S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 6

Aim: To perform data retrieval operations in MongoDB using Comparison query operators ($eq, $ne, $gt, $lt, $gte, $lte) and Logical operators ($and, $or, $not, $nor) on a sample database.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: 7th / 4th

Academic Session: 2025-2026

Date of Performance: __________

Date of Submission: __________

AIM: To perform data retrieval operations in MongoDB using comparison query operators ($eq, $ne, $gt, $lt, $gte, $lte) and logical operators ($and, $or, $not, $nor) on a sample database.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

Objective:

- To understand and apply various MongoDB comparison query operators such as $eq, $ne, $gt, $lt, $gte, $lte.
- To implement logical operators like $and, $or, $not, and $nor for complex query conditions.
- To analyze and retrieve data efficiently from MongoDB collections using multiple filtering techniques.

Expected Learning Outcomes:

After successful completion of this practical, the student will be able to:

1. Understand the use of comparison query operators to filter data based on different conditions.
2. Implement logical operators to combine multiple query conditions in MongoDB.
3. Design and execute optimized queries on structured and semi-structured data.
4. Analyze datasets and extract meaningful information using compound conditions.
5. Develop hands-on proficiency in writing MongoDB queries for real-time applications.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement: High Configuration computer

Software Requirement: MongoDB-8.0, Mongo Shell

THEORY:

MongoDB – Comparison Query Operators

MongoDB uses various comparison query operators to compare the values of the documents. The following table contains the comparison query operators:

Operators	**Description**
$eq	It is used to match the values of the fields that are **equal** to a specified value.
$ne	It is used to match all values of the field that are **not equal** to a specified value.
$gt	It is used to match values of the fields that are **greater than** a specified value.

$gte	It is used to match values of the fields that are **greater than equal** to the specified value.
$lt	It is used to match values of the fields that are **less than** a specified value.
$lte	It is used to match values of the fields that are **less than equals** to the specified value
$in	It is used to match any of the values specified in an array.
$nin	It is used to match none of the values specified in an array.

Suppose we create our database name is 'myinfo' and our collection name is 'testtable'.

If we want to fetch documents from the collection "testtable" which contains the value of "age " is more than 22, the following mongodb command can be used :

1) MongoDB (=) equality operator - $eq

```
>db.testtable.find({age : {$eq : 22}}).pretty();
```

2) MongoDB (>) greater than operator - $gt

```
>db.testtable.find({age : {$gt : 22}}).pretty();
```

find() method displays the documents in a non structured format but to display the results in a formatted way, the pretty() method can be used.

3) MongoDB (>=) greater than equal to operator - $gte

If we want to fetch documents from the collection "testtable" which contains the value of "age " is more than or equal to 22, the following mongodb command can be used :

```
>db.testtable.find({age : {$gte : 22}}).pretty();
```

4) MongoDB (<) less than operator - $lt

If we want to fetch documents from the collection "testtable" which contains the value of "age " is less than 19, the following mongodb command can be used :

```
>db.testtable.find({age : {$lt : 19}}).pretty();
```

5) MongoDB (<=) less than equal to operator - $lte

If we want to fetch documents from the collection "testtable" which contains the value of "age " is less than or equal to 19, the following mongodb command can be used :

```
>db.testtable.find({age : {$lte : 19}}).pretty();
```

6) MongoDB query using (<) and (>) operator - $lt and $gt

If we want to fetch documents from the collection "testtable" which contains the value of "age " is greater than 17 and less than 24, the following mongodb command can be used :

```
>db.testtable.find({age : {$lt :24, $gt : 17}}).pretty();
```

7) Inequality Operator ($ne): This operator matches values that are not equal to a specified value.

```
db.testtable.find({age : {$ne : 22}})
```

8) Not In Operator ($nin): This operator matches none of the values specified in an array.

db.users.find({ role: { $nin: ["Admin", "Superuser"] } })

MongoDB – Logical Query Operators

Operator	Description
$and	It Joins query clauses with a logical AND. Returns documents that match all the conditions.
$or	{ $or: [{ <expression1> }, { <expression2> }, ...] }
$nor	It Joins query clauses with a logical NOR. Returns documents that fail to match all the conditions.
$not	It Inverts the effect of a query expression. Returns documents that do not match the query expression.

Matches docs where all conditions are true:

Employees older than 25 and high salary
{ $and: [{ age: { $gt: 25 } }, { salary: { $gt: 40000 } }] }

Matches docs where any condition is true

Lives in Nagpur or earns well

{ $or: [{ city: "Nagpur" }, { salary: { $gt: 50000 } }] }

Negates a condition

Employees not earning more than 50K

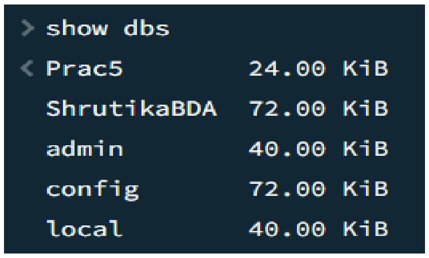
{ salary: { $not: { $gt: 50000 } } }

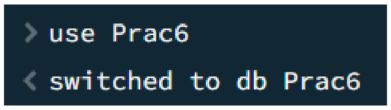
Returns docs where all conditions fail.

