Lab Session 11: MongoDB – Databases, Collections and Records

Date of the Session: ----/-----/------ **Time of the Session: to**

Program Title: MongoDB – Databases, Collections and Records

Pre-Lab Task:
Answer the following question before entering into lab.

1. **How do find(), limit(), and sort() queries work in MongoDB?**

2. **What is an index in MongoDB, and why is it useful?**

3. **What are the common aggregation stages in MongoDB?**

4. **What is the purpose of db.createCollection() if MongoDB can create collections automatically?**

5. **How can you update multiple documents at once?**

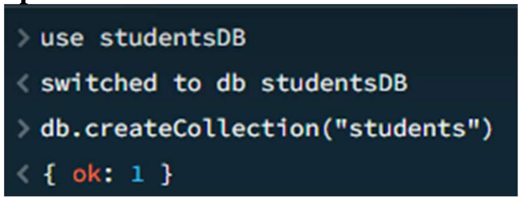
In Lab Task

A. Write MongoDB queries to Create and drop databases and collections.

Create:

```
use studentDB
db.createCollection("students")
```

Output:

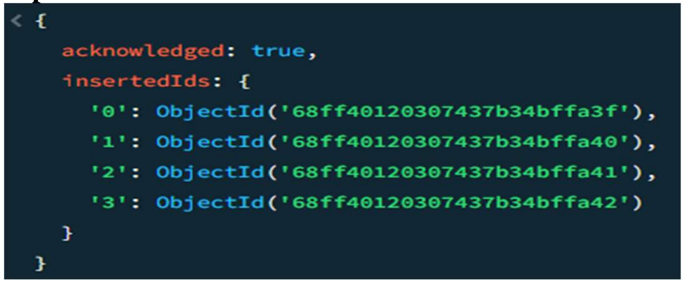


```
> use studentsDB
< switched to db studentsDB
> db.createCollection("students")
< { ok: 1 }
```

Insert into collection:

```
db.students.insertOne({
  name: "Meera",
  age: 20,
  department: "AI&DS"
})
db.students.insertMany([
  { name: "Meera", age: 20, department: "AI&DS" },
  { name: "Karan", age: 24, department: "IT" },
  { name: "Divya", age: 22, department: "ECE" },
  { name: "Sameer", age: 19, department: "CSE" }
])
```

Output:

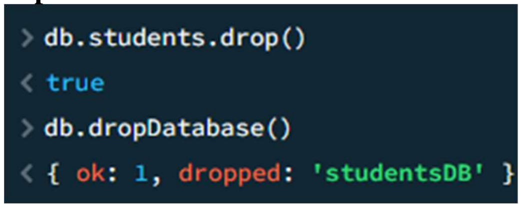


```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68ff40120307437b34bffa3f'),
      '1': ObjectId('68ff40120307437b34bffa40'),
      '2': ObjectId('68ff40120307437b34bffa41'),
      '3': ObjectId('68ff40120307437b34bffa42')
    }
  }
```

Drop:

```
db.students.drop()
db.dropDatabase()
```

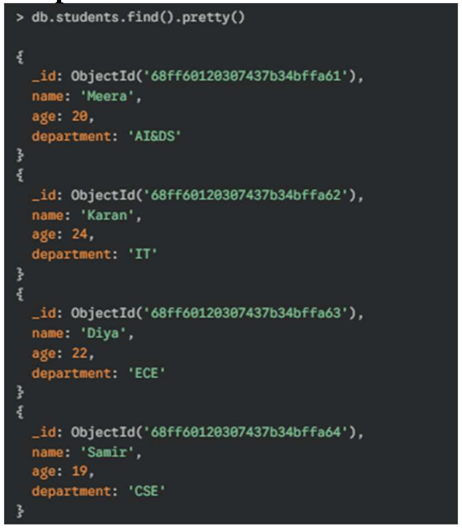
Output:



```
> db.students.drop()
< true
> db.dropDatabase()
< { ok: 1, dropped: 'studentsDB' }
```

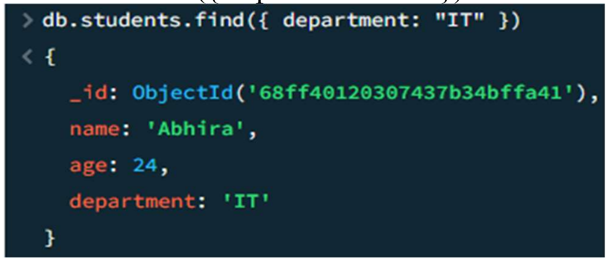
B. Write MongoDB queries to work with records using find(), limit(), sort(), createIndex(), aggregate().

