

Name: Patel Shivam S.
Enroll no: 21162101019
Sem: 5 Batch: 51 Branch: CBA
Sub: Microservices

Practical 8

AIM: Implement an application with CRUD operation using MongoDB with NodeJS.
Symptoms Pvt Ltd company wants to manage their employee records and this task is given to you.
They are seeking functionalities:

Practical 8.1: New joiner data should be stored in DB.
Practical 8.2: The admin can see the whole DB,
Practical 8.3: The admin can filter out the DB.
Practical 8.4: The admin can delete the record from DB.
Practical 8.5: Create a database with the name "student". Create a collection named "cba".
Perform the following tasks:

Insert the details of 10 students with the following details: Name, age, address, phone number, and email.

Filter the students whose age is greater than 16.

Delete the student record whose phone number contains the digit '6' in it.

GITHUB LINK:

https://github.com/Shivam3783/microservice_practicals/tree/main/prac8

Practical 8.1: New joiner data should be stored in DB.

The screenshot shows a development environment with two main windows: VS Code and MongoDB Compass.

VS Code (Top Window):

- Explorer:** Shows files in the workspace, including `demo.js`, `collection.js`, `insert_one.js`, `insert_multiple.js` (selected), and `exam-question`.
- Editor:** Displays the `insert_multiple.js` script:

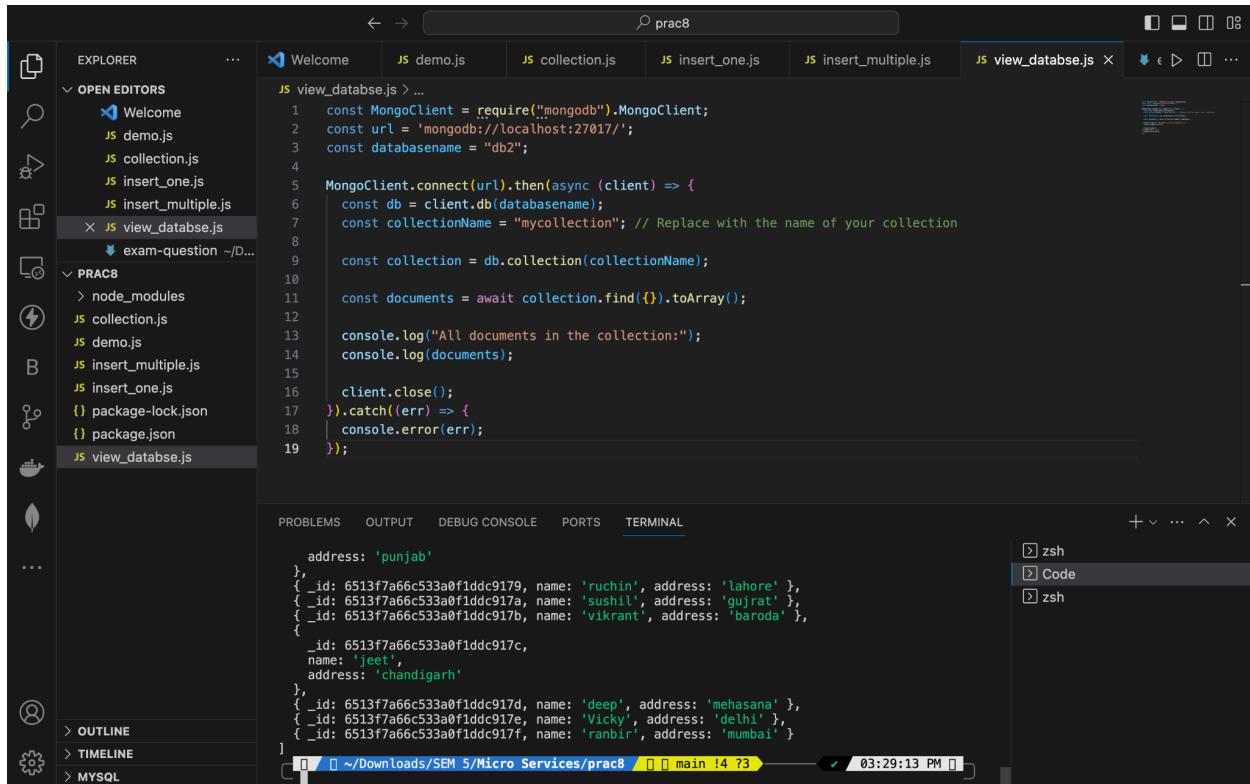
```
1 var MongoClient = require('mongodb').MongoClient;
2 var url = "mongodb://localhost:27017/";
3 MongoClient.connect(url, function(err, db2) {
4   if (err) throw err;
5   var db = db2.db("db2");
6   var empdetails = [
7     { name: 'YASH_GNU', address: 'HARYANA' },
8     { name: 'vikas', address: 'sciencecity' },
9     { name: 'aman', address: 'drivein' },
10    { name: 'mahendra', address: 'punjab' },
11    { name: 'ruchin', address: 'lahore' },
12    { name: 'sushil', address: 'gujrat' },
13    { name: 'vikrant', address: 'baroda' },
14    { name: 'jeet', address: 'chandigarh' },
15    { name: 'deep', address: 'mehasana' },
16    { name: 'Vicky', address: 'delhi' },
17    { name: 'ranbir', address: 'mumbai' },
18  ];
19  dbo.collection("mycollection").insertMany(empdetails, function(err, res) {
20    if (err) throw err;
21    console.log(res);
22    console.log("The count of total employees who's data has been inserted are : " + res.insertedCount);
23    db2.close();
24  });
25 });
26 }
```