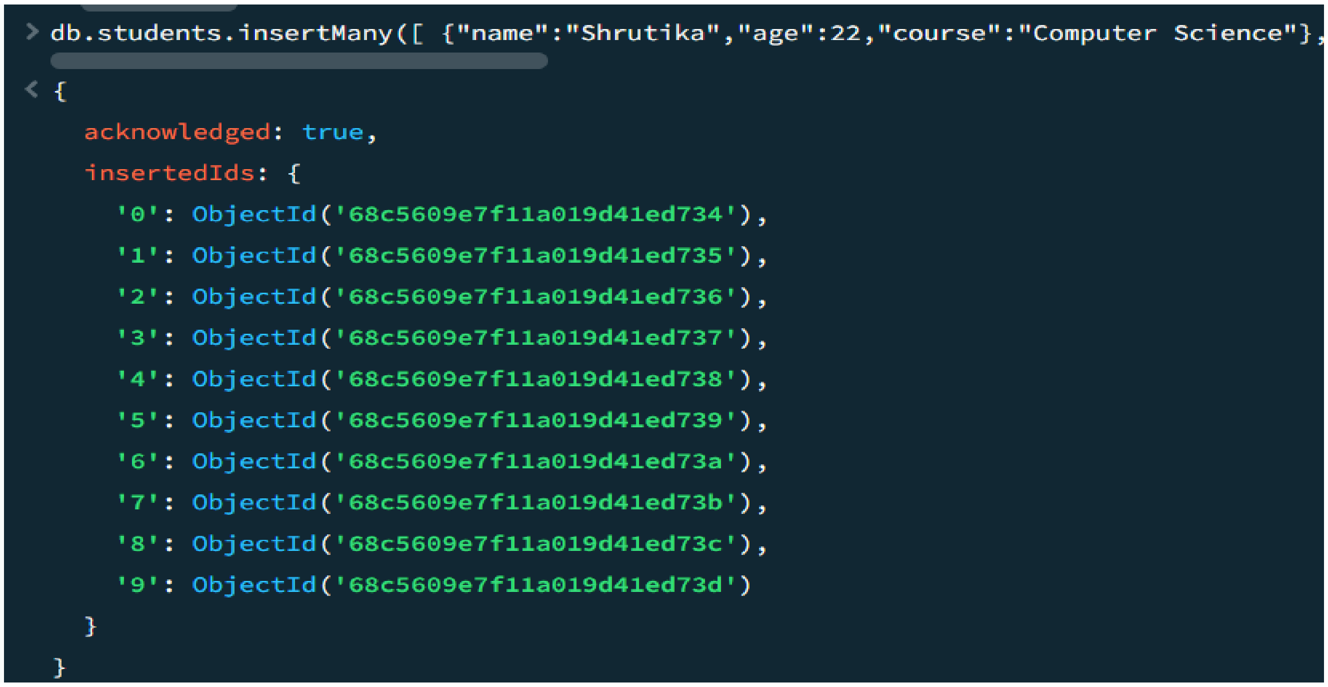
Not from Nagpur and lower salary

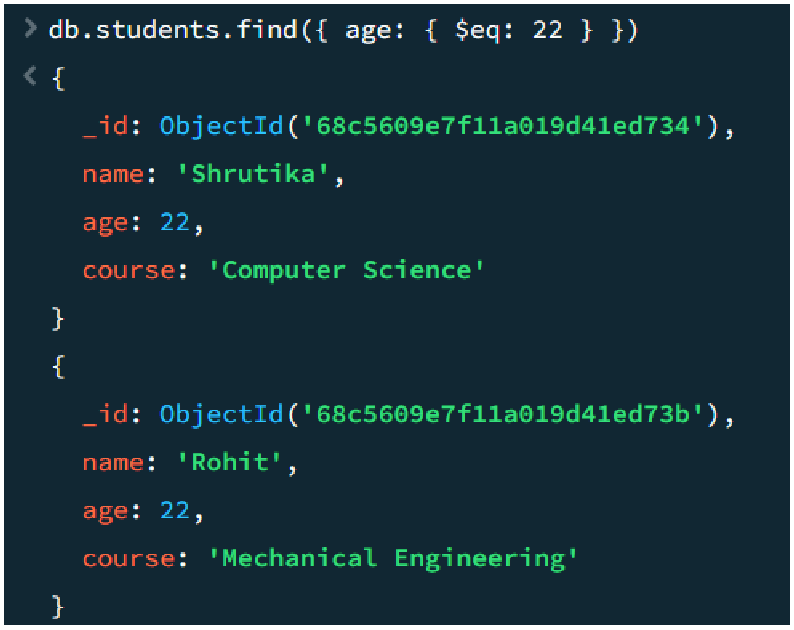
{ $nor: [{ city: "Nagpur" }, { salary: {$gt: 40000 } }] }

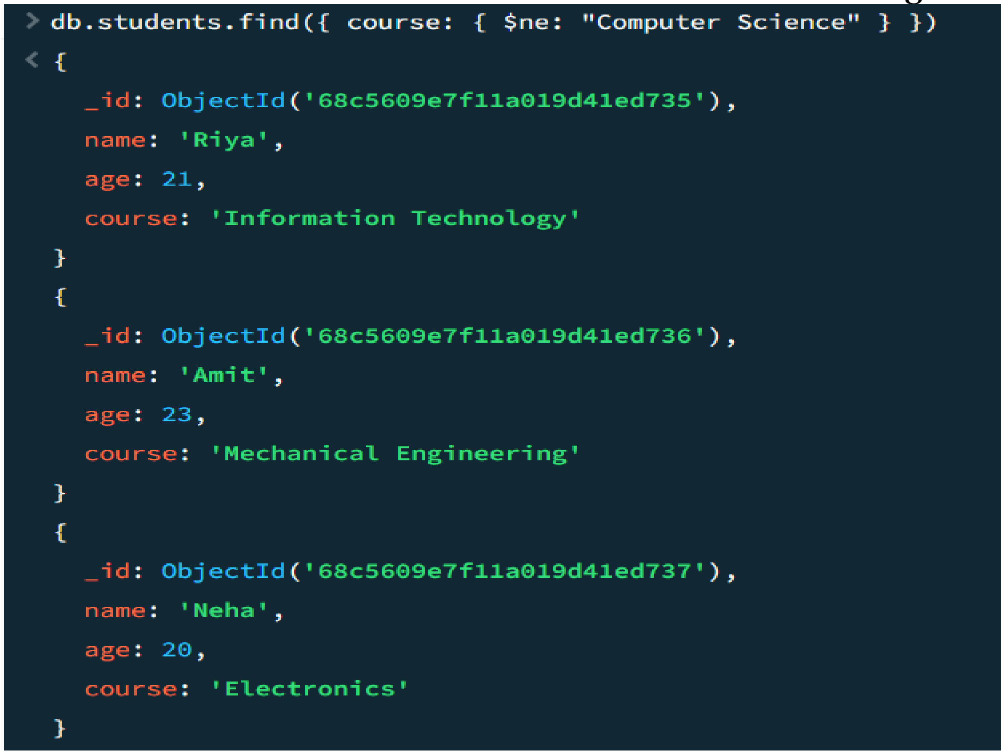
INPUT / OUTPUT (SCREENSHOTS):


```
> show dbs
< Prac5         24.00 KiB
  ShrutikaBDA   72.00 KiB
  admin         40.00 KiB
  config        72.00 KiB
  local         40.00 KiB
```