i) find():
db.students.find().pretty()
Output:



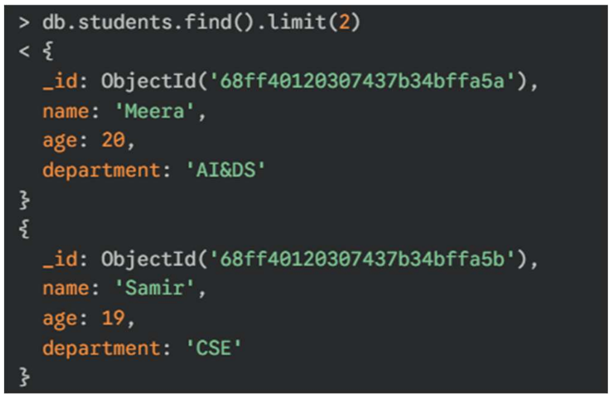
```
> db.students.find().pretty()

{
  _id: ObjectId('68ff60120307437b34bffa61'),
  name: 'Meera',
  age: 20,
  department: 'AI&DS'
}
{
  _id: ObjectId('68ff60120307437b34bffa62'),
  name: 'Karan',
  age: 24,
  department: 'IT'
}
{
  _id: ObjectId('68ff60120307437b34bffa63'),
  name: 'Diya',
  age: 22,
  department: 'ECE'
}
{
  _id: ObjectId('68ff60120307437b34bffa64'),
  name: 'Samir',
  age: 19,
  department: 'CSE'
}
```

db.students.find({ department: "IT" })



```
> db.students.find({ department: "IT" })
< {
    _id: ObjectId('68ff40120307437b34bffa41'),
    name: 'Abhira',
    age: 24,
    department: 'IT'
  }
```

ii) **limit():**
db.students.find().limit(2)
Output:

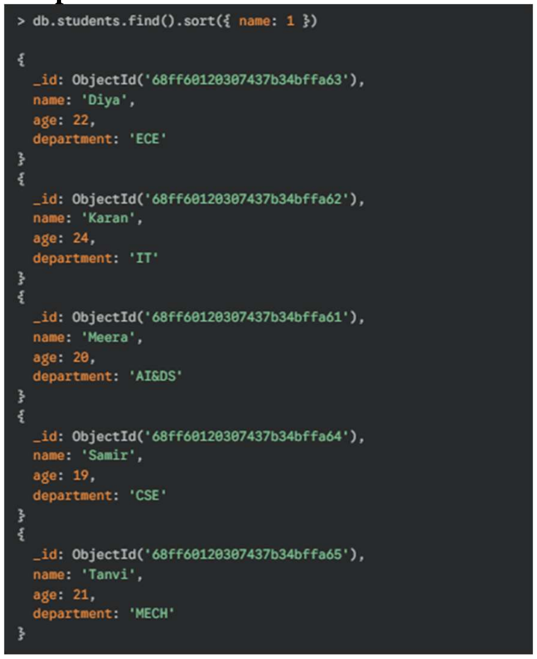


```
> db.students.find().limit(2)
< {
    _id: ObjectId('68ff40120307437b34bffa5a'),
    name: 'Meera',
    age: 20,
    department: 'AI&DS'
  }
  {
    _id: ObjectId('68ff40120307437b34bffa5b'),
    name: 'Samir',
    age: 19,
    department: 'CSE'
  }
```

iii) **sort():**

db.students.find().sort({ name: 1 })