MongoDB Compass (Bottom Window):

- Left Sidebar:** Shows databases (`localhost:27017`, `admin`, `config`, `db2`, `local`) and collections (`db2.mycollection` selected).
- Main Area:** Displays the `db2.mycollection` documents. There are 12 documents listed, each with an _id, name, and address.

Document ID	Name	Address
<code>_id: ObjectId('650a9cad3af3954c96601835')</code>	"shivam(cba-19)"	"ganpat university"
<code>_id: ObjectId('650a9e0787a4bc4cdf706ca9')</code>	"YASH_GNU"	"HARYANA"
<code>_id: ObjectId('650a9e0787a4bc4cdf706caa')</code>	"vikas"	"sciencecity"
<code>_id: ObjectId('650a9e0787a4bc4cdf706cab')</code>	"aman"	"drivein"
<code>_id: ObjectId('650a9e0787a4bc4cdf706cac')</code>	"mahendra"	"punjab"

Practical 8.2: The admin can see the whole DB



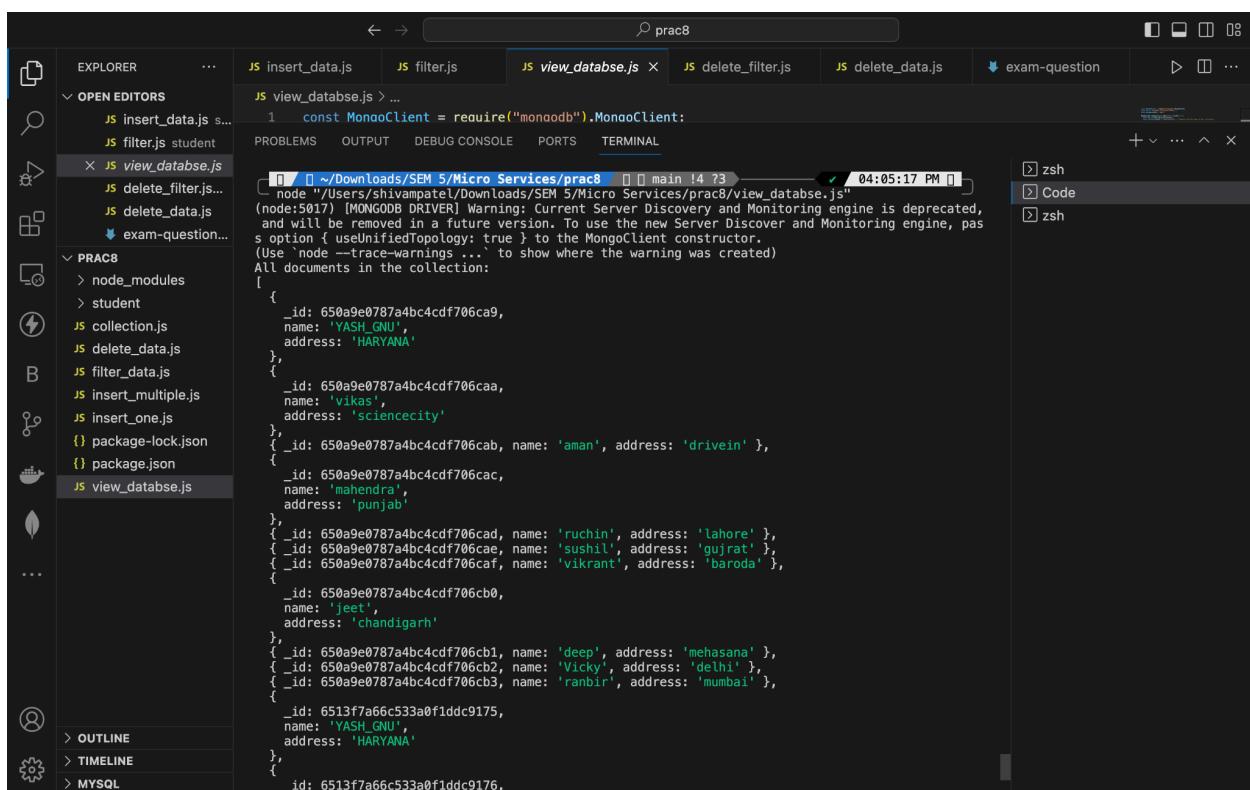
The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the output of a MongoDB script named `view_database.js`. The script connects to a MongoDB database and prints all documents in a collection named `mycollection`.

```

1 const MongoClient = require("mongodb").MongoClient;
2 const url = "mongodb://localhost:27017/";
3 const databaseName = "db2";
4
5 MongoClient.connect(url).then(async (client) => {
6   const db = client.db(databaseName);
7   const collectionName = "mycollection"; // Replace with the name of your collection
8
9   const collection = db.collection(collectionName);
10
11   const documents = await collection.find({}).toArray();
12
13   console.log("All documents in the collection:");
14   console.log(documents);
15
16   client.close();
17 }).catch((err) => {
18   console.error(err);
19 });

```

The terminal also shows the output of the script, which lists several documents with names like `ruchin`, `sushil`, `vikrant`, etc., and addresses like `punjab`, `lahore`, `gujrat`, etc.