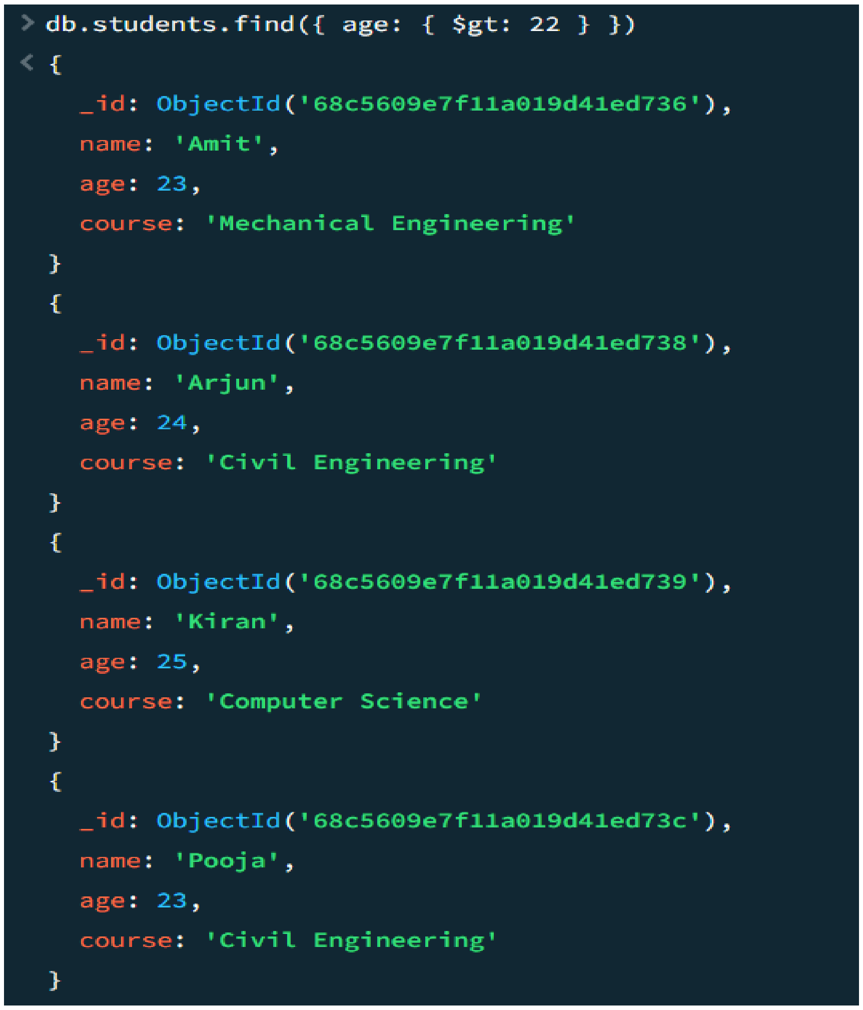

```
> use Prac6
< switched to db Prac6
```


```
> db.students.insertMany([ {"name":"Shrutika","age":22,"course":"Computer Science"},
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68c5609e7f11a019d41ed734'),
      '1': ObjectId('68c5609e7f11a019d41ed735'),
      '2': ObjectId('68c5609e7f11a019d41ed736'),
      '3': ObjectId('68c5609e7f11a019d41ed737'),
      '4': ObjectId('68c5609e7f11a019d41ed738'),
      '5': ObjectId('68c5609e7f11a019d41ed739'),
      '6': ObjectId('68c5609e7f11a019d41ed73a'),
      '7': ObjectId('68c5609e7f11a019d41ed73b'),
      '8': ObjectId('68c5609e7f11a019d41ed73c'),
      '9': ObjectId('68c5609e7f11a019d41ed73d')
    }
  }
```

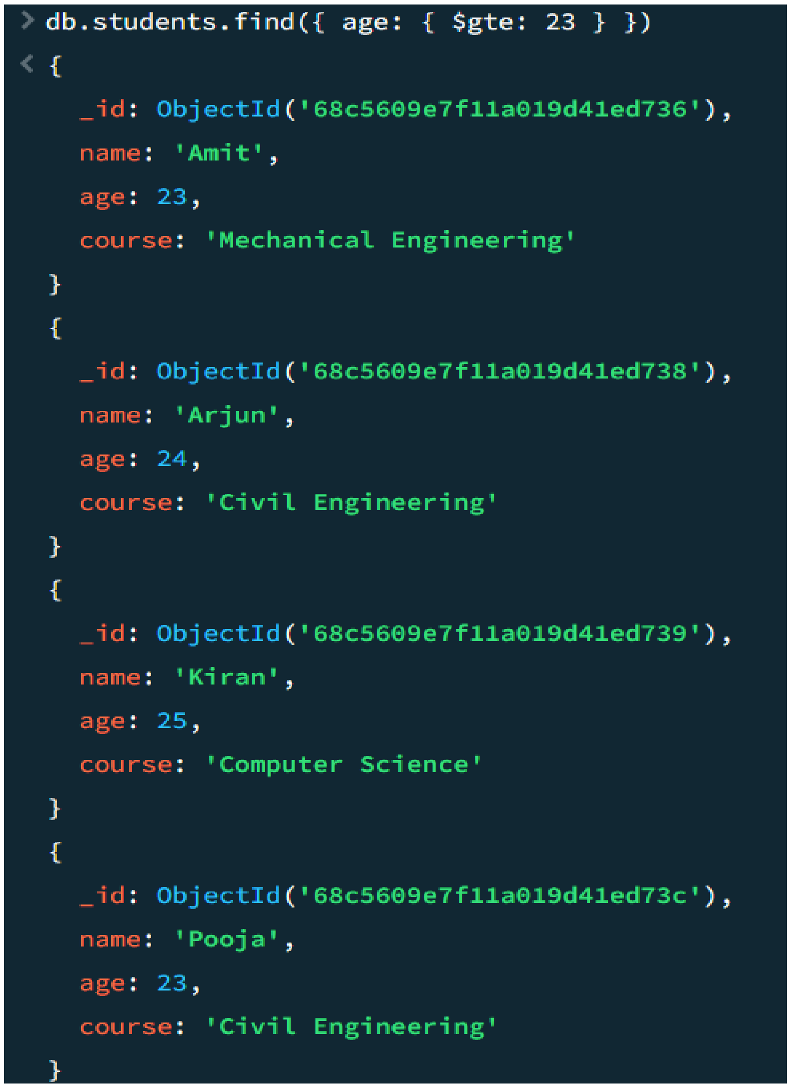

```
> db.students.find({ age: { $eq: 22 } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73b'),
    name: 'Rohit',
    age: 22,
    course: 'Mechanical Engineering'
  }
```



