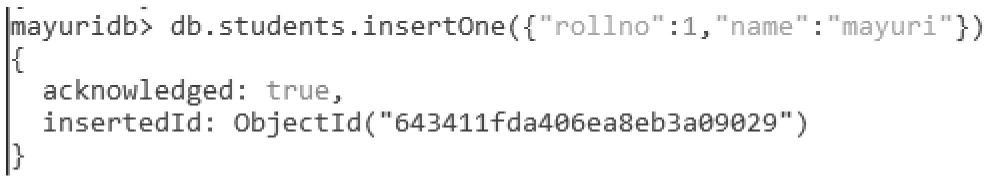
```
mayuridb> db.createCollection("students")
{ ok: 1 }
```

```
mayuridb> db.createCollection("teachers")
{ ok: 1 }
mayuridb> db.createCollection("subjects")
{ ok: 1 }
```



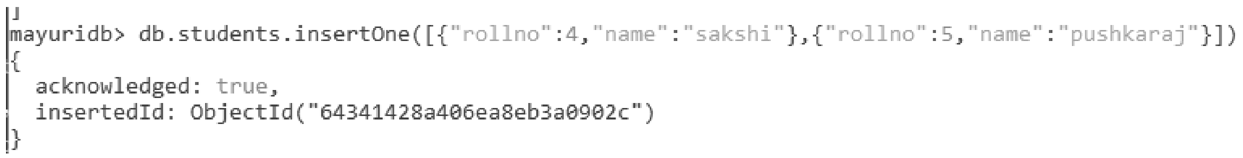
```
mayuridb> show collections
students
subjects
teachers
```

- **Inserting values**
  - ➢ **(Inserting one value at a time)**

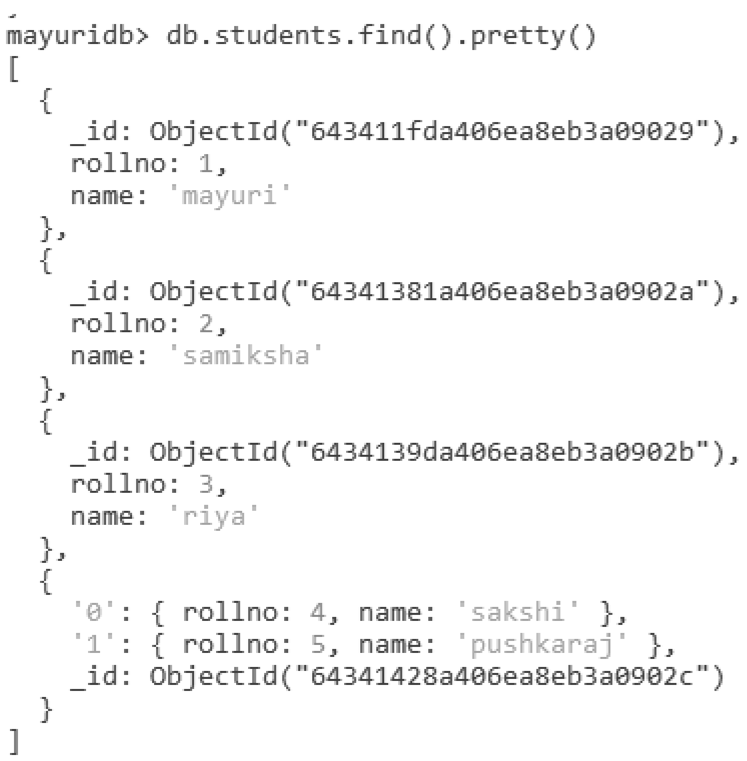


```
mayuridb> db.students.insertOne({"rollno":1,"name":"mayuri"})
{
  acknowledged: true,
  insertedId: ObjectId("643411fda406ea8eb3a09029")
}
```