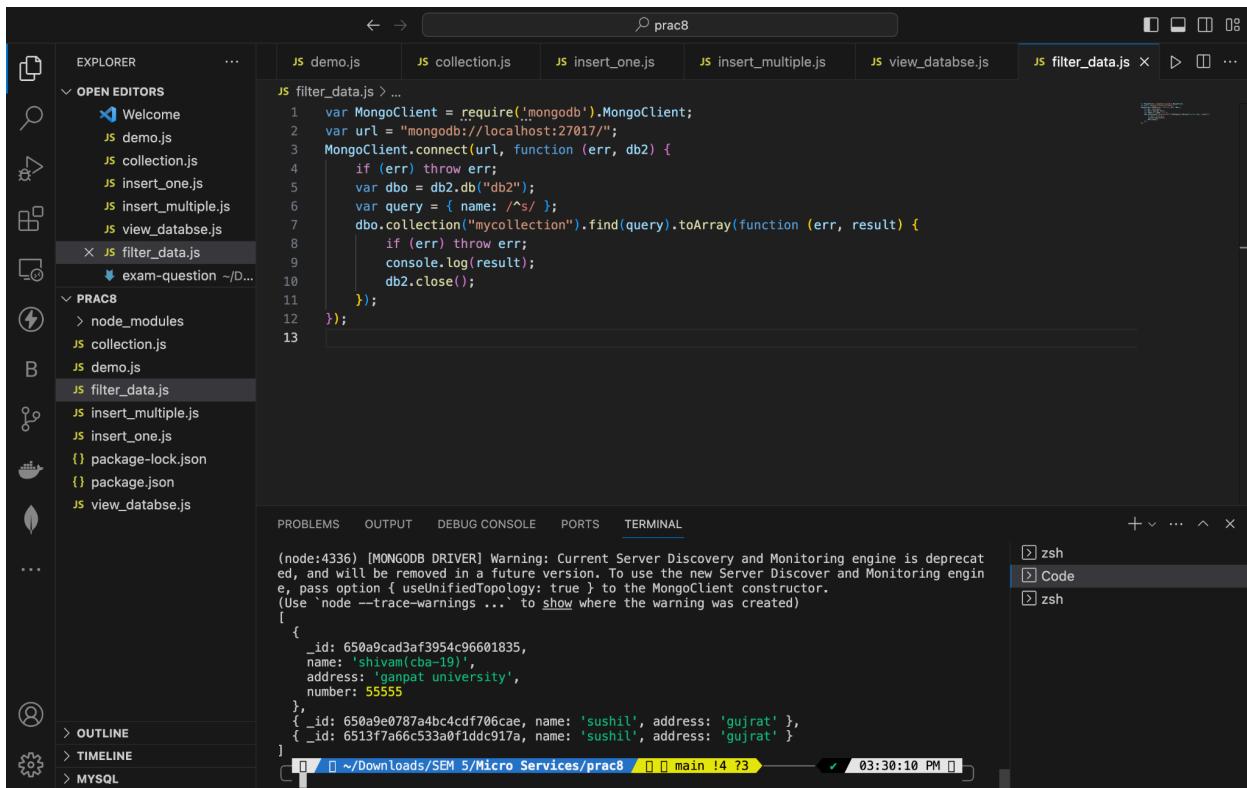


This screenshot shows the same setup as above, but the terminal output includes a warning message from the MongoDB driver:

```
(node:0017) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
```