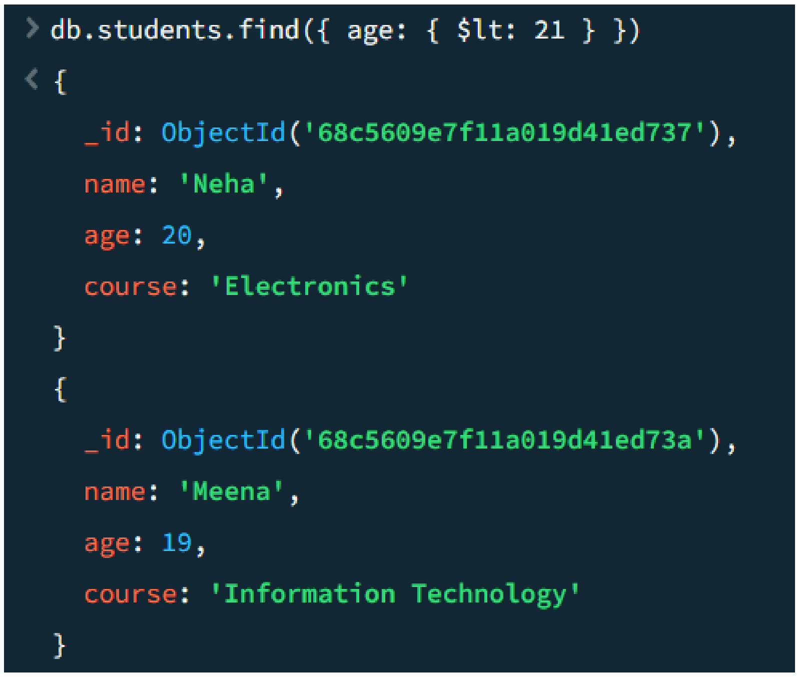
```
> db.students.find({ course: { $ne: "Computer Science" } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed736'),
    name: 'Amit',
    age: 23,
    course: 'Mechanical Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
```



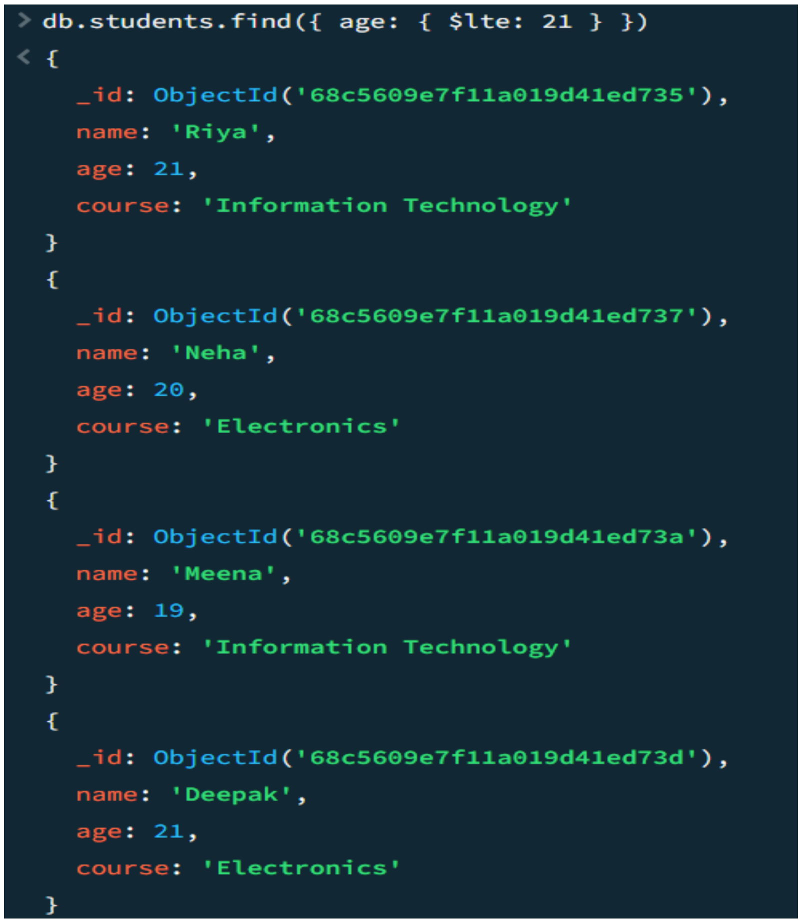
```
> db.students.find({ age: { $gt: 22 } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed736'),
    name: 'Amit',
    age: 23,
    course: 'Mechanical Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed738'),
    name: 'Arjun',
    age: 24,
    course: 'Civil Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed739'),
    name: 'Kiran',
    age: 25,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73c'),
    name: 'Pooja',
    age: 23,
    course: 'Civil Engineering'
  }
```