- ➢ **(Inserting many values at a time)**

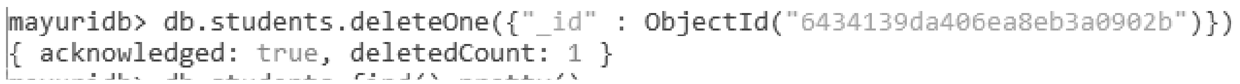


```
mayuridb> db.students.insertOne([{"rollno":4,"name":"sakshi"},{"rollno":5,"name":"pushkaraj"}])
{
  acknowledged: true,
  insertedId: ObjectId("64341428a406ea8eb3a0902c")
}
```



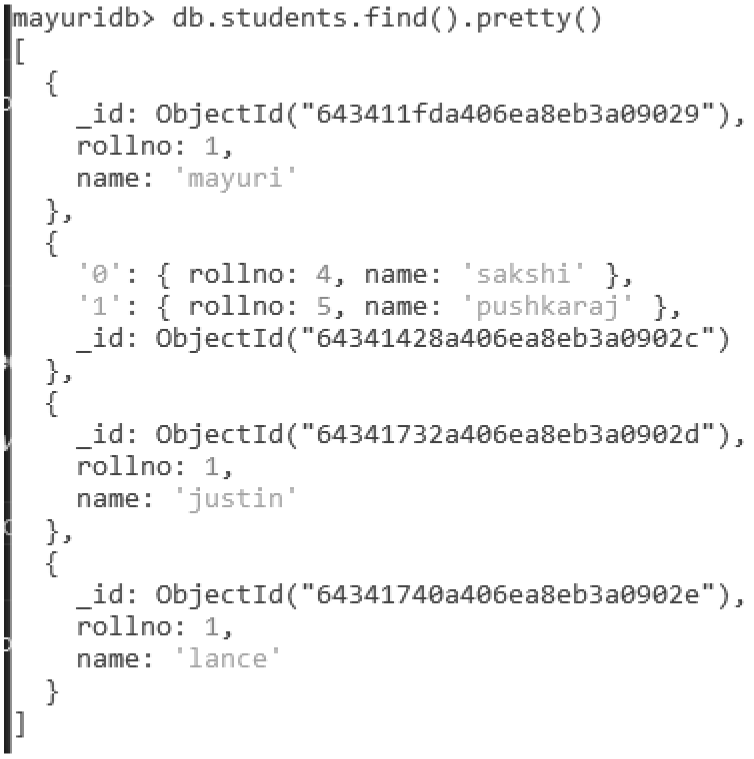
```
mayuridb> db.students.find().pretty()
[
  {
    _id: ObjectId("643411fda406ea8eb3a09029"),
    rollno: 1,
    name: 'mayuri'
  },
  {
    _id: ObjectId("64341381a406ea8eb3a0902a"),
    rollno: 2,
    name: 'samiksha'
  },
  {
    _id: ObjectId("6434139da406ea8eb3a0902b"),
    rollno: 3,
    name: 'riya'
  },
  {
    '0': { rollno: 4, name: 'sakshi' },
    '1': { rollno: 5, name: 'pushkaraj' },
    _id: ObjectId("64341428a406ea8eb3a0902c")
  }
]
```

- **Deleting Values**
  - ➢ **(Deleting one value at a time)**
  - ➢ **(Deleting using object ID of that value)**

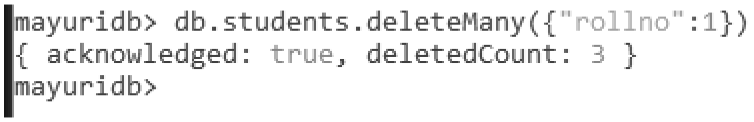


```
mayuridb> db.students.deleteOne({"_id" : ObjectId("6434139da406ea8eb3a0902b")})
{ acknowledged: true, deletedCount: 1 }
```

- ➢ **(Deleting many values at a time)**
- ➢ **There are three roll number 1 hence we will drop them all**

