

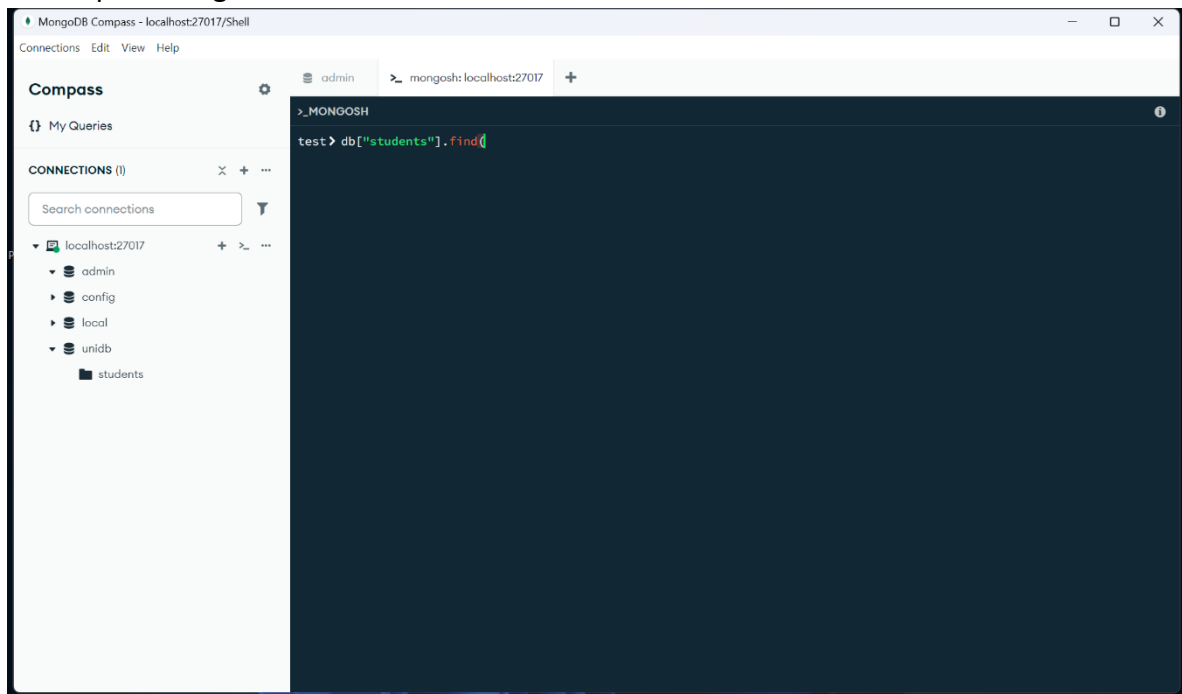
IT2234 (p) Web Service and Server Technologies.

University of Vavuniya
Faculty of Applied science
Department of physical science
Tutorial – 01 2021/ICT/84

Date: 2024.04.21

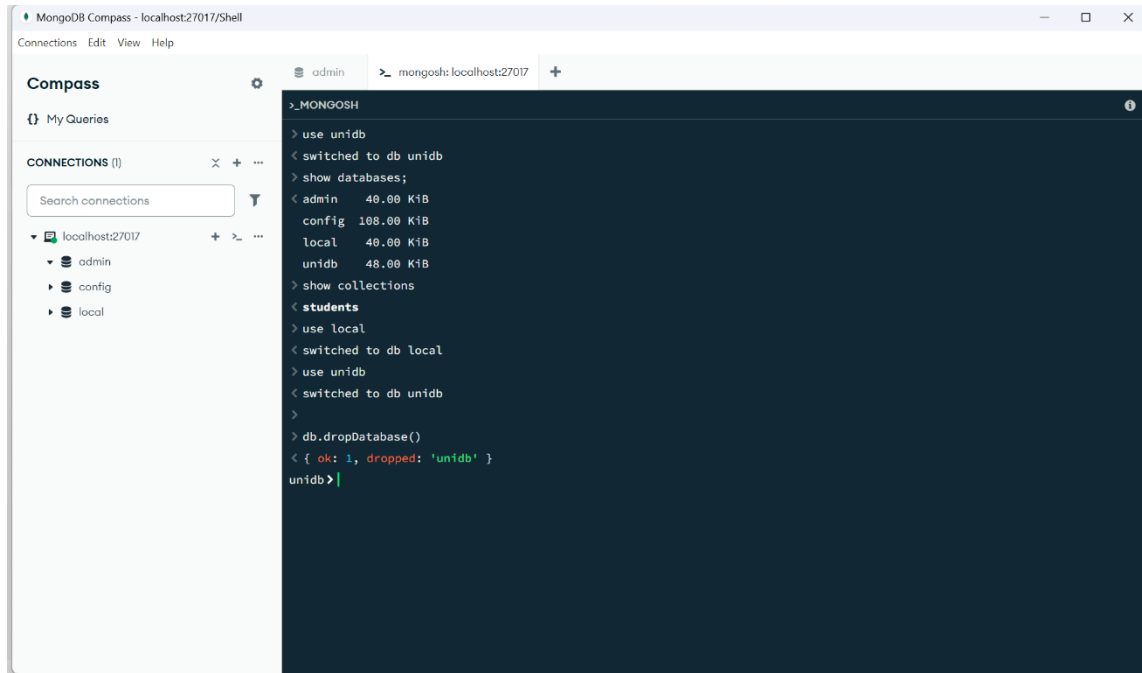
1. How to install and use MongoDB.