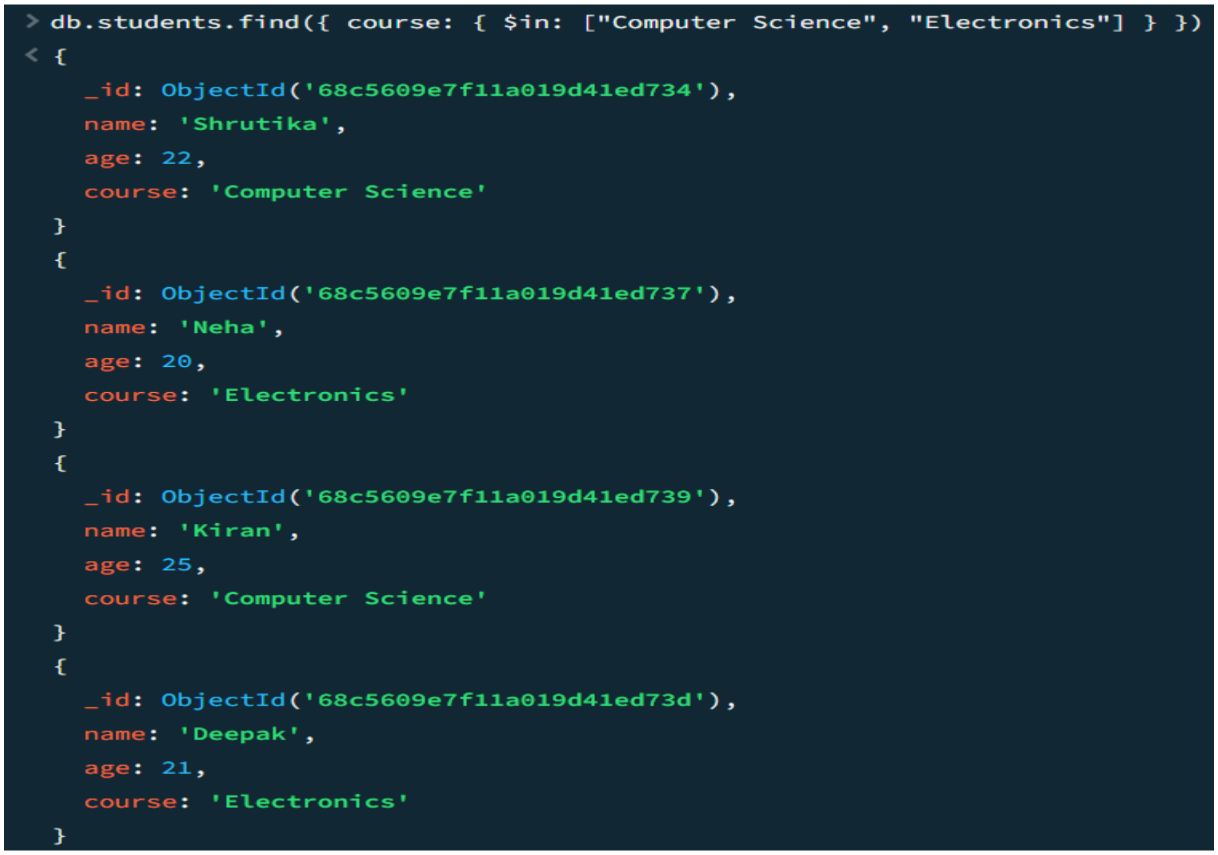
```
> db.students.find({ age: { $gte: 23 } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed736'),
    name: 'Amit',
    age: 23,
    course: 'Mechanical Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed738'),
    name: 'Arjun',
    age: 24,
    course: 'Civil Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed739'),
    name: 'Kiran',
    age: 25,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73c'),
    name: 'Pooja',
    age: 23,
    course: 'Civil Engineering'
  }
```



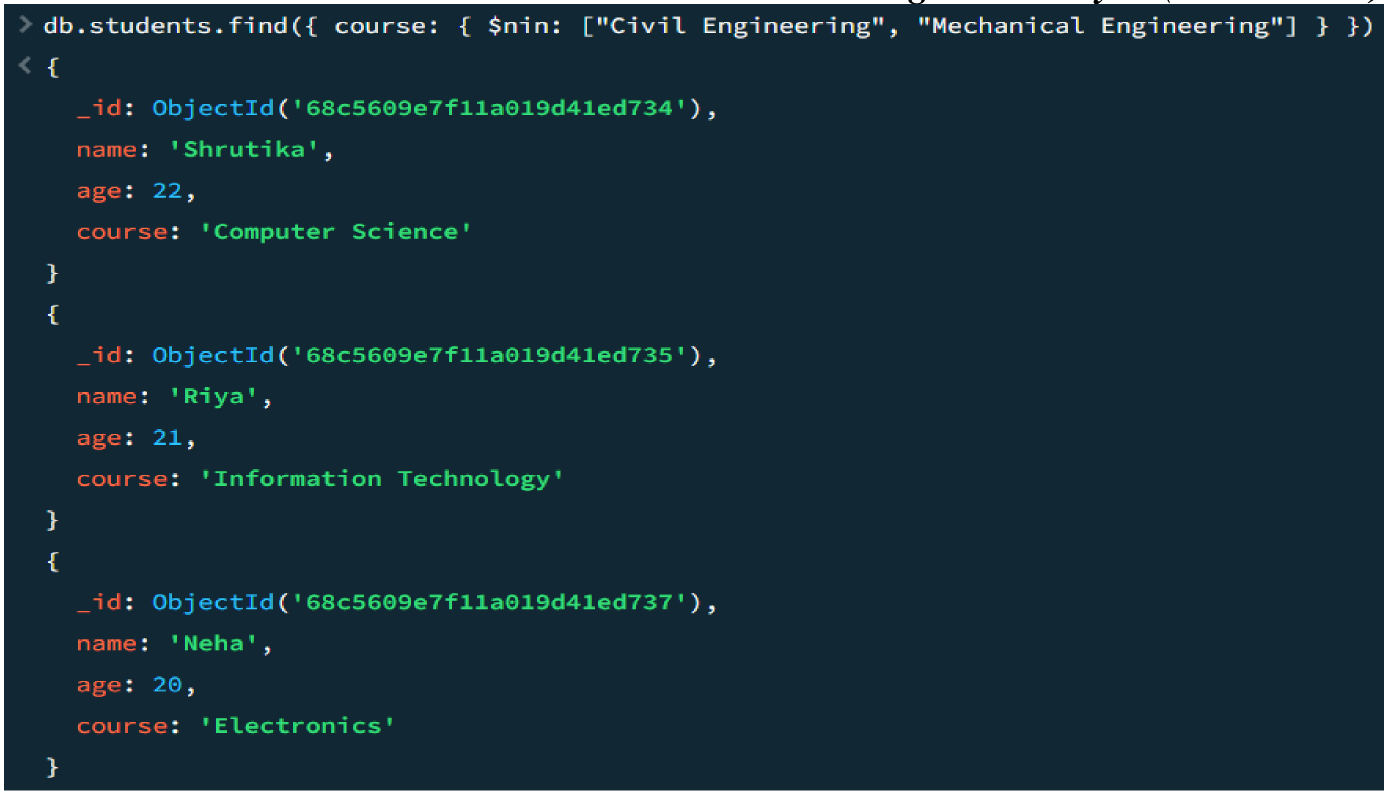
```
> db.students.find({ age: { $lt: 21 } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73a'),
    name: 'Meena',
    age: 19,
    course: 'Information Technology'
  }
```