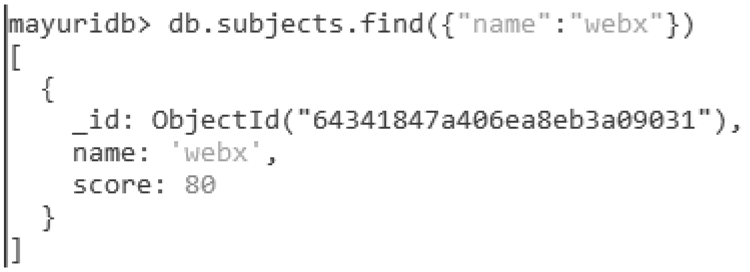


```
mayuridb> db.students.find().pretty()
[
  {
    _id: ObjectId("643411fda406ea8eb3a09029"),
    rollno: 1,
    name: 'mayuri'
  },
  {
    '0': { rollno: 4, name: 'sakshi' },
    '1': { rollno: 5, name: 'pushkaraj' },
    _id: ObjectId("64341428a406ea8eb3a0902c")
  },
  {
    _id: ObjectId("64341732a406ea8eb3a0902d"),
    rollno: 1,
    name: 'justin'
  },
  {
    _id: ObjectId("64341740a406ea8eb3a0902e"),
    rollno: 1,
    name: 'lance'
  }
]
```



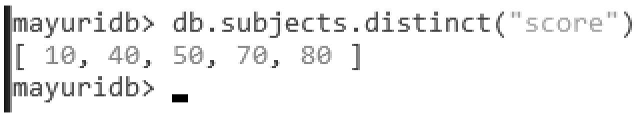
```
mayuridb> db.students.deleteMany({"rollno":1})
{ acknowledged: true, deletedCount: 3 }
mayuridb>
```

- **Search with conditions**
- ➢ **Find**



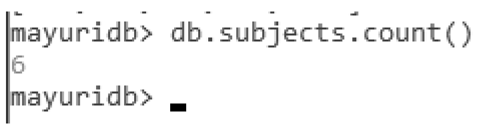
```
mayuridb> db.subjects.find({"name":"webx"})
[
  {
    _id: ObjectId("64341847a406ea8eb3a09031"),
    name: 'webx',
    score: 80
  }
]
```

- ➢ **Distinct**



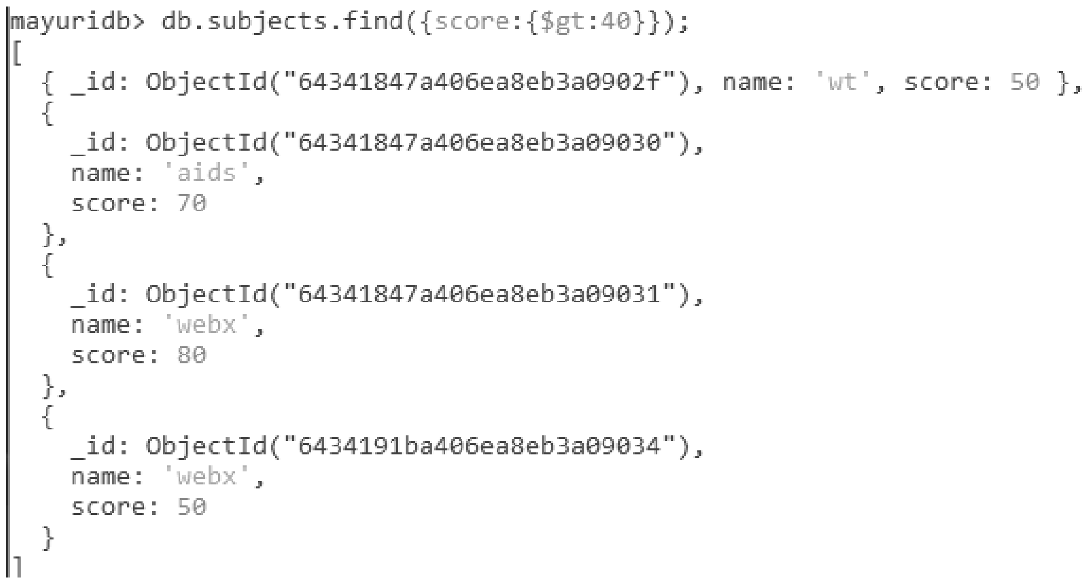
```
mayuridb> db.subjects.distinct("score")
[ 10, 40, 50, 70, 80 ]
mayuridb>
```