- Go to <https://www.mongodb.com/try/download/community>
 - Select your OS and download the MongoDB Community Server.
 - Run the installer and follow these steps:
 - Choose Complete installation.
 - Tick Install MongoDB Compass (GUI tool – optional but helpful).
 - Click Install.
 - During installation, enable the option "Add MongoDB to PATH environment variable".
 - If missed, manually add C:\Program Files\MongoDB\Server\bin to system PATH.
1. Open MongoDB Compass from the Start Menu. MongoDB server will start and wait for connections on `mongodb://localhost:27017`.
 2. Click add new connection, create new database (unldb) and create collection (students)
 3. Open MongoDB shell



2. Write the query to view the databases and collections in MongoDB shell.

- Database command
 - show databases - List all databases
 - use unidb - Switch to or create a database
 - db.dropDatabase() - Delete current database
- Collection command
 - show collections - List all collections in current database
 - db.createCollection("myCol") - Create a new collection
 - db.myCol.drop() - Drop a collection

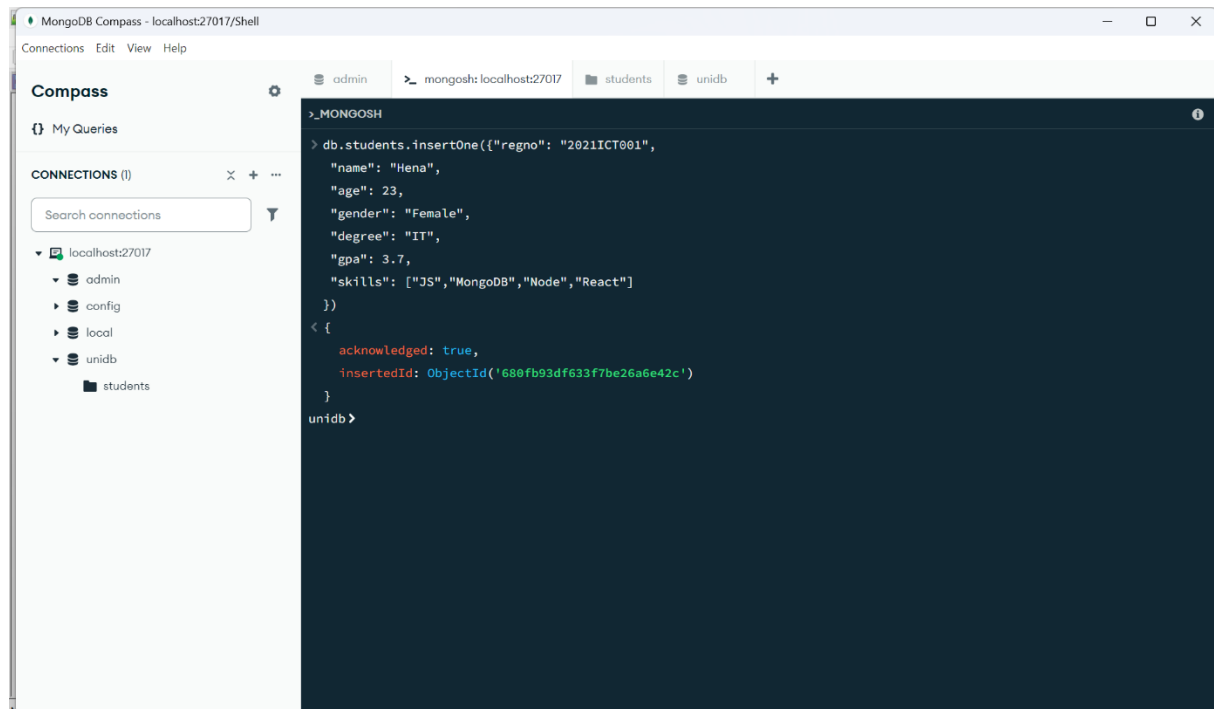


The screenshot shows the MongoDB Compass application window. On the left, the 'Connections' panel shows a connection to 'localhost:27017' with databases 'admin', 'config', and 'local'. The main panel displays the MongoDB shell output for the following commands:

```
> use unidb
< switched to db unidb
> show databases;
< admin 48.00 KiB
  config 108.00 KiB
  local 40.00 KiB
  unidb 48.00 KiB
> show collections
< students
> use local
< switched to db local
> use unidb
< switched to db unidb
>
> db.dropDatabase()
< { ok: 1, dropped: 'unidb' }
unidb>
```

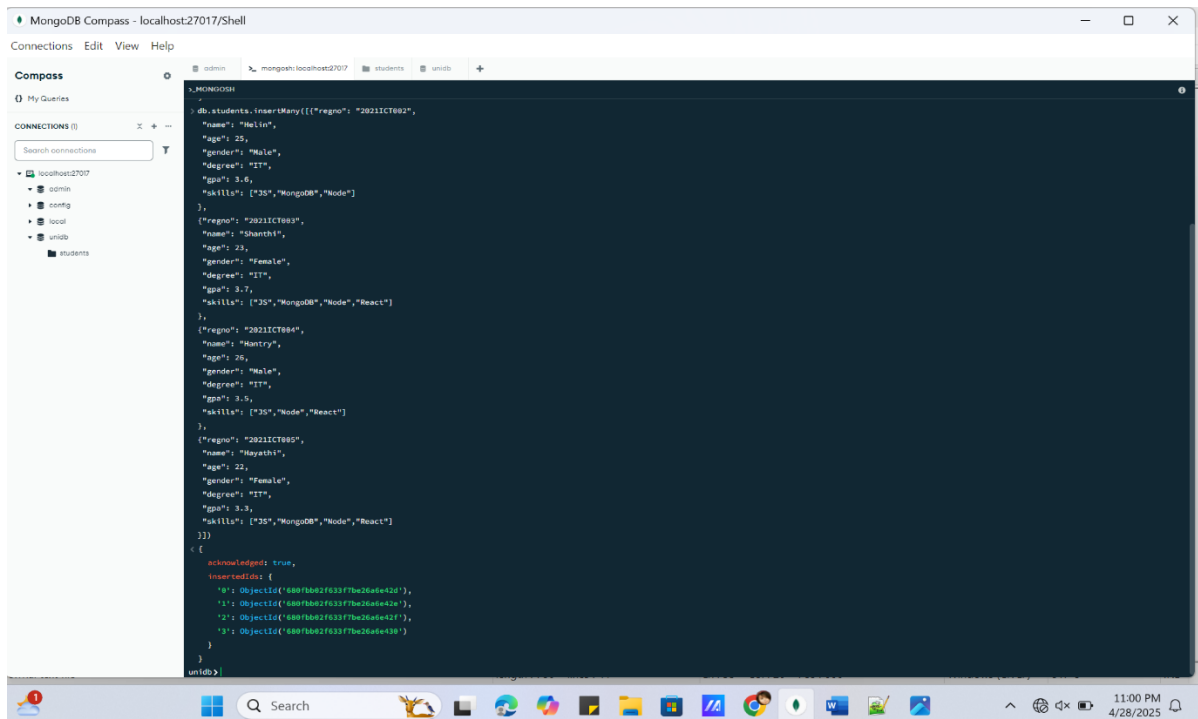
3. Write the query to insert documents into collection.

- insertOne() – inserts a single document at a time.
- insertMany() – inserts multiple documents at a time.



The screenshot shows the MongoDB Compass interface. The left sidebar displays the 'connections' list with 'localhost:27017' expanded, showing databases 'admin', 'config', 'local', and 'unldb', and the 'students' collection under 'unldb'. The main panel shows the MongoDB Shell with the following command and output:

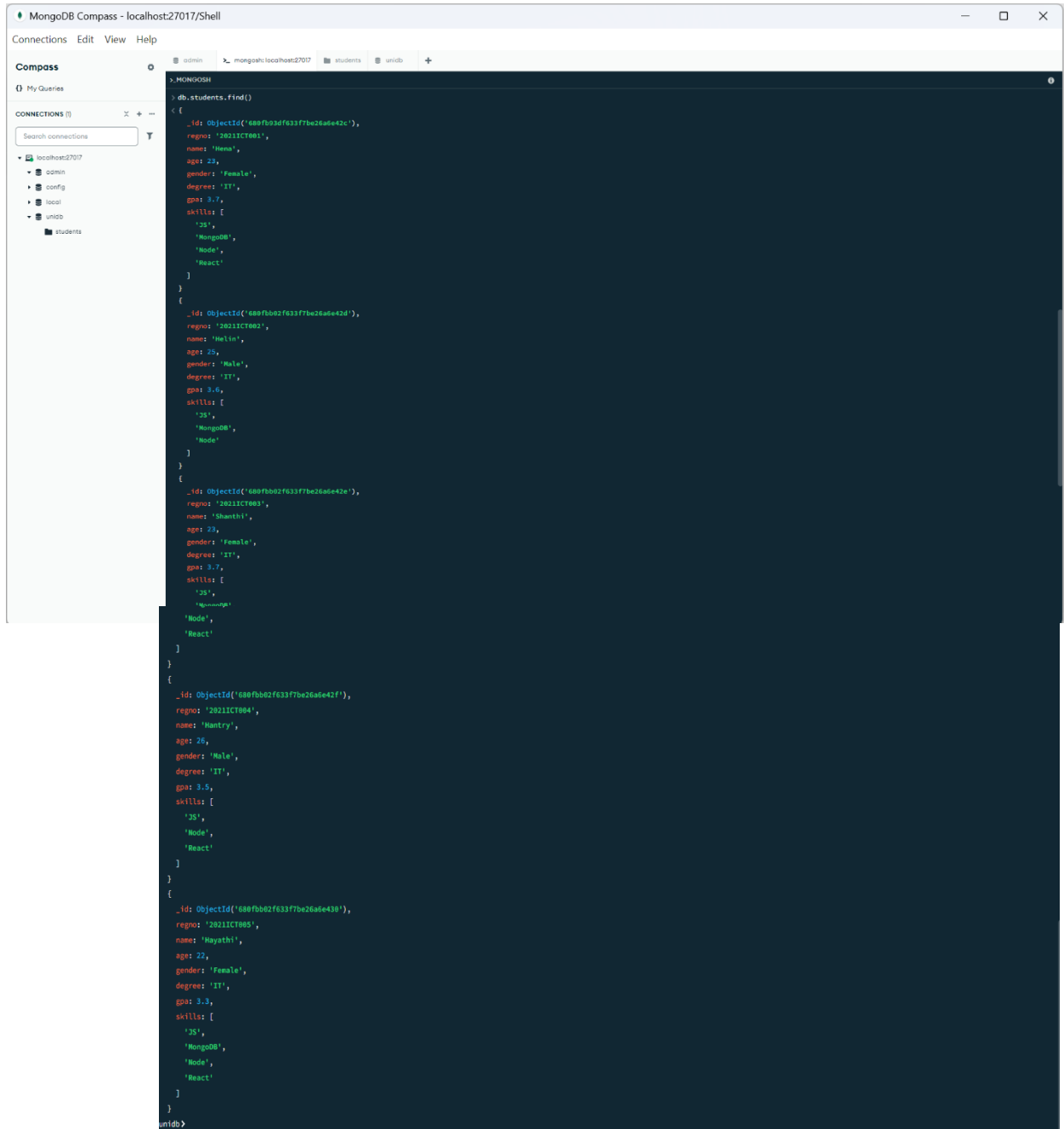
```
>_MONGOSH
> db.students.insertOne({"regno": "2021ICT001",
  "name": "Hena",
  "age": 23,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.7,
  "skills": ["JS","MongoDB","Node","React"]
})
{
  acknowledged: true,
  insertedId: ObjectId('600fb93df633f7be26a6e42c')
}
unldb>
```



The screenshot shows the MongoDB Compass interface. The left sidebar displays the 'connections' list with 'localhost:27017' expanded, showing databases 'admin', 'config', 'local', and 'unldb', and the 'students' collection under 'unldb'. The main panel shows the MongoDB Shell with the following command and output:

```
>_MONGOSH
> db.students.insertMany([{"regno": "2021ICT002",
  "name": "Malin",
  "age": 25,
  "gender": "Male",
  "degree": "IT",
  "gpa": 3.6,
  "skills": ["JS","MongoDB","Node"]
},
{"regno": "2021ICT003",
  "name": "Shanthi",
  "age": 23,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.7,
  "skills": ["JS","MongoDB","Node","React"]
},
{"regno": "2021ICT004",
  "name": "Mantry",
  "age": 26,
  "gender": "Male",
  "degree": "IT",
  "gpa": 3.5,
  "skills": ["JS","Node","React"]
},
{"regno": "2021ICT005",
  "name": "Megathi",
  "age": 22,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.3,
  "skills": ["JS","MongoDB","Node","React"]
}])
{
  acknowledged: true,
  insertedIds: {
    "0": ObjectId('600fb93df633f7be26a6e42d'),
    "1": ObjectId('600fb93df633f7be26a6e42e'),
    "2": ObjectId('600fb93df633f7be26a6e42f'),
    "3": ObjectId('600fb93df633f7be26a6e430')
  }
}
unldb>
```

4. Write the query to show all details of the students.



The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' panel shows a connection to 'localhost:27017' with a database named 'students'. The main panel displays a query in the 'MongoShell' tab: `> db.students.find()`. The results are shown as a JSON array of five student documents. Each document contains fields for `_id`, `regno`, `name`, `age`, `gender`, `degree`, `gpa`, and `skills`. The skills are listed as an array: `['JS', 'MongoDB', 'Node', 'React']`.

```
> db.students.find()
< [
  {
    _id: ObjectId('680fb02f633f7be26ae42c'),
    regno: '2021ICT001',
    name: 'Hema',
    age: 23,
    gender: 'Female',
    degree: 'IT',
    gpa: 3.7,
    skills: [
      'JS',
      'MongoDB',
      'Node',
      'React'
    ]
  },
  {
    _id: ObjectId('680fb02f633f7be26ae42d'),
    regno: '2021ICT002',
    name: 'Helina',
    age: 25,
    gender: 'Male',
    degree: 'IT',
    gpa: 3.6,
    skills: [
      'JS',
      'MongoDB',
      'Node'
    ]
  },
  {
    _id: ObjectId('680fb02f633f7be26ae42e'),
    regno: '2021ICT003',
    name: 'Shanthi',
    age: 23,
    gender: 'Female',
    degree: 'IT',
    gpa: 3.7,
    skills: [
      'JS',
      'MongoDB',
      'Node',
      'React'
    ]
  },
  {
    _id: ObjectId('680fb02f633f7be26ae42f'),
    regno: '2021ICT004',
    name: 'Mantry',
    age: 26,
    gender: 'Male',
    degree: 'IT',
    gpa: 3.5,
    skills: [
      'JS',
      'Node',
      'React'
    ]
  },
  {
    _id: ObjectId('680fb02f633f7be26ae430'),
    regno: '2021ICT005',
    name: 'Mayathi',
    age: 22,
    gender: 'Female',
    degree: 'IT',
    gpa: 3.3,
    skills: [
      'JS',
      'MongoDB',
      'Node',
      'React'
    ]
  }
]
```

5. Write the query to find the detail of a student using registration number.

```
> db.students.find({regno:"2021ICT002"})
< {
  _id: ObjectId('680fbb02f633f7be26a6e42d'),
  regno: '2021ICT002',
  name: 'Helin',
  age: 25,
  gender: 'Male',
  degree: 'IT',
  gpa: 3.6,
  skills: [
    'JS',
    'MongoDB',
    'Node'
  ]
}
```

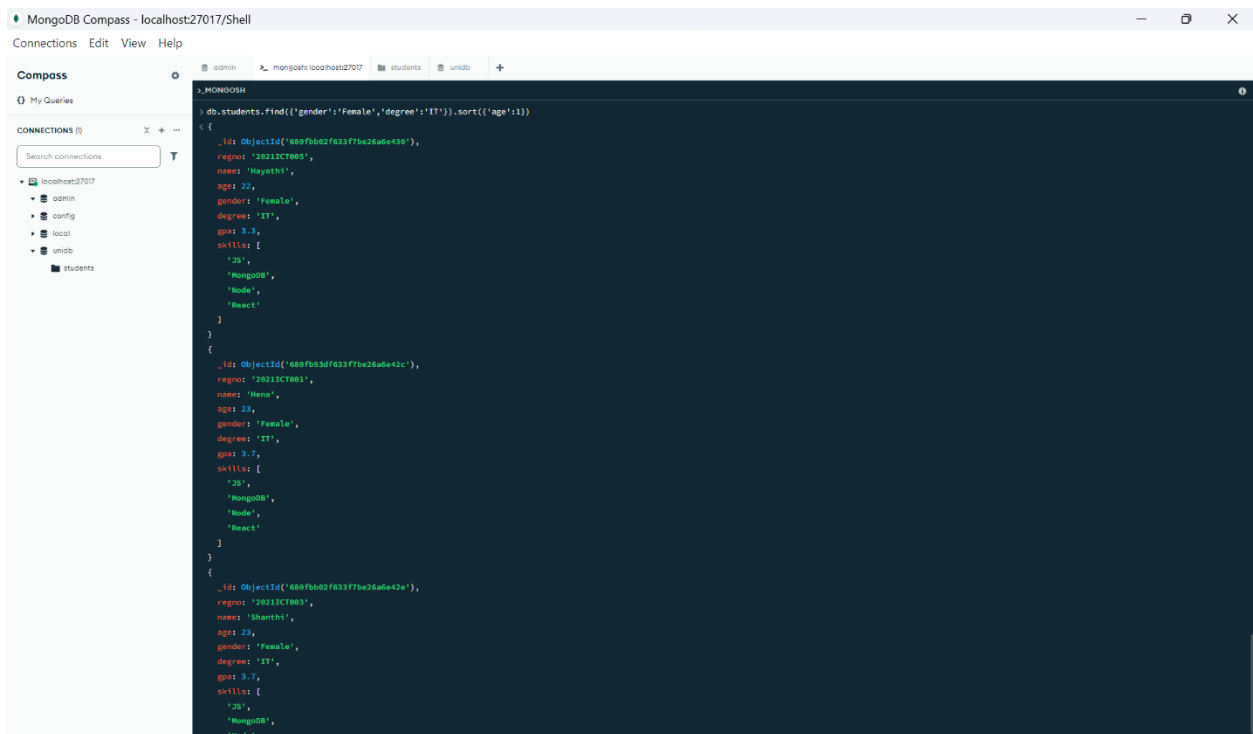
unidb> |

6. Write the query to find the students whose age is greater than 25.

```
> db.students.find({age:{$gt:25}})
< {
  _id: ObjectId('680fbb02f633f7be26a6e42f'),
  regno: '2021ICT004',
  name: 'Hantry',
  age: 26,
  gender: 'Male',
  degree: 'IT',
  gpa: 3.5,
  skills: [
    'JS',
    'Node',
    'React'
  ]
}
```

unidb> |

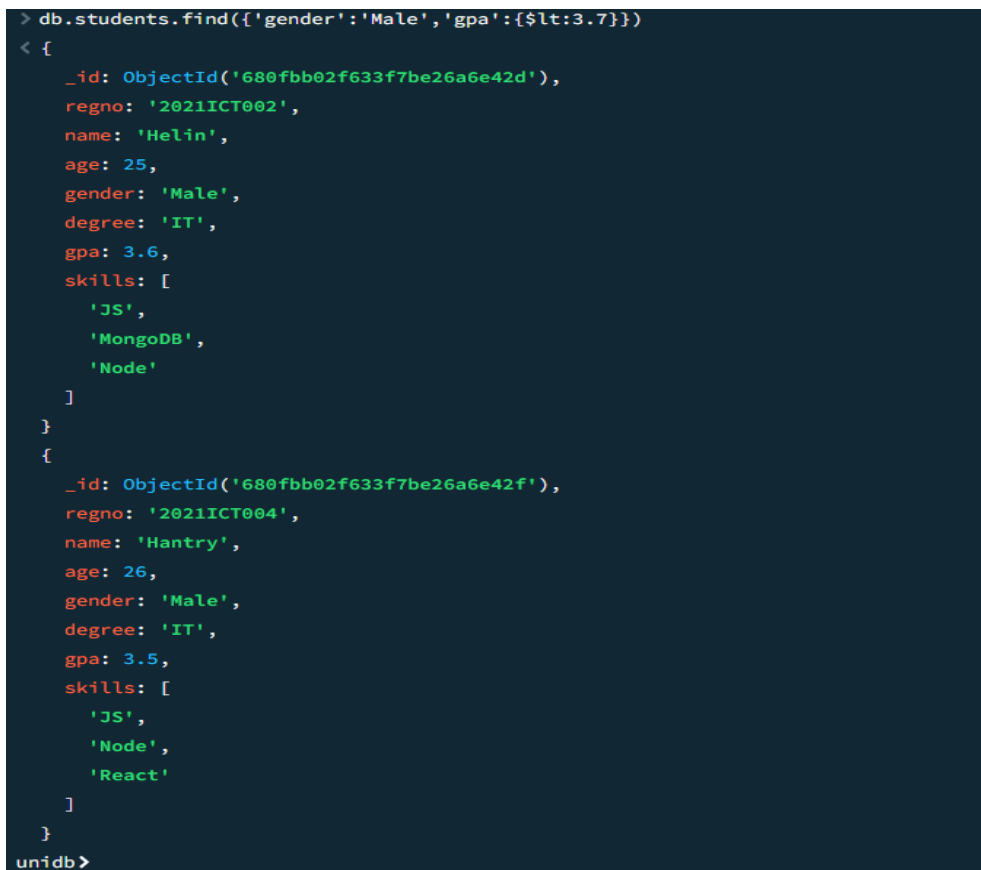
7. Write the query to find the female students doing IT degree and sort by age.



The screenshot shows the MongoDB Compass interface. The left sidebar displays the 'connections' list with 'localhost:27017' selected, showing databases 'admin', 'config', 'local', and 'unibd', and a collection 'students'. The main window shows a query in the 'My Queries' tab: `db.students.find({'gender':'Female','degree':'IT'}).sort({'age':1})`. The results are displayed in a JSON format, showing three female students with IT degrees, sorted by age. The first student is 'Hayathi' (age 22), the second is 'Hena' (age 23), and the third is 'Shanathi' (age 23). All three have a GPA of 3.7 and skills in 'JS', 'MongoDB', and 'Node'.

```
> db.students.find({'gender':'Female','degree':'IT'}).sort({'age':1})
< {
  "_id": ObjectId("689fbb02f633f7be26ae430"),
  "regno": "2021ICT001",
  "name": "Hayathi",
  "age": 22,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.7,
  "skills": [
    "JS",
    "MongoDB",
    "Node",
    "React"
  ]
}
{
  "_id": ObjectId("689fbb02f633f7be26ae42c"),
  "regno": "2021ICT001",
  "name": "Hena",
  "age": 23,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.7,
  "skills": [
    "JS",
    "MongoDB",
    "Node",
    "React"
  ]
}
{
  "_id": ObjectId("689fbb02f633f7be26ae42e"),
  "regno": "2021ICT001",
  "name": "Shanathi",
  "age": 23,
  "gender": "Female",
  "degree": "IT",
  "gpa": 3.7,
  "skills": [
    "JS",
    "MongoDB",
    "Node",
    "React"
  ]
}
```

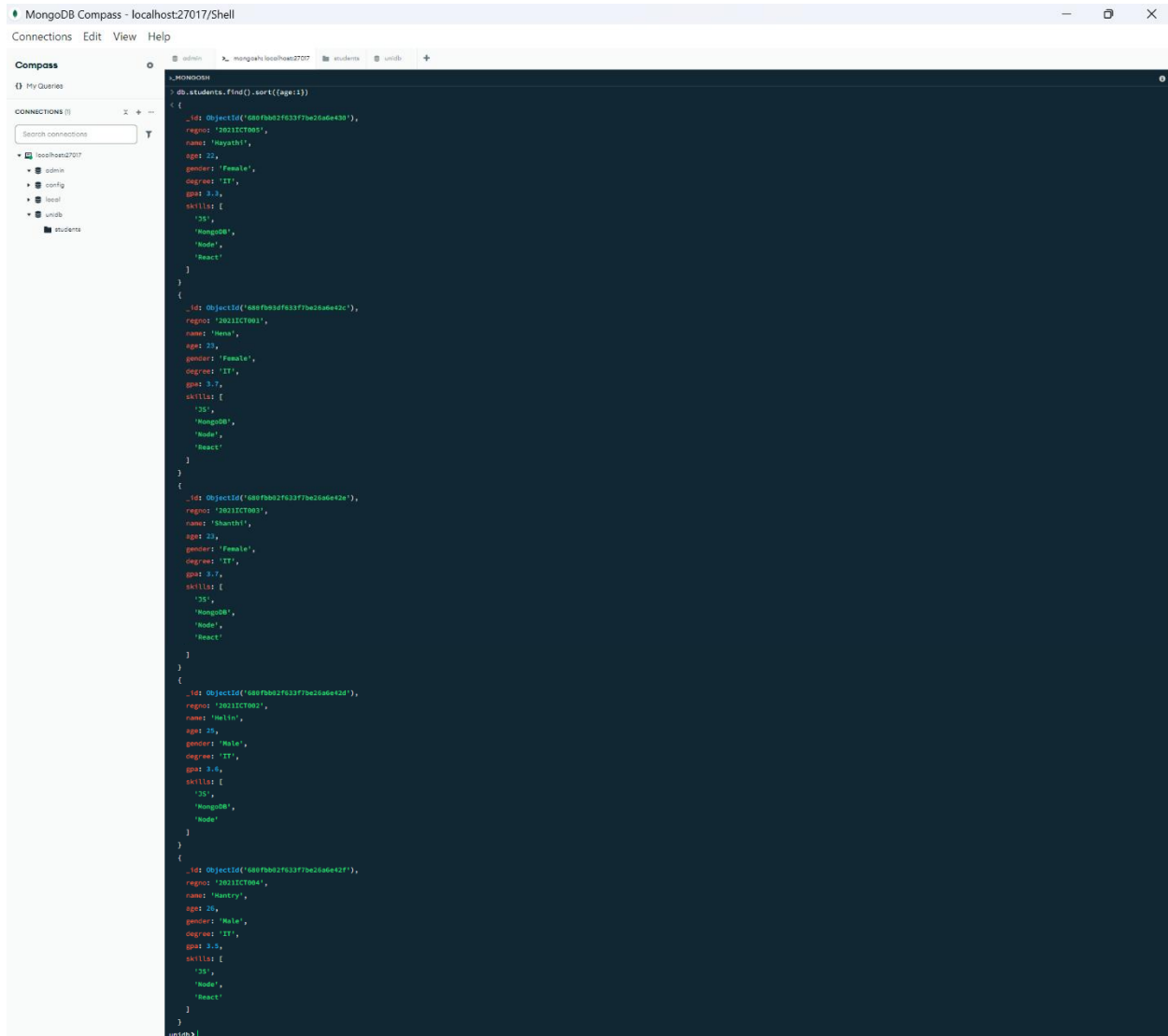
8. Write the query to find male students who got the gpa below 3.7.



The screenshot shows a MongoDB shell session. The query entered is `> db.students.find({'gender':'Male','gpa':{'$lt':3.7}})`. The results are displayed in a JSON format, showing two male students with a GPA below 3.7. The first student is 'Helin' (age 25, GPA 3.6) and the second is 'Hantry' (age 26, GPA 3.5). Both have skills in 'JS', 'Node', and 'React'.

```
> db.students.find({'gender':'Male','gpa':{'$lt':3.7}})
< {
  "_id": ObjectId("689fbb02f633f7be26ae42d"),
  "regno": "2021ICT002",
  "name": "Helin",
  "age": 25,
  "gender": "Male",
  "degree": "IT",
  "gpa": 3.6,
  "skills": [
    "JS",
    "MongoDB",
    "Node"
  ]
}
{
  "_id": ObjectId("689fbb02f633f7be26ae42f"),
  "regno": "2021ICT004",
  "name": "Hantry",
  "age": 26,
  "gender": "Male",
  "degree": "IT",
  "gpa": 3.5,
  "skills": [
    "JS",
    "Node",
    "React"
  ]
}
unibd>
```

9. Write the query to sort by age in ascending order.



The screenshot shows the MongoDB Compass interface. The title bar reads "MongoDB Compass - localhost:27017/Shell". The left sidebar shows the "Connections" list with "localhost:27017" selected. The main editor area displays a query in the "Query" tab:

```
db.students.find().sort([age:1])
```

The "Results" tab shows the output of the query, which is a JSON array of student documents sorted by age in ascending order. The documents are as follows:

```
[{"_id": "608f8b02763377be20de4301", "regno": "2021ICT005", "name": "Nayanthi", "age": 22, "gender": "Female", "degree": "IT", "gpa": 3.3, "skills": ["JS", "MongoDB", "Node", "React"]}, {"_id": "608f8b02763377be20de4302", "regno": "2021ICT001", "name": "Hema", "age": 23, "gender": "Female", "degree": "IT", "gpa": 3.7, "skills": ["JS", "MongoDB", "Node", "React"]}, {"_id": "608f8b02763377be20de4303", "regno": "2021ICT003", "name": "Shanthi", "age": 23, "gender": "Female", "degree": "IT", "gpa": 3.7, "skills": ["JS", "MongoDB", "Node", "React"]}, {"_id": "608f8b02763377be20de4304", "regno": "2021ICT002", "name": "Helin", "age": 25, "gender": "Male", "degree": "IT", "gpa": 3.6, "skills": ["JS", "MongoDB", "Node"]}, {"_id": "608f8b02763377be20de4307", "regno": "2021ICT004", "name": "Hantry", "age": 26, "gender": "Male", "degree": "IT", "gpa": 3.3, "skills": ["JS", "Node", "React"]}]
```

10. Write the query to find the male students and sort them by age in descending order.

```
> db.students.find({'gender':'Male'}).sort({'age':-1})
< {
  _id: ObjectId('680fbb02f633f7be26a6e42f'),
  regno: '2021ICT004',
  name: 'Hantry',
  age: 26,
  gender: 'Male',
  degree: 'IT',
  gpa: 3.5,
  skills: [
    'JS',
    'Node',
    'React'
  ]
}
{
  _id: ObjectId('680fbb02f633f7be26a6e42d'),
  regno: '2021ICT002',
  name: 'Helin',
  age: 25,
  gender: 'Male',
  degree: 'IT',
  gpa: 3.6,
  skills: [
    'JS',
    'MongoDB',
    'Node'
  ]
}
unidb>
```