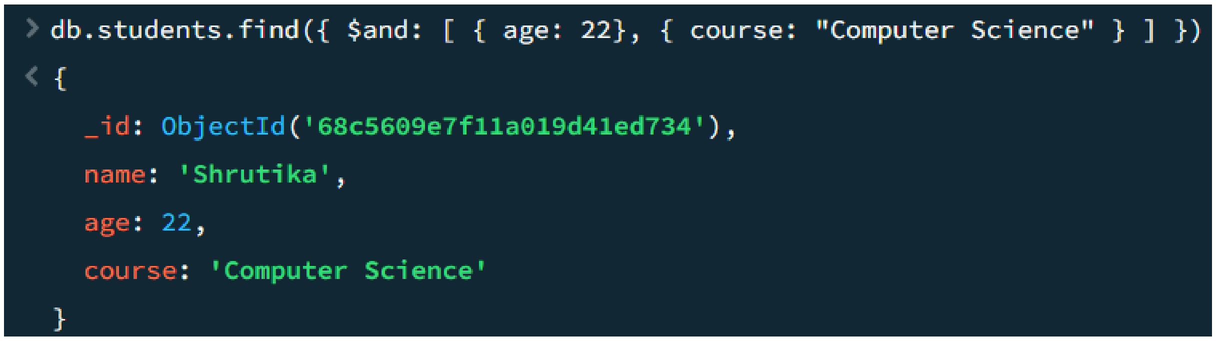
```
> db.students.find({ age: { $lte: 21 } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73a'),
    name: 'Meena',
    age: 19,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73d'),
    name: 'Deepak',
    age: 21,
    course: 'Electronics'
  }
```



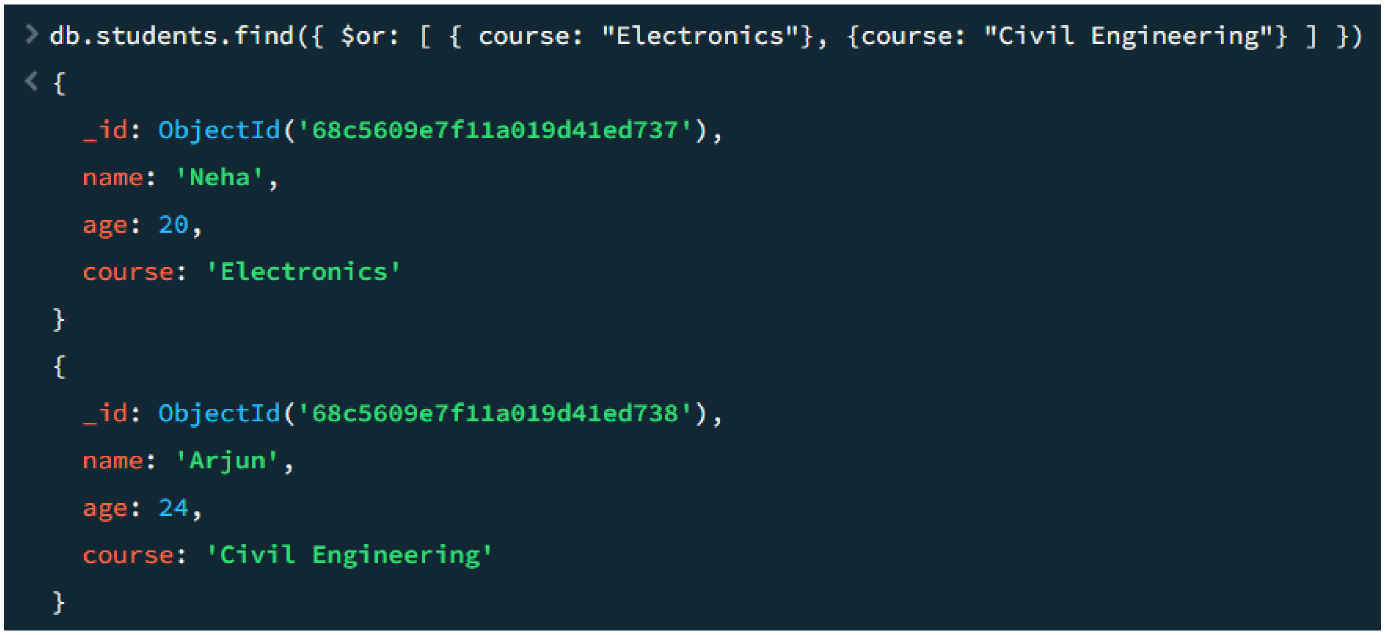
```
> db.students.find({ course: { $in: ["Computer Science", "Electronics"] } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed739'),
    name: 'Kiran',
    age: 25,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73d'),
    name: 'Deepak',
    age: 21,
    course: 'Electronics'
  }
```