- ➢ **Count**



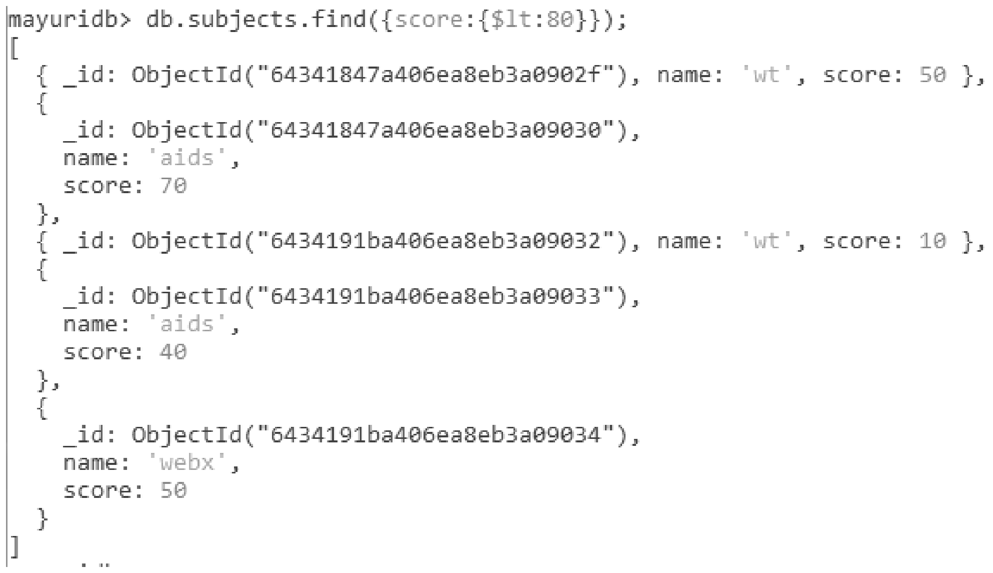
```
mayuridb> db.subjects.count()
6
mayuridb> 
```

➢ **Displaying score of subjects greater than 40**



```
mayuridb> db.subjects.find({score:{$gt:40}});
[
  { _id: ObjectId("64341847a406ea8eb3a0902f"), name: 'wt', score: 50 },
  {
    _id: ObjectId("64341847a406ea8eb3a09030"),
    name: 'aids',
    score: 70
  },
  {
    _id: ObjectId("64341847a406ea8eb3a09031"),
    name: 'webx',
    score: 80
  },
  {
    _id: ObjectId("6434191ba406ea8eb3a09034"),
    name: 'webx',
    score: 50
  }
]
```

➢ **Displaying score of subjects lower than 80**



```
mayuridb> db.subjects.find({score:{$lt:80}});
[
  { _id: ObjectId("64341847a406ea8eb3a0902f"), name: 'wt', score: 50 },
  {
    _id: ObjectId("64341847a406ea8eb3a09030"),
    name: 'aids',
    score: 70
  },
  { _id: ObjectId("6434191ba406ea8eb3a09032"), name: 'wt', score: 10 },
  {
    _id: ObjectId("6434191ba406ea8eb3a09033"),
    name: 'aids',
    score: 40
  },
  {
    _id: ObjectId("6434191ba406ea8eb3a09034"),
    name: 'webx',
    score: 50
  }
]
```

<u>CONCLUSION</u>: Thus we successfully installed MongoDB and Mongo Shell in our system. We implemented the REST API with it. Then we performed CRUD operations on the database in mongo shell.