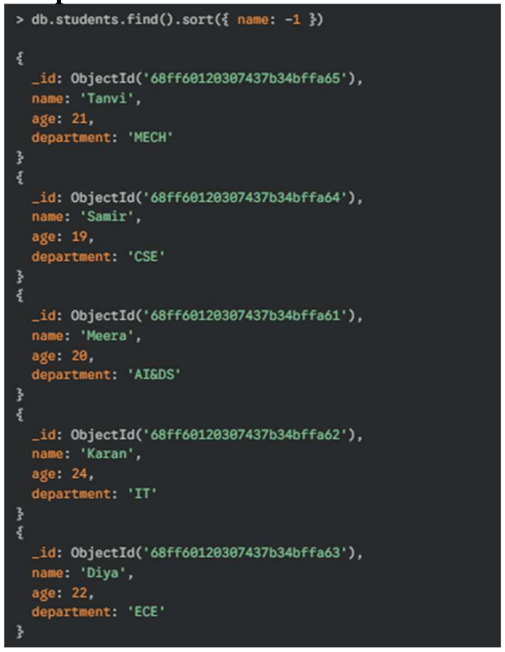
Output:



```
> db.students.find().sort({ name: 1 })

{
  _id: ObjectId('68ff60120307437b34bffa63'),
  name: 'Diya',
  age: 22,
  department: 'ECE'
}
{
  _id: ObjectId('68ff60120307437b34bffa62'),
  name: 'Karan',
  age: 24,
  department: 'IT'
}
{
  _id: ObjectId('68ff60120307437b34bffa61'),
  name: 'Meera',
  age: 20,
  department: 'AI&DS'
}
{
  _id: ObjectId('68ff60120307437b34bffa64'),
  name: 'Samir',
  age: 19,
  department: 'CSE'
}
{
  _id: ObjectId('68ff60120307437b34bffa65'),
  name: 'Tanvi',
  age: 21,
  department: 'MECH'
}
```

db.students.find().sort({ age: -1 })

Output:

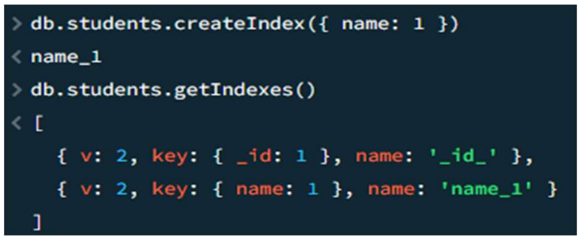


```
> db.students.find().sort({ name: -1 })

{
  _id: ObjectId('68ff60120307437b34bffa65'),
  name: 'Tanvi',
  age: 21,
  department: 'MECH'
}
{
  _id: ObjectId('68ff60120307437b34bffa64'),
  name: 'Samir',
  age: 19,
  department: 'CSE'
}
{
  _id: ObjectId('68ff60120307437b34bffa61'),
  name: 'Meera',
  age: 20,
  department: 'AI&DS'
}
{
  _id: ObjectId('68ff60120307437b34bffa62'),
  name: 'Karan',
  age: 24,
  department: 'IT'
}
{
  _id: ObjectId('68ff60120307437b34bffa63'),
  name: 'Diya',
  age: 22,
  department: 'ECE'
}
```

iv) **createIndex():**

db.students.createIndex({ name: 1 })
db.students.getIndexes()

Output:

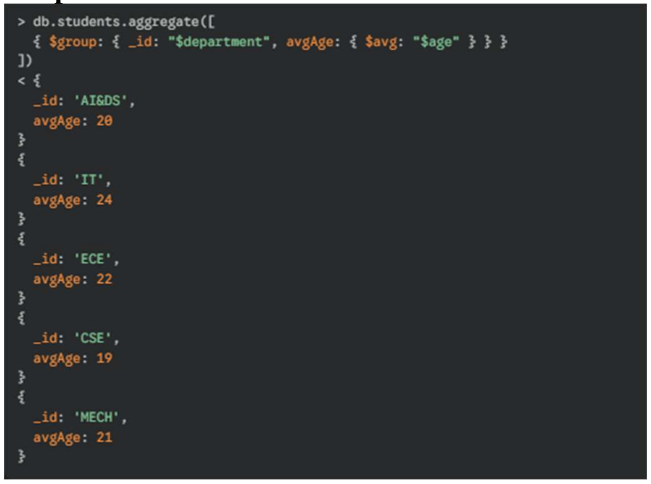


```
> db.students.createIndex({ name: 1 })
< name_1
> db.students.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { name: 1 }, name: 'name_1' }
  ]
```

v) **aggregate():**

```
db.students.aggregate([
  { $group: { _id: "$department", avgAge: { $avg: "$age" } } }
])
```

Output:



```
> db.students.aggregate([
    { $group: { _id: "$department", avgAge: { $avg: "$age" } } }
  ])
< {
    _id: 'AI&DS',
    avgAge: 20
  }
  {
    _id: 'IT',
    avgAge: 24
  }
  {
    _id: 'ECE',
    avgAge: 22
  }
  {
    _id: 'CSE',
    avgAge: 19
  }
  {
    _id: 'MECH',
    avgAge: 21
  }
```

Post Lab Questions:

Employee Management System

Description: Manage employee records for a company.

Tasks:

- Store employee info: name, age, department, salary.
- CRUD operations using REST API.
- Implement a route to calculate average salary per department.
- Sort employees by salary or age.

models/Employee.js:

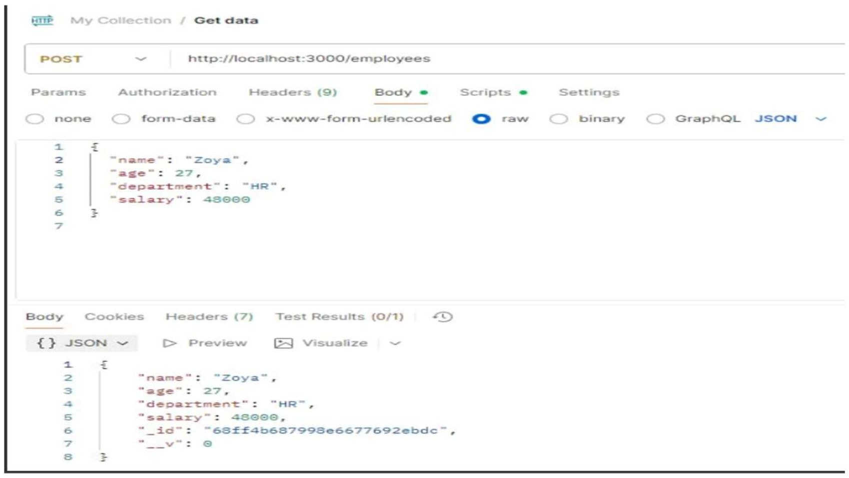
```
const mongoose = require('mongoose');
const employeeSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number, required: true },
  department: { type: String, required: true },
  salary: { type: Number, required: true },
});
module.exports = mongoose.model('Employee', employeeSchema);
```

index.js:

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const app = express();
app.use(bodyParser.json());
mongoose.connect("mongodb://localhost:27017/employeeDB", {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
```

```
const employeeSchema = new mongoose.Schema({
  name: String,
  age: Number,
  department: String,
  salary: Number
});
const Employee = mongoose.model("Employee", employeeSchema);
app.post("/employees", async (req, res) => {
  const emp = new Employee(req.body);
  await emp.save();
  res.send(emp);
});
app.get("/employees", async (req, res) => {
  const employees = await Employee.find();
  res.send(employees);
});
app.put("/employees/:id", async (req, res) => {
  const emp = await Employee.findByIdAndUpdate(req.params.id, req.body, { new: true });
  res.send(emp);
});
app.delete("/employees/:id", async (req, res) => {
  await Employee.findByIdAndDelete(req.params.id);
  res.send({ message: "Employee deleted" });
});
app.get("/employees/average-salary", async (req, res) => {
  const result = await Employee.aggregate([
    { $group: { _id: "$department", avgSalary: { $avg: "$salary" } } }
  ]);
  res.send(result);
});
app.get("/employees/sort", async (req, res) => {
  const { by, order } = req.query;
  const sortOrder = order === "desc" ? -1 : 1;
  const employees = await Employee.find().sort({ [by]: sortOrder });
  res.send(employees);
});
app.listen(3000, () => console.log("Server running on port 3000"));
```

POST: http://localhost:3000/employees
Output:



POST http://localhost:3000/employees

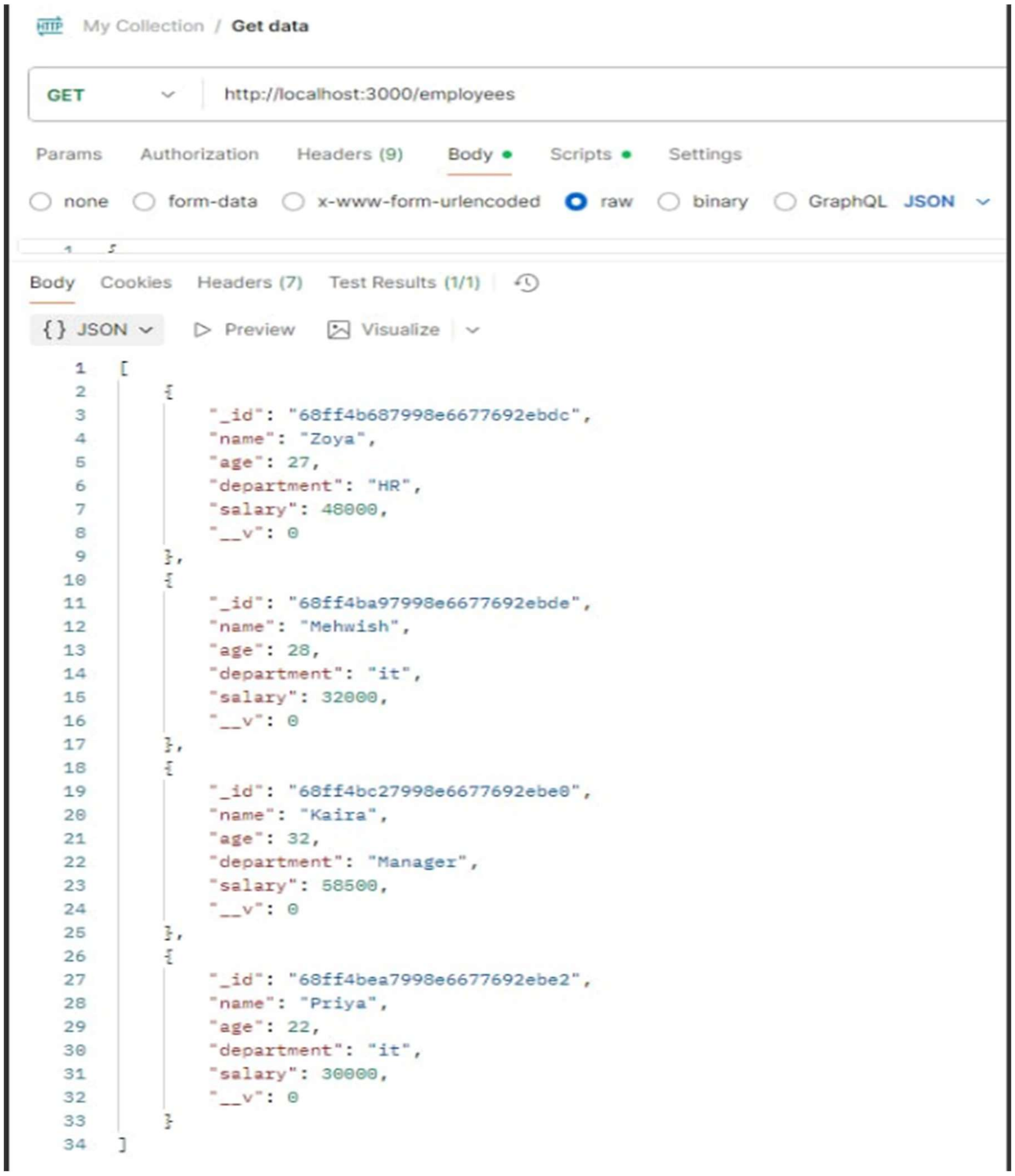
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
  "name": "Zoya",
  "age": 27,
  "department": "HR",
  "salary": 48000
}
```

```
{
    "name": "Zoya",
    "age": 27,
    "department": "HR",
    "salary": 48000,
    "_id": "68ff4b687998e6677692ebdc",
    "__v": 0
}
```

GET: http://localhost:3000/employees

Output:


My Collection / Get data

GET http://localhost:3000/employees

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

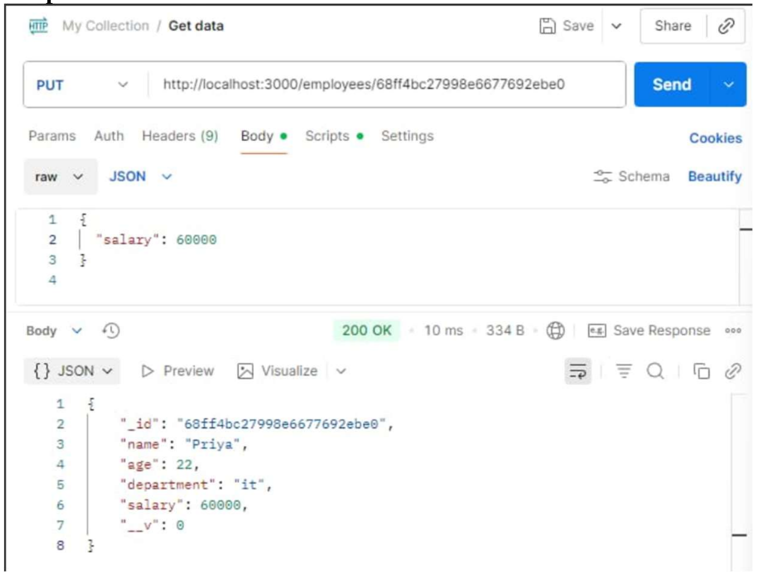
Body Cookies Headers (7) Test Results (1/1)

{} JSON Preview Visualize

```
[
    {
        "_id": "68ff4b687998e6677692ebdc",
        "name": "Zoya",
        "age": 27,
        "department": "HR",
        "salary": 48000,
        "__v": 0
    },
    {
        "_id": "68ff4ba97998e6677692ebde",
        "name": "Mehwish",
        "age": 28,
        "department": "it",
        "salary": 32000,
        "__v": 0
    },
    {
        "_id": "68ff4bc27998e6677692ebe0",
        "name": "Kaira",
        "age": 32,
        "department": "Manager",
        "salary": 58500,
        "__v": 0
    },
    {
        "_id": "68ff4bea7998e6677692ebe2",
        "name": "Priya",
        "age": 22,
        "department": "it",
        "salary": 30000,
        "__v": 0
    }
]
```

PUT: http://localhost:3000/employees/:68ff4bc27998e6677692ebe0d

Output:



PUT http://localhost:3000/employees/68ff4bc27998e6677692ebe0

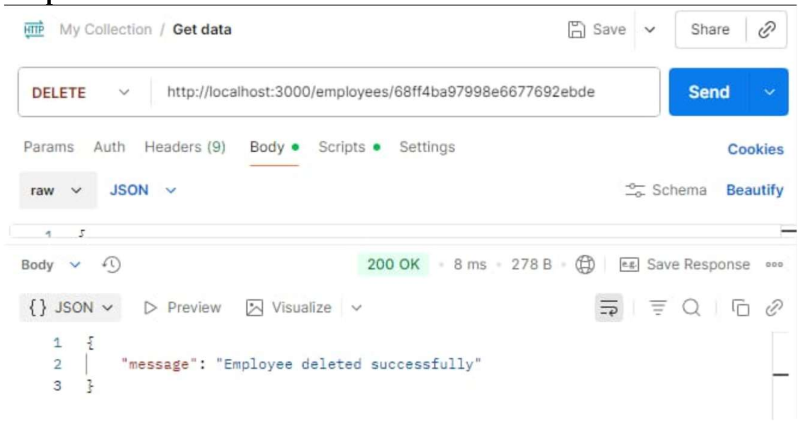
```
{
  "salary": 60000
}
```

200 OK 10 ms 334 B

```
{
    "_id": "68ff4bc27998e6677692ebe0",
    "name": "Priya",
    "age": 22,
    "department": "it",
    "salary": 60000,
    "__v": 0
}
```

DELETE: http://localhost:3000/employees/:68ff4ba97998e667792ebde

Output:



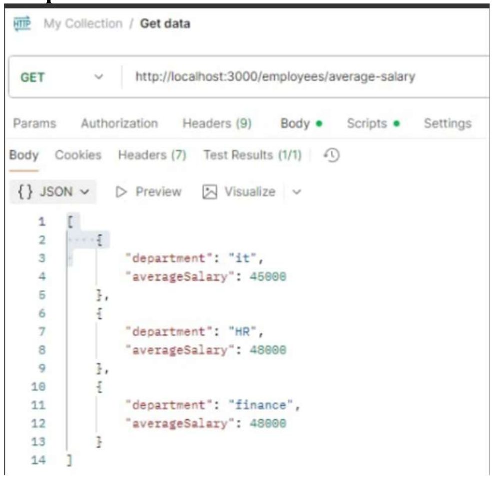
DELETE http://localhost:3000/employees/68ff4ba97998e6677692ebde

200 OK 8 ms 278 B

```
{
    "message": "Employee deleted successfully"
}
```

Route to calculate average salary:

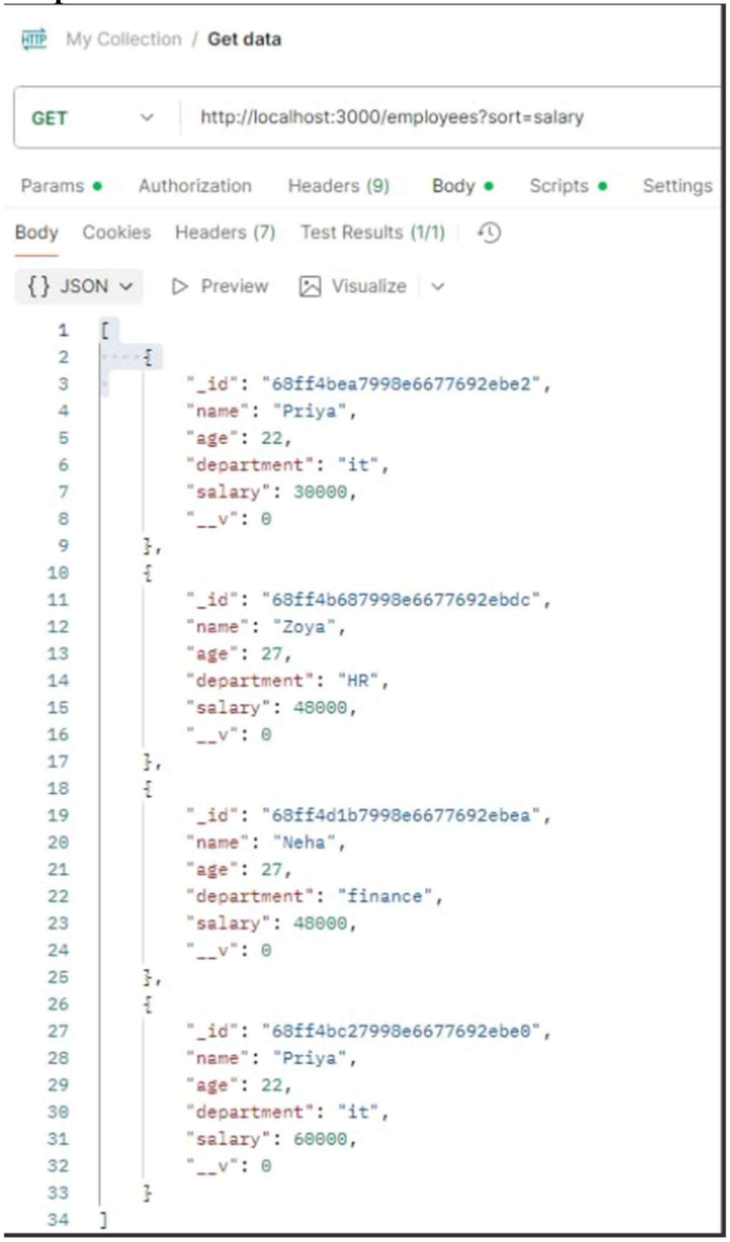
Output:



GET http://localhost:3000/employees/average-salary

```
[
    {
        "department": "it",
        "averageSalary": 45000
    },
    {
        "department": "HR",
        "averageSalary": 48000
    },
    {
        "department": "finance",
        "averageSalary": 48000
    }
]
```

Sort employees by salary:
Output:



My Collection / Get data

GET http://localhost:3000/employees?sort=salary

Params Authorization Headers (9) Body Scripts Settings

Body Cookies Headers (7) Test Results (1/1)

{} JSON Preview Visualize

```
[
    {
        "_id": "68ff4bea7998e6677692ebe2",
        "name": "Priya",
        "age": 22,
        "department": "it",
        "salary": 30000,
        "__v": 0
    },
    {
        "_id": "68ff4b687998e6677692ebdc",
        "name": "Zoya",
        "age": 27,
        "department": "HR",
        "salary": 48000,
        "__v": 0
    },
    {
        "_id": "68ff4d1b7998e6677692ebea",
        "name": "Neha",
        "age": 27,
        "department": "finance",
        "salary": 48000,
        "__v": 0
    },
    {
        "_id": "68ff4bc27998e6677692ebe0",
        "name": "Priya",
        "age": 22,
        "department": "it",
        "salary": 60000,
        "__v": 0
    }
]
```

(For Evaluator's use only)	
Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:________ out of __________ Signature of the Evaluator