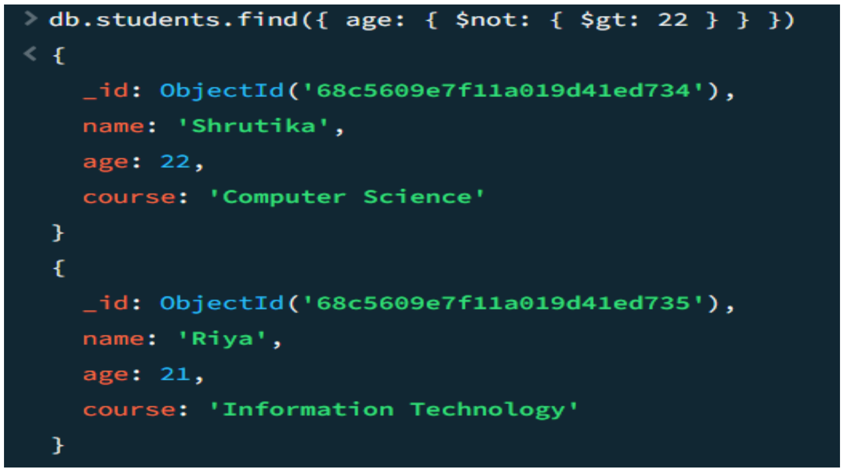
```
> db.students.find({ course: { $nin: ["Civil Engineering", "Mechanical Engineering"] } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
```



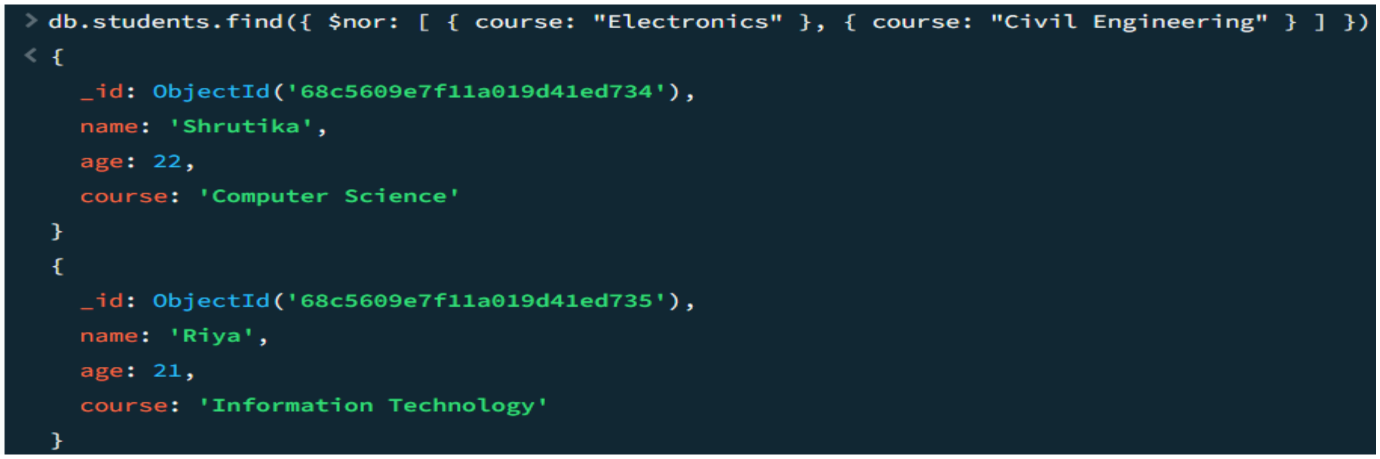
```
> db.students.find({ $and: [ { age: 22}, { course: "Computer Science" } ] })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
```



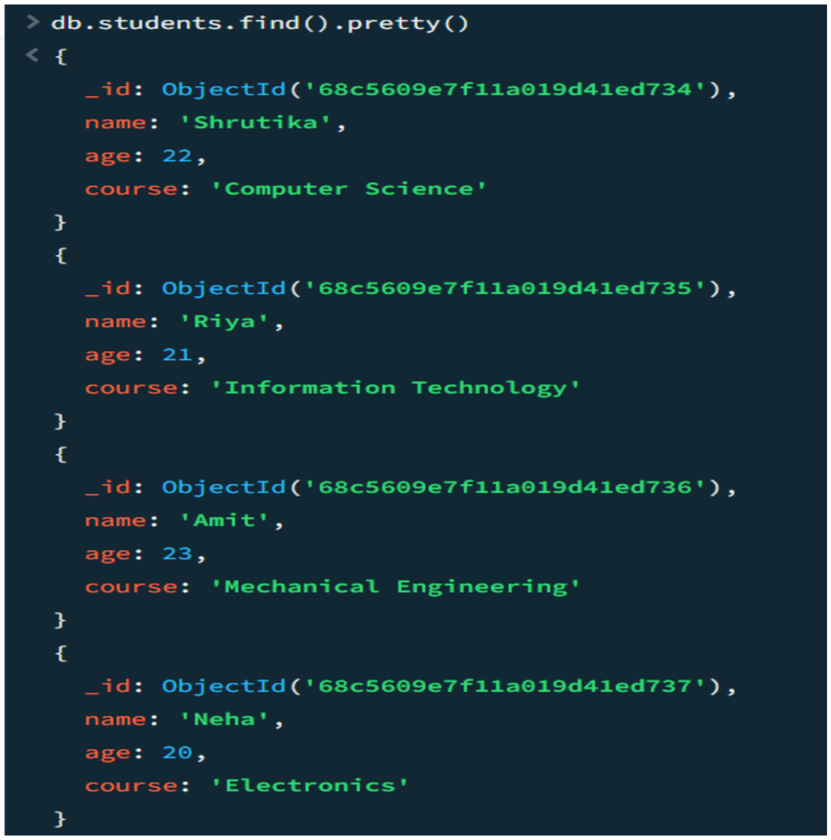
```
> db.students.find({ $or: [ { course: "Electronics"}, {course: "Civil Engineering"} ] })
< {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed738'),
    name: 'Arjun',
    age: 24,
    course: 'Civil Engineering'
  }
```