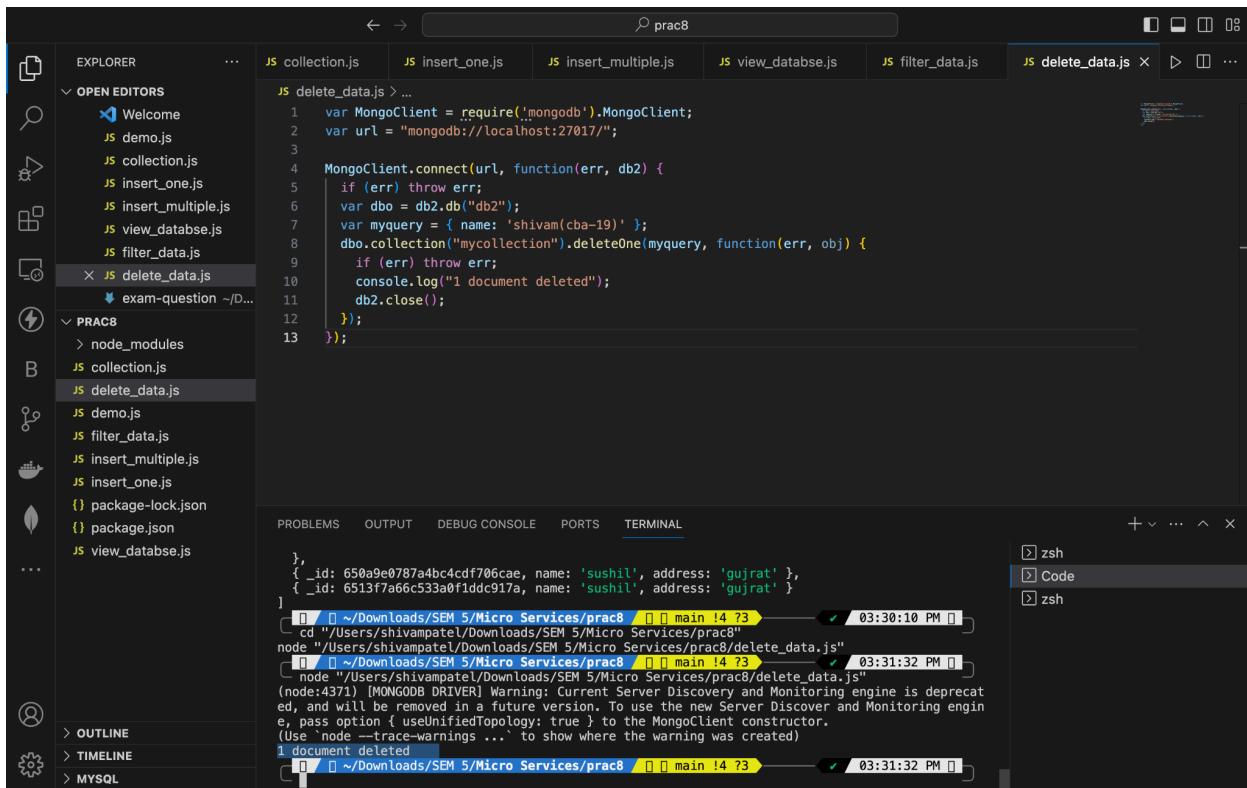
The rest of the terminal output is identical to the first screenshot, showing the list of documents.

Practical 8.3: The admin can filter out the DB.



```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function (err, db2) {
  if (err) throw err;
  var dbo = db2.db("db2");
  var query = { name: /s/ };
  dbo.collection("mycollection").find(query).toArray(function (err, result) {
    if (err) throw err;
    console.log(result);
    db2.close();
  });
});
```

Practical 8.4: The admin can delete the record from DB.



```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db2) {
  if (err) throw err;
  var dbo = db2.db("db2");
  var myquery = { name: 'shivam(cba-19)' };
  dbo.collection("mycollection").deleteOne(myquery, function(err, obj) {
    if (err) throw err;
    console.log("1 document deleted");
    db2.close();
  });
});
```

Practical 8.5: Create a database with the name "student". Create a collection named "cba". Perform the following tasks:

The screenshot shows a code editor interface with several tabs open. The active tab is `JS database.js`. The code in the editor is as follows:

```

1 const MongoClient = require('mongodb').MongoClient;
2 const url = 'mongodb://localhost:27017/';
3 const databaseName = 'student';
4 const collectionName = 'cba';
5
6 MongoClient.connect(url, (err, client) => {
7   if (err) {
8     console.error('Error connecting to MongoDB:', err);
9     return;
10 }
11
12 const db = client.db(databaseName);
13
14 db.createCollection(collectionName, (createErr, collection) => {
15   if (createErr) {
16     console.error('Error creating collection:', createErr);
17   } else {
18     console.log(`Collection "${collectionName}" created successfully in database "${databaseName}"`);
19   }
20
21   client.close();
22 });
23 });

```

Below the code editor, there is a terminal window showing the execution of the script. The terminal output includes:

- `cd student`
- `node database.js`
- (node:4472) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
- (Use 'node --trace-warnings ...' to show where the warning was created)
- `Collection "cba" created successfully in database "student"`

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: admin, config, db2 (with mycollection), local, and student. The student database is selected. Inside the student database, the 'cba' collection is highlighted. The main pane displays the 'student.cba' collection with the following details:

- Documents: 0
- Indexes: 1
- DOCUMENTS INDEXES

The collection view shows a single document icon. Below the document list, it says "This collection has no data". A note states: "It only takes a few seconds to import data from a JSON or CSV file." A green "Import Data" button is visible at the bottom.

→ Insert the details of 10 students with the following details: Name, age, address, phone number, and email.

```

JS insert_data.js x exam-question
student > JS insert_data.js > MongoClient.connect() callback
1 var MongoClient = require('mongodb').MongoClient;
2 var url = "mongodb://localhost:27017/";
3 const databaseName = 'student';
4 const collectionName = 'cba';
5
6 MongoClient.connect(url, function (err, db2) {
7   if (err) throw err;
8   var dbo = db2.db(databaseName);
9   var studentDetails = [
10     {name: 'Student 1', age: 20, address: 'Address 1', phoneNumber: '123-456-7890', email: 'student1@example.com'},
11     {name: 'Student 2', age: 21, address: 'Address 2', phoneNumber: '987-654-3210', email: 'student2@example.com'},
12     {name: 'Student 3', age: 22, address: 'Address 3', phoneNumber: '111-222-3333', email: 'student3@example.com'},
13     {name: 'Student 4', age: 23, address: 'Address 4', phoneNumber: '555-555-5555', email: 'student4@example.com'},
14     {name: 'Student 5', age: 24, address: 'Address 5', phoneNumber: '999-999-9999', email: 'student5@example.com'},
15     {name: 'Student 6', age: 25, address: 'Address 6', phoneNumber: '444-444-4444', email: 'student6@example.com'},
16     {name: 'Student 7', age: 26, address: 'Address 7', phoneNumber: '777-777-7777', email: 'student7@example.com'},
17     {name: 'Student 8', age: 27, address: 'Address 8', phoneNumber: '888-888-8888', email: 'student8@example.com'},
18     {name: 'Student 9', age: 28, address: 'Address 9', phoneNumber: '666-666-6666', email: 'student9@example.com'},
19     {name: 'Student 10', age: 29, address: 'Address 10', phoneNumber: '222-222-2222', email: 'student10@example.com'}
20   ];
21
22   dbo.collection(collectionName).insertMany(studentDetails, function (err, res) {
23     if (err) throw err;
24     console.log(res);
25     console.log("The count of total students whose data has been inserted are : " + res.insertedCount);
26     db2.close();
27   });
}

```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass the option { useUnifiedTopology: true } to the MongoClient constructor.
(Use 'node --trace-warnings ...' to show where the warning was created)

zsh
Code student
zsh

```

JS insert_data.js x exam-question
student > JS insert_data.js > MongoClient.connect() callback
1 var MongoClient = require('mongodb').MongoClient;
2 var url = "mongodb://localhost:27017/";
3 const databaseName = 'student';
4 const collectionName = 'cfa';
5
6 MongoClient.connect(url, function (err, db2) {
7   if (err) throw err;
8   var dbo = db2.db(databaseName);
9   var studentDetails = [
10     {name: 'Student 1', age: 20, address: 'Address 1', phoneNumber: '123-456-7890', email: 'student1@example.com'},
11     {name: 'Student 2', age: 21, address: 'Address 2', phoneNumber: '987-654-3210', email: 'student2@example.com'},
12     {name: 'Student 3', age: 22, address: 'Address 3', phoneNumber: '111-222-3333', email: 'student3@example.com'},
13     {name: 'Student 4', age: 23, address: 'Address 4', phoneNumber: '555-555-5555', email: 'student4@example.com'},
14     {name: 'Student 5', age: 24, address: 'Address 5', phoneNumber: '999-999-9999', email: 'student5@example.com'},
15     {name: 'Student 6', age: 25, address: 'Address 6', phoneNumber: '444-444-4444', email: 'student6@example.com'},
16     {name: 'Student 7', age: 26, address: 'Address 7', phoneNumber: '777-777-7777', email: 'student7@example.com'},
17     {name: 'Student 8', age: 27, address: 'Address 8', phoneNumber: '888-888-8888', email: 'student8@example.com'},
18     {name: 'Student 9', age: 28, address: 'Address 9', phoneNumber: '666-666-6666', email: 'student9@example.com'},
19     {name: 'Student 10', age: 29, address: 'Address 10', phoneNumber: '222-222-2222', email: 'student10@example.com'}
20   ],
21   insertedCount: 10,
22   insertedIds: {
23     '0': 651400b693b39511f0e9b4f3,
24     '1': 651400b693b39511f0e9b4f4,
25     '2': 651400b693b39511f0e9b4f5,
26     '3': 651400b693b39511f0e9b4f6,
27     '4': 651400b693b39511f0e9b4f7,
28     '5': 651400b693b39511f0e9b4f8,
29     '6': 651400b693b39511f0e9b4f9,
30     '7': 651400b693b39511f0e9b4fa,
31     '8': 651400b693b39511f0e9b4fb,
32     '9': 651400b693b39511f0e9b4fc
33   }
34 }

```

The count of total students whose data has been inserted are : 10

zsh
Code student
zsh

localhost:27017 ...

Documents student.cba +

My Queries Databases Search

student.cba

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options >

ADD DATA EXPORT DATA 1 - 10 of 10 < > ⌂ ⌃ ⌄ ⌅

`_id: ObjectId('6514072e58c442143045d900')`
name: "Student 1"
age: 20
address: "Address 1"
phoneNumber: "123-456-7890"
email: "student1@example.com"

`_id: ObjectId('6514072e58c442143045d901')`
name: "Student 2"
age: 21
address: "Address 2"
phoneNumber: "987-654-3210"
email: "student2@example.com" 21

`_id: ObjectId('6514072e58c442143045d902')`
name: "Student 3"
age: 22
address: "Address 3"
phoneNumber: "111-222-3333"
email: "student3@example.com"

`_id: ObjectId('6514072e58c442143045d903')`
name: "Student 4"
age: 23
address: "Address 4"
phoneNumber: "555-555-5555"
email: "student4@example.com"

> MONGOSH ^

→ Filter the students whose age is greater than 16.

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command-line interface for a MongoDB session. The user has run a query to find all students whose age is greater than 16. The output shows one student document matching the criteria.

```
student > JS filter.js > ...
1 var MongoClient = require('mongodb').MongoClient;
2 var url = "mongodb://localhost:27017/";
3 const databaseName = 'student';
4 const collectionName = 'cba';
5
6 MongoClient.connect(url, function (err, db2) {
7   if (err) throw err;
8   var dbo = db2.db(databaseName);
9
10  var query = { age: { $gt: 16 } };
11
12  dbo.collection(collectionName).find(query).toArray(function (err, result) {
13    if (err) throw err;
14    console.log("The students whose age is greater than 16: ");
15    console.log(result);
16    db2.close();
17  });
18});
19
[{"_id": "651400b693b39511f0e9b4f3", "name": "Student 1", "age": 20, "address": "Address 1", "phoneNumber": "123-456-7890", "email": "student1@example.com"}]
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command-line interface for a MongoDB session. The user has run a query to find all students whose age is greater than 16. The output shows five student documents matching the criteria.

```
student > JS filter.js > ...
1 var MongoClient = require('mongodb').MongoClient;
2
3 s option { useUnifiedTopology: true } to the MongoClient constructor.
(Use 'node --trace-warnings ...' to show where the warning was created)
The students whose age is greater than 16:
[
  {"_id": "651400b693b39511f0e9b4f3", "name": "Student 1", "age": 20, "address": "Address 1", "phoneNumber": "123-456-7890", "email": "student1@example.com"}, {"_id": "651400b693b39511f0e9b4f4", "name": "Student 2", "age": 21, "address": "Address 2", "phoneNumber": "987-654-3210", "email": "student2@example.com"}, {"_id": "651400b693b39511f0e9b4f5", "name": "Student 3", "age": 22, "address": "Address 3", "phoneNumber": "111-222-3333", "email": "student3@example.com"}, {"_id": "651400b693b39511f0e9b4f6", "name": "Student 4", "age": 23, "address": "Address 4", "phoneNumber": "555-555-5555", "email": "student4@example.com"}, {"_id": "651400b693b39511f0e9b4f7", "name": "Student 5", "age": 24, "address": "Address 5", "phoneNumber": "999-999-9999", "email": "student5@example.com"}]
```

→ Delete the student record whose phone number contains the digit '6' in it.

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- EXPLORER**: Shows files in the current workspace, including `insert_data.js`, `database.js`, `filter.js`, `delete_filter.js`, `delete_data.js`, and `exam-question...`. A file named `PRAC8` is expanded, showing sub-files like `node_modules`, `student`, `database.js`, `delete_filter.js`, `filter.js`, `insert_data.js`, `collection.js`, `delete_data.js`, `filter_data.js`, `insert_multiple.js`, `insert_one.js`, `package-lock.json`, `package.json`, and `view_database.js`.
- CODE**: The main editor area displays `delete_filter.js` with code for connecting to MongoDB, finding student records by phone number, and deleting them.
- PROBLEMS**: Shows no errors or warnings.
- OUTPUT**: Shows the terminal output of running the script, including a deprecation warning about the MONGOSERVER DRIVER and the deletion of student records.
- DEBUG CONSOLE**: Not visible in the screenshot.
- PORTS**: Not visible in the screenshot.
- TERMINAL**: Shows the command `node "/Users/shivampatel/Downloads/SEM 5/Micro Services/prac8/student/delete_filter.js"` and its output.
- STATUS BAR**: Shows the path `~/Downloads/SEM 5/Micro Services/prac8/student`, the file name `main.l4.js`, the line number `3`, the column number `23`, and the date/time `03:57:01 PM`.

The screenshot shows the MongoDB Compass application interface. The top navigation bar includes the host "localhost:27017", a "Documents" section for "student.cba", and a "+" button. On the left, a sidebar lists databases: "admin", "config", "db2" (with a collection "mycollection"), "local", "student" (with a collection "cba"), and an ellipsis "...". A search bar is also present. The main content area is titled "student.cba" and contains tabs for "Documents", "Aggregations", "Schema", "Indexes", and "Validation". Below these tabs is a search bar with placeholder text "Type a query: { field: 'value' }" and buttons for "Filter", "Explain", "Reset", "Find", and "Options". There are four document cards displayed:

- Document 1:** _id: ObjectId('651400b693b39511f0e9b4f8'), name: "Student 6", age: 25, address: "Address 6", phoneNumber: "444-444-4444", email: "student6@example.com".
- Document 2:** _id: ObjectId('651400b693b39511f0e9b4f9'), name: "Student 7", age: 26, address: "Address 7", phoneNumber: "777-777-7777", email: "student7@example.com".
- Document 3:** _id: ObjectId('651400b693b39511f0e9b4fa'), name: "Student 8", age: 27, address: "Address 8", phoneNumber: "888-888-8888", email: "student8@example.com".
- Document 4:** _id: ObjectId('651400b693b39511f0e9b4fc'), name: "Student 10", age: 29, address: "Address 10", phoneNumber: "222-222-2222", email: "student10@example.com".

On the right side of the document cards are edit, delete, and other action icons. The bottom right corner shows statistics: 0 DOCUMENTS and 1 INDEXES.