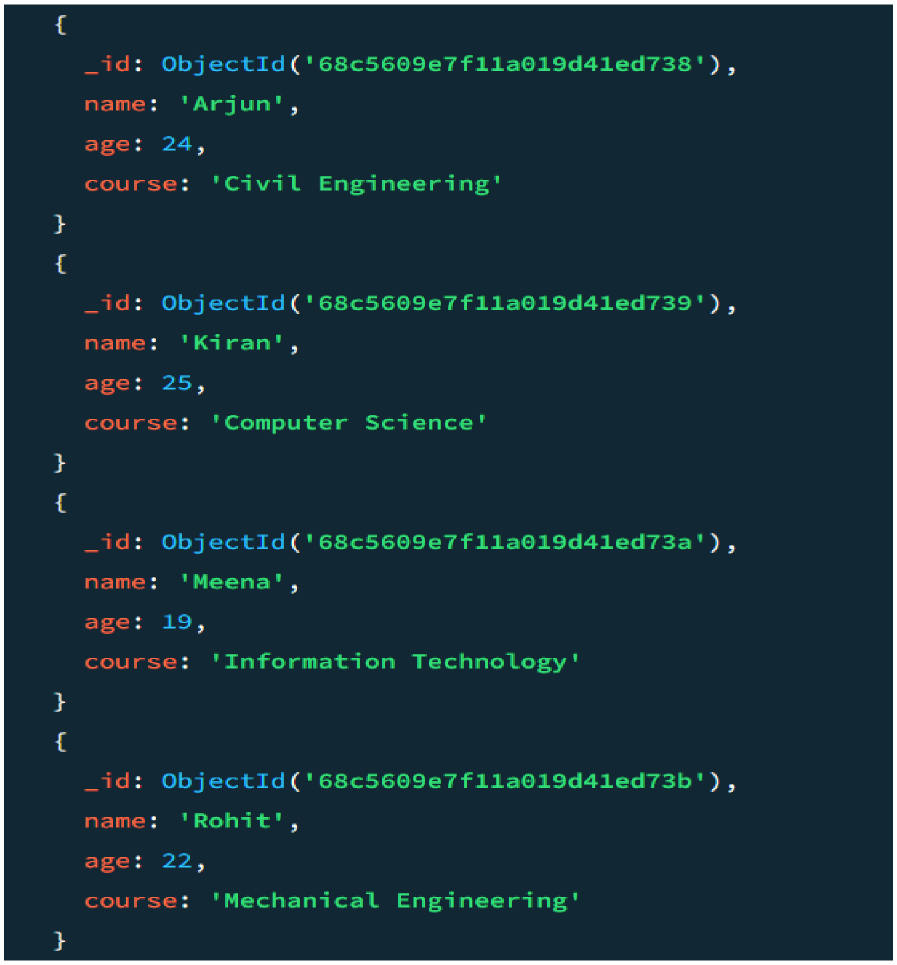
```
> db.students.find({ age: { $not: { $gt: 22 } } })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
```



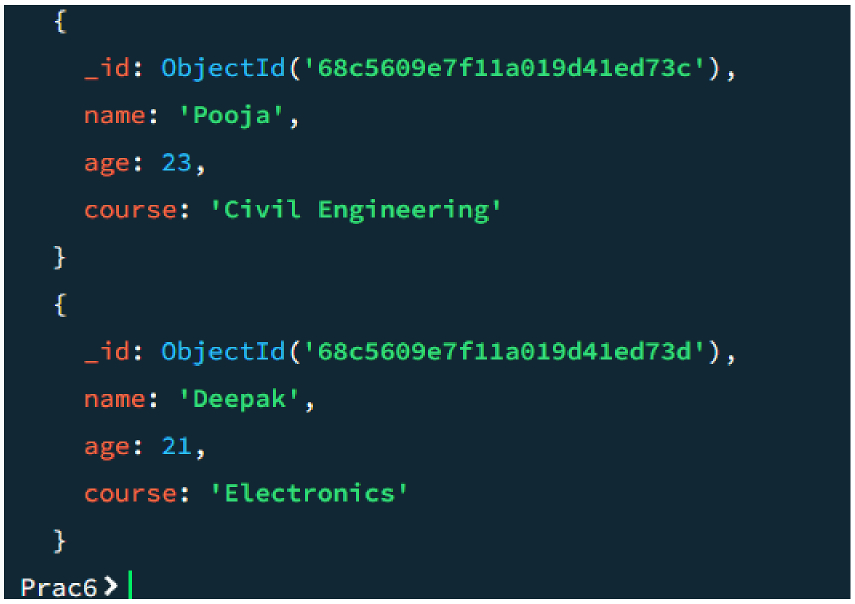
```
> db.students.find({ $nor: [ { course: "Electronics" }, { course: "Civil Engineering" } ] })
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
```



```
> db.students.find().pretty()
< {
    _id: ObjectId('68c5609e7f11a019d41ed734'),
    name: 'Shrutika',
    age: 22,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed735'),
    name: 'Riya',
    age: 21,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed736'),
    name: 'Amit',
    age: 23,
    course: 'Mechanical Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed737'),
    name: 'Neha',
    age: 20,
    course: 'Electronics'
  }
```



```
  {
    _id: ObjectId('68c5609e7f11a019d41ed738'),
    name: 'Arjun',
    age: 24,
    course: 'Civil Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed739'),
    name: 'Kiran',
    age: 25,
    course: 'Computer Science'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73a'),
    name: 'Meena',
    age: 19,
    course: 'Information Technology'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73b'),
    name: 'Rohit',
    age: 22,
    course: 'Mechanical Engineering'
  }
```



```
  {
    _id: ObjectId('68c5609e7f11a019d41ed73c'),
    name: 'Pooja',
    age: 23,
    course: 'Civil Engineering'
  }
  {
    _id: ObjectId('68c5609e7f11a019d41ed73d'),
    name: 'Deepak',
    age: 21,
    course: 'Electronics'
  }
Prac6>
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- Question: Explain a scenario where an $and operator is technically redundant but improves query clarity, and provide the one-line query.
- What is a key behavioral difference between $not: { $ne: 'value' } and $eq: 'value'?
- How can a single $nor query be functionally equivalent to a combination of $and and $not operators?
- Write a one-line query to find all documents where a number field x is between 10 and 20 (inclusive) OR not equal to 50.

REFERENCE:

- https://blog.sqlauthority.com/2020/05/22/mongodb-fundamentals-crud-deleting-objects-day-5-of-6/
- https://www.tutorialspoint.com/mongodb/mongodb_quick_guide.htm
- https://dbdmg.polito.it/wordpress/wp-content/uploads/2019/11/02-MongoDB-query.pdf

Observation book: (3)	Viva-Voce (3)	Quality of Submission and timely Evaluation (4)
Total:		Sign with date: