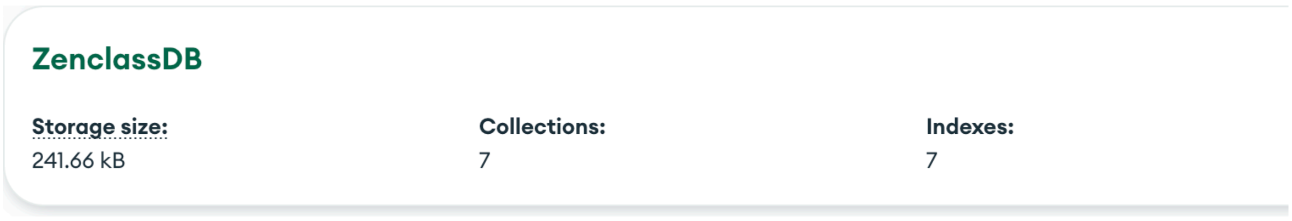
MongoDB Task

Databases Creation :

Created database under the name ‘ZenclassDB’.



ZenclassDB

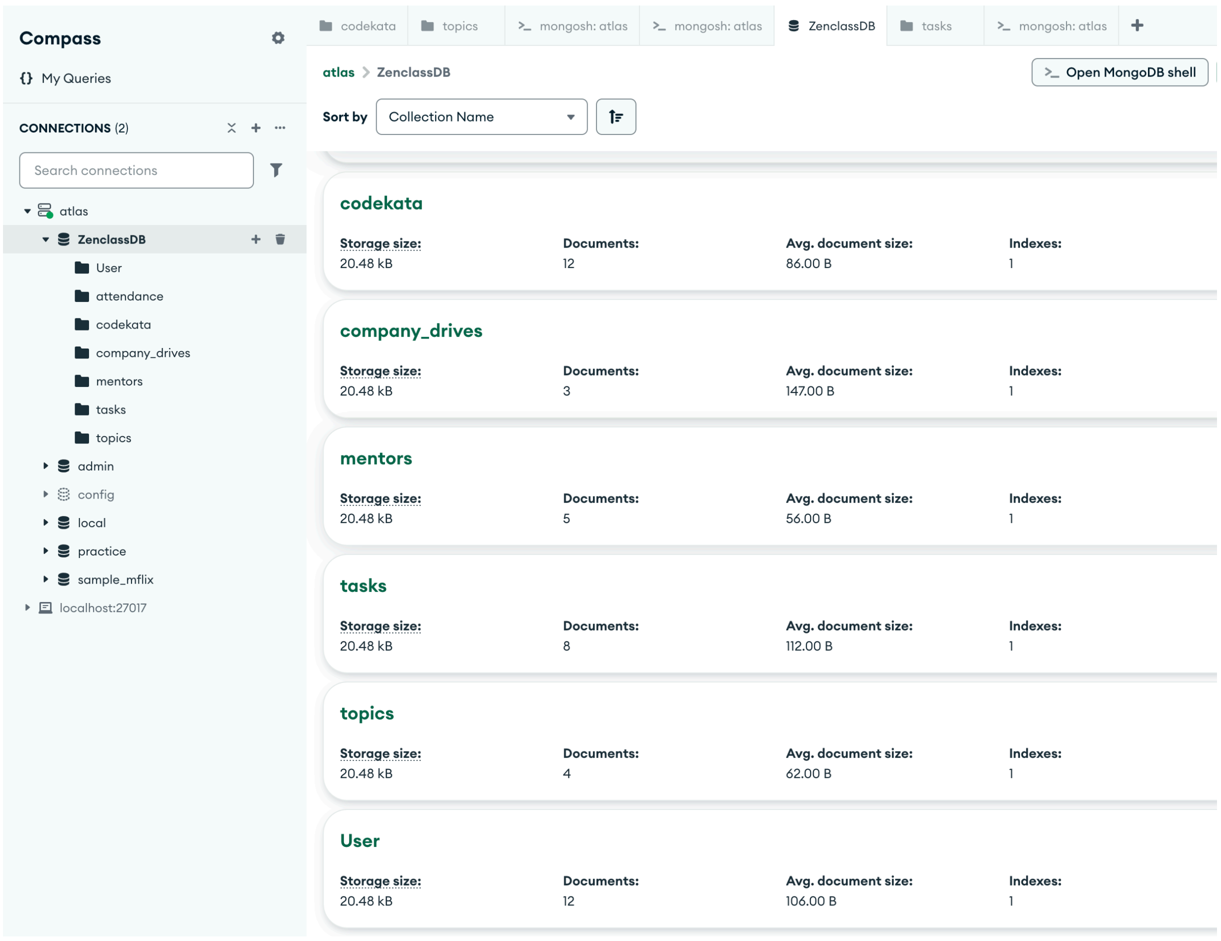
Storage size: 241.66 kB

Collections: 7

Indexes: 7

Collections:

All the required collections are created using the following command,
db.createCollection(‘ ’);

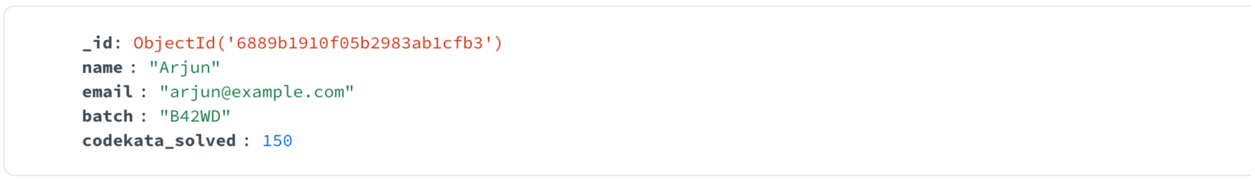


	Storage size:	Documents:	Avg. document size:	Indexes:
codekata	20.48 kB	12	86.00 B	1
company_drives	20.48 kB	3	147.00 B	1
mentors	20.48 kB	5	56.00 B	1
tasks	20.48 kB	8	112.00 B	1
topics	20.48 kB	4	62.00 B	1
User	20.48 kB	12	106.00 B	1

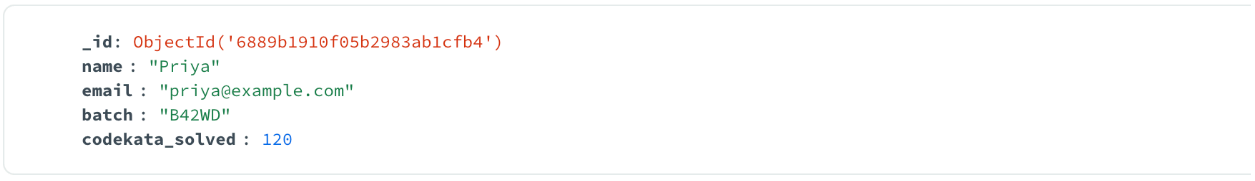
The collections are created and documents are inserted using the command
“db.collection_name.insertMany([{ },…,{ },]);”

User :

These are the sample data from the collection ‘user’.



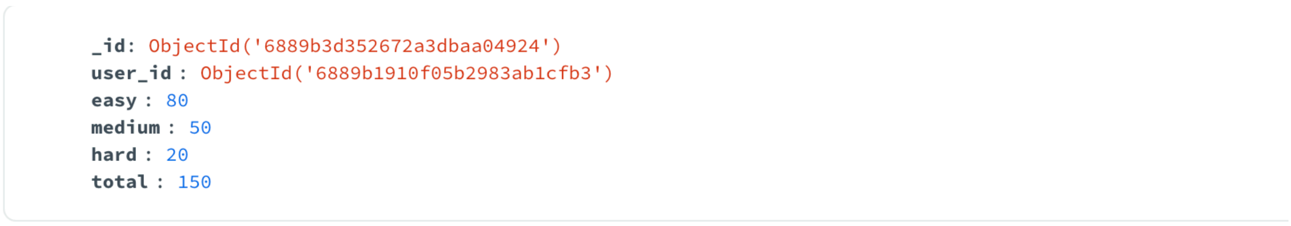
```
_id: ObjectId('6889b1910f05b2983ab1cfb3')
name : "Arjun"
email : "arjun@example.com"
batch : "B42WD"
codekata_solved : 150
```



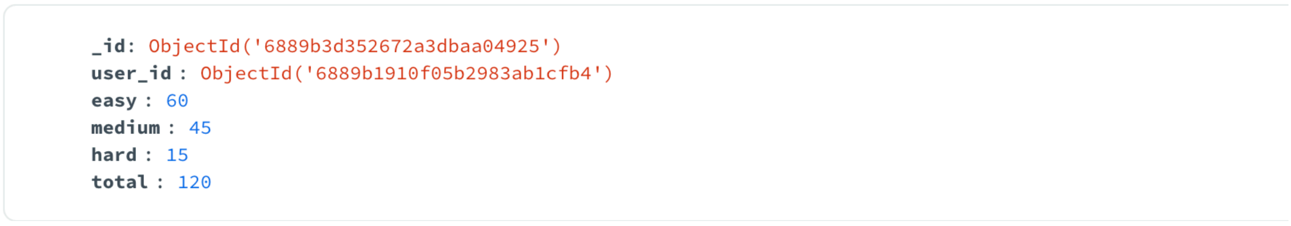
```
_id: ObjectId('6889b1910f05b2983ab1cfb4')
name : "Priya"
email : "priya@example.com"
batch : "B42WD"
codekata_solved : 120
```

Codekata :

These are the sample data from the ‘codekata’ collection .



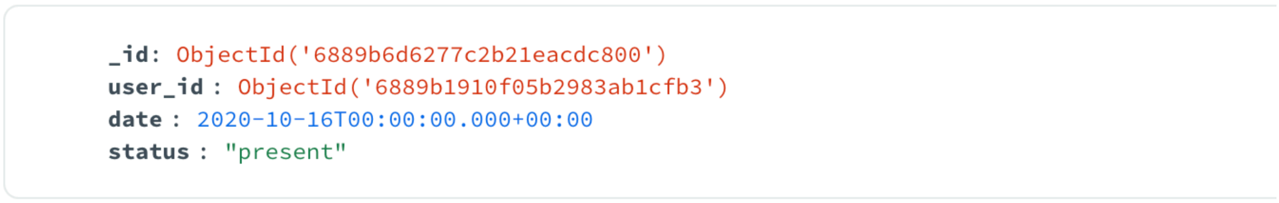
```
_id: ObjectId('6889b3d352672a3dbaa04924')
user_id : ObjectId('6889b1910f05b2983ab1cfb3')
easy : 80
medium : 50
hard : 20
total : 150
```



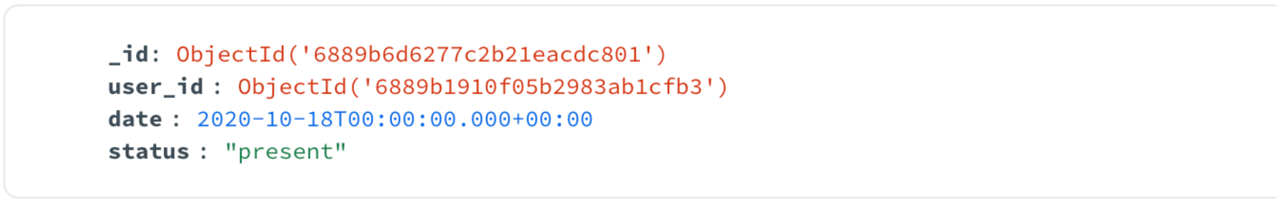
```
_id: ObjectId('6889b3d352672a3dbaa04925')
user_id : ObjectId('6889b1910f05b2983ab1cfb4')
easy : 60
medium : 45
hard : 15
total : 120
```

Attendance:

These are the sample data from the ‘Attendance’ collection .



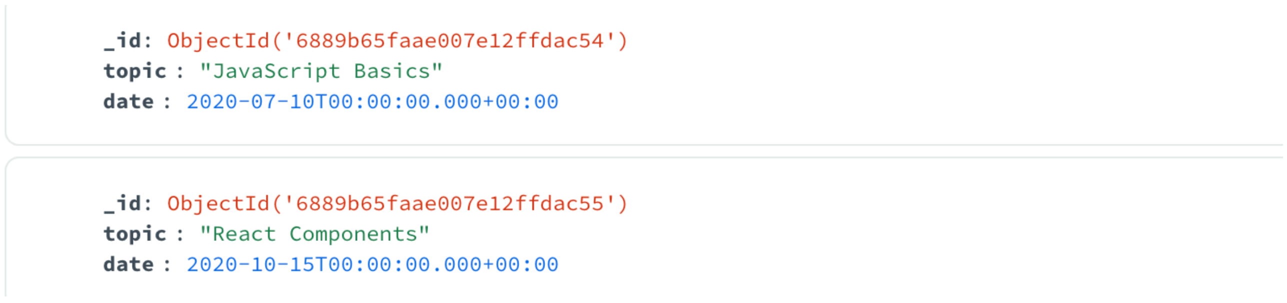
```
_id: ObjectId('6889b6d6277c2b21eacdc800')
user_id : ObjectId('6889b1910f05b2983ab1cfb3')
date : 2020-10-16T00:00:00.000+00:00
status : "present"
```



```
_id: ObjectId('6889b6d6277c2b21eacdc801')
user_id : ObjectId('6889b1910f05b2983ab1cfb3')
date : 2020-10-18T00:00:00.000+00:00
status : "present"
```

Topics:

These are the sample data from the ‘Topics’ collection .

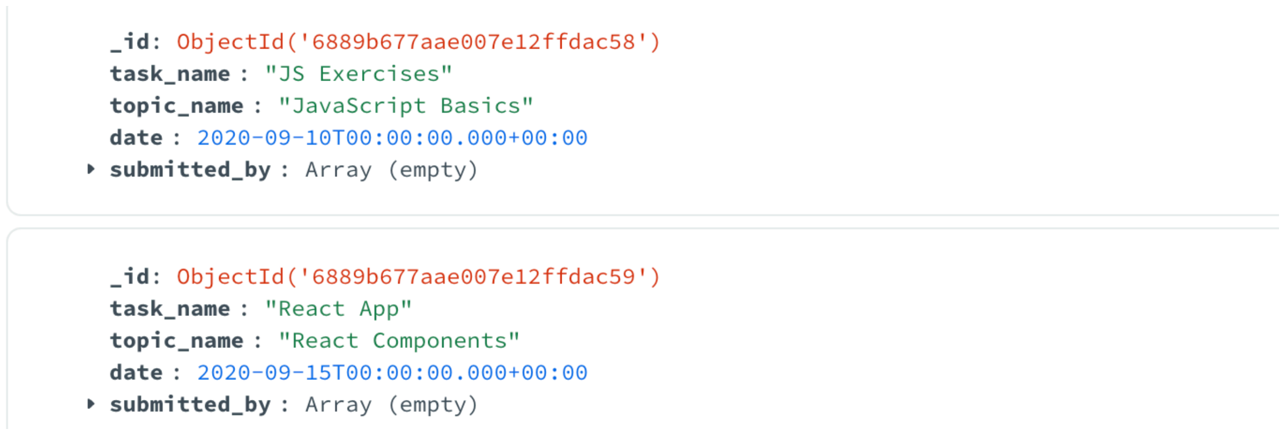


```
_id: ObjectId('6889b65faae007e12ffdac54')
topic : "JavaScript Basics"
date : 2020-07-10T00:00:00.000+00:00
```

```
_id: ObjectId('6889b65faae007e12ffdac55')
topic : "React Components"
date : 2020-10-15T00:00:00.000+00:00
```

Tasks:

These are the sample data from the ‘tasks’ collection .

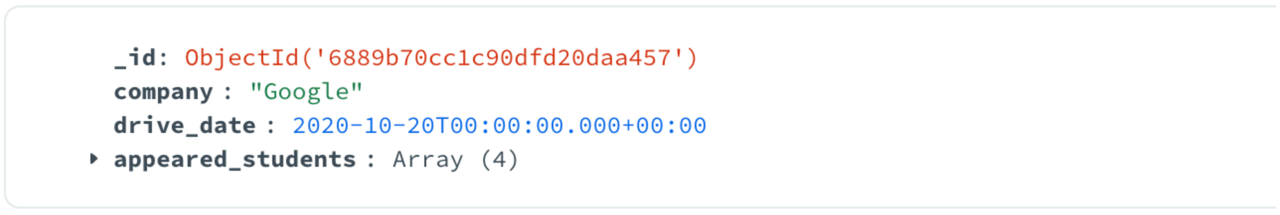


```
  _id: ObjectId('6889b677aae007e12ffdac58')
  task_name : "JS Exercises"
  topic_name : "JavaScript Basics"
  date : 2020-09-10T00:00:00.000+00:00
▸ submitted_by : Array (empty)
```

```
  _id: ObjectId('6889b677aae007e12ffdac59')
  task_name : "React App"
  topic_name : "React Components"
  date : 2020-09-15T00:00:00.000+00:00
▸ submitted_by : Array (empty)
```

Company Drives:

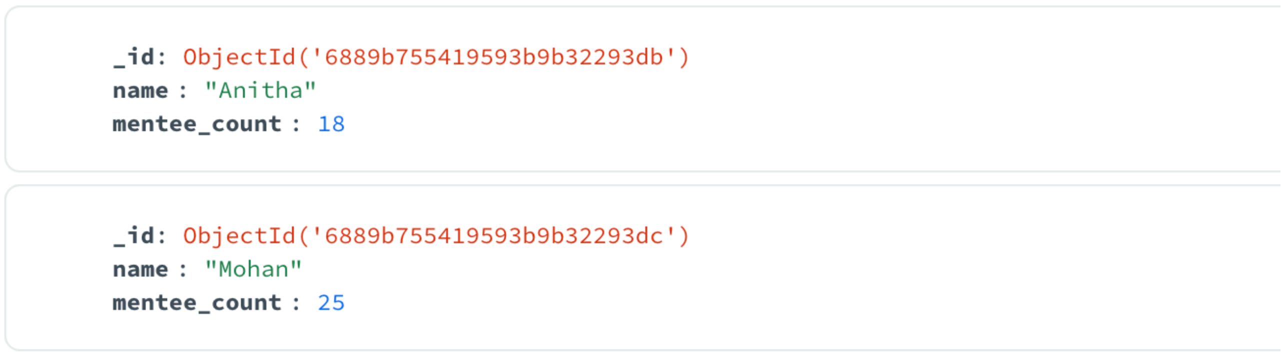
These are the sample data from the ‘Company_drives’ collection .



```
  _id: ObjectId('6889b70cc1c90dfd20daa457')
  company : "Google"
  drive_date : 2020-10-20T00:00:00.000+00:00
▸ appeared_students : Array (4)
```

Mentors:

These are the sample data from the ‘mentors’ collection .

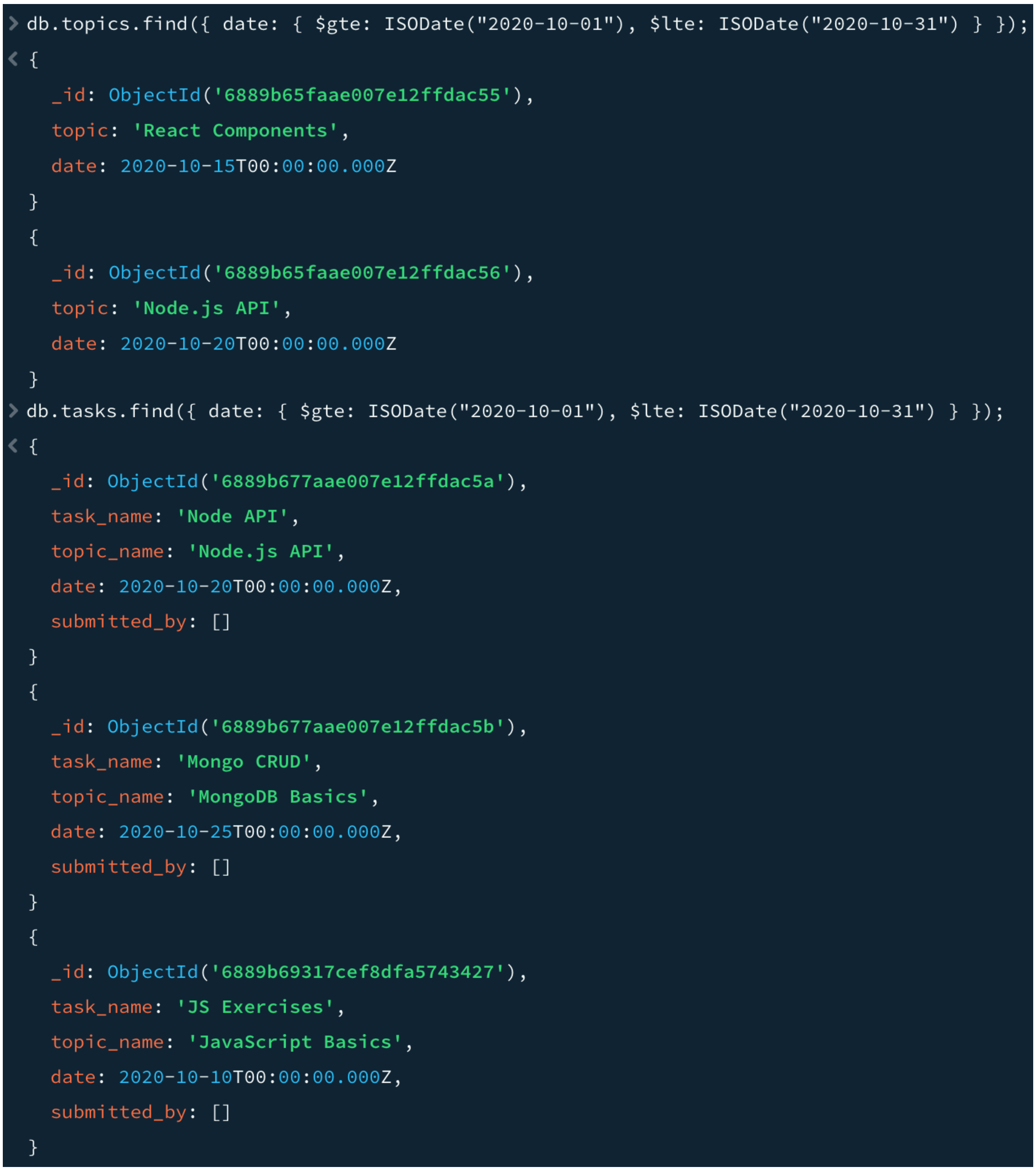


```
_id: ObjectId('6889b755419593b9b32293db')
name : "Anitha"
mentee_count : 18
```

```
_id: ObjectId('6889b755419593b9b32293dc')
name : "Mohan"
mentee_count : 25
```

1. Find all the topics and tasks which are thought in the month of October.

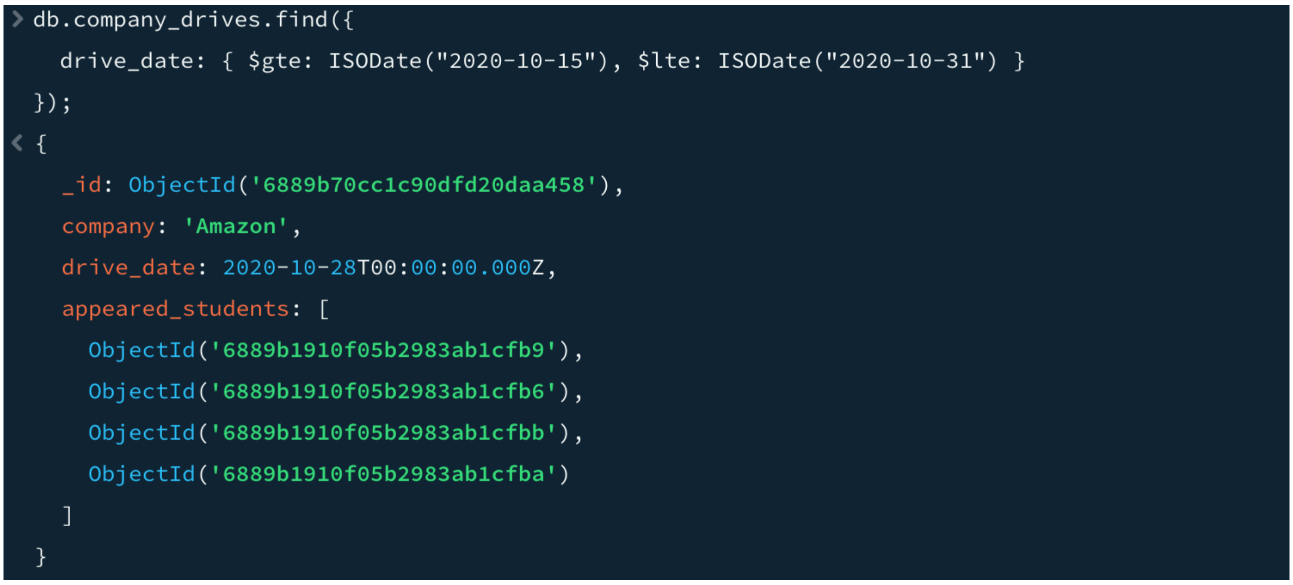
This query retrieves records from the topics and tasks collections where the date field falls between 1st October 2020 and 31st October 2020.MongoDB date comparison operators $gte (greater than or equal) and $lte (less than or equal) along with ISODate are used to filter the results within this date range.



```
> db.topics.find({ date: { $gte: ISODate("2020-10-01"), $lte: ISODate("2020-10-31") } });
< {
    _id: ObjectId('6889b65faae007e12ffdac55'),
    topic: 'React Components',
    date: 2020-10-15T00:00:00.000Z
  }
  {
    _id: ObjectId('6889b65faae007e12ffdac56'),
    topic: 'Node.js API',
    date: 2020-10-20T00:00:00.000Z
  }
> db.tasks.find({ date: { $gte: ISODate("2020-10-01"), $lte: ISODate("2020-10-31") } });
< {
    _id: ObjectId('6889b677aae007e12ffdac5a'),
    task_name: 'Node API',
    topic_name: 'Node.js API',
    date: 2020-10-20T00:00:00.000Z,
    submitted_by: []
  }
  {
    _id: ObjectId('6889b677aae007e12ffdac5b'),
    task_name: 'Mongo CRUD',
    topic_name: 'MongoDB Basics',
    date: 2020-10-25T00:00:00.000Z,
    submitted_by: []
  }
  {
    _id: ObjectId('6889b69317cef8dfa5743427'),
    task_name: 'JS Exercises',
    topic_name: 'JavaScript Basics',
    date: 2020-10-10T00:00:00.000Z,
    submitted_by: []
  }
```

2. Find all the company drives which appeared between 15 oct-2020 and 31-oct-2020.

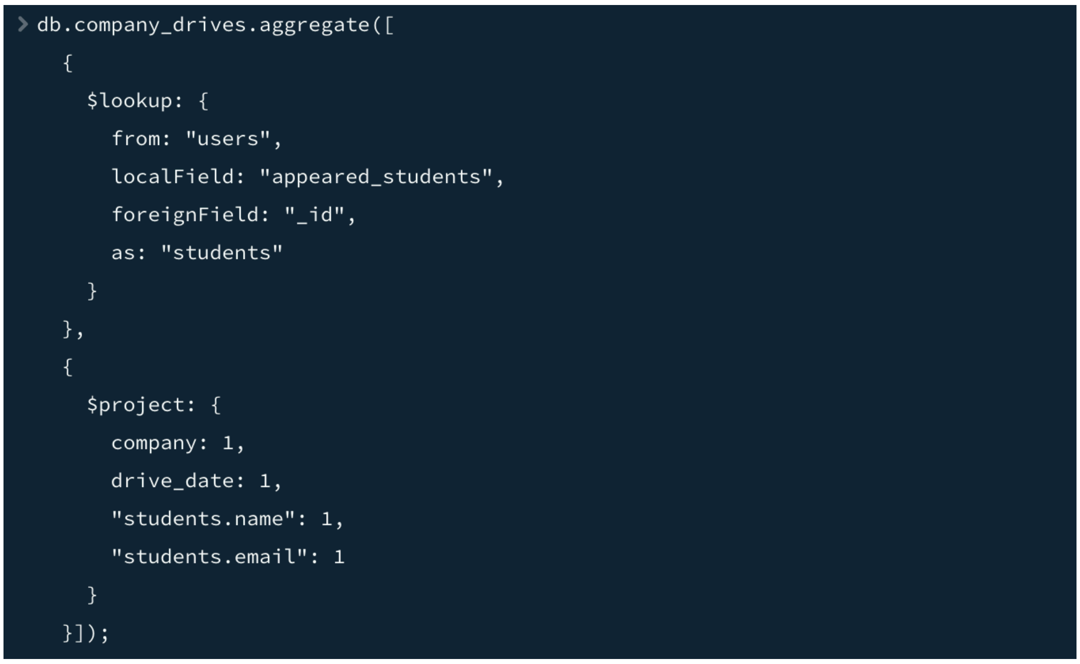
This query retrieves all documents from the company_drives collection where drive_date is between 15th October 2020 and 31st October 2020, using $gte and $lte for the date range filter.



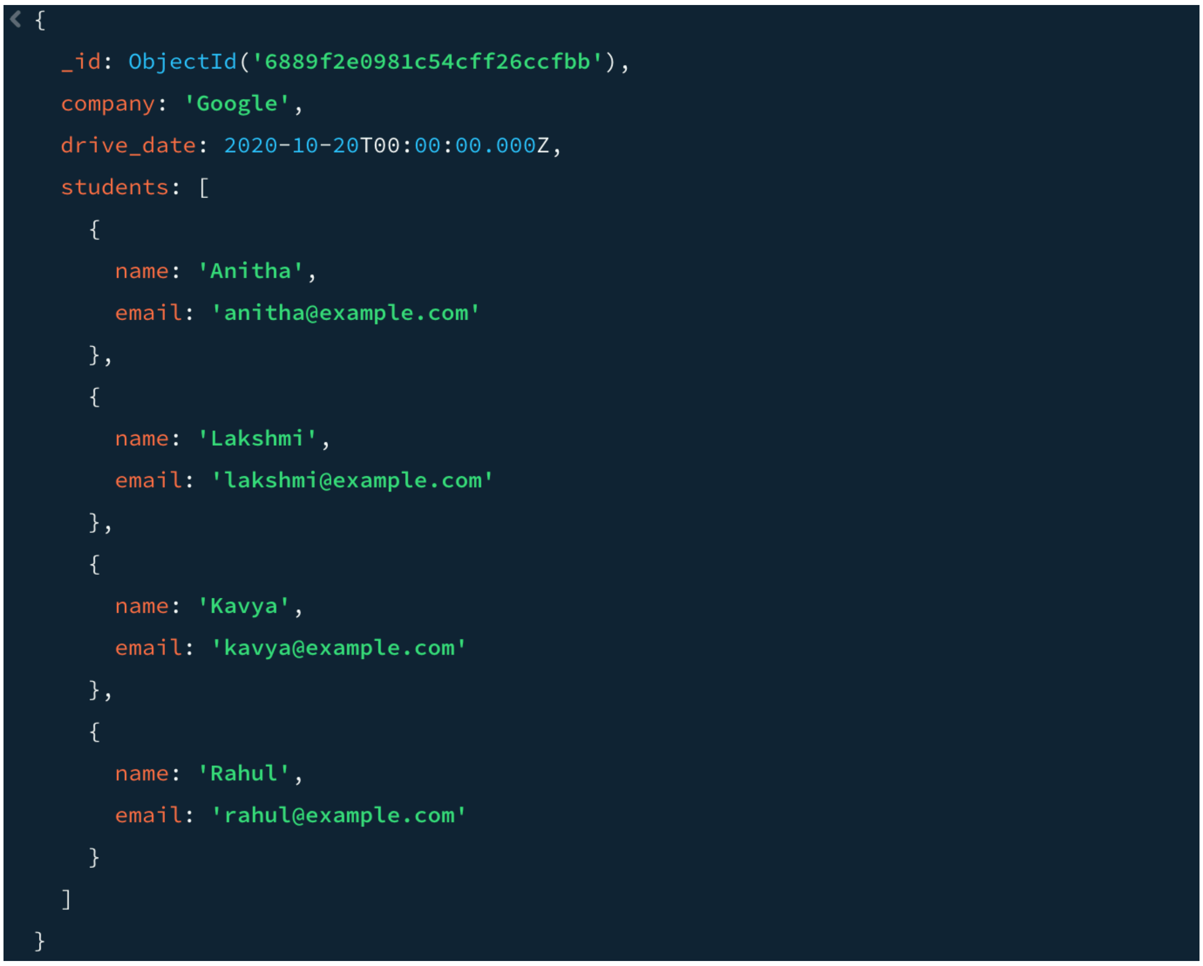
```
> db.company_drives.find({
    drive_date: { $gte: ISODate("2020-10-15"), $lte: ISODate("2020-10-31") }
  });
< {
    _id: ObjectId('6889b70cc1c90dfd20daa458'),
    company: 'Amazon',
    drive_date: 2020-10-28T00:00:00.000Z,
    appeared_students: [
      ObjectId('6889b1910f05b2983ab1cfb9'),
      ObjectId('6889b1910f05b2983ab1cfb6'),
      ObjectId('6889b1910f05b2983ab1cfbb'),
      ObjectId('6889b1910f05b2983ab1cfba')
    ]
  }
```

3. Find all the company drives and students who are appeared for the placement.

This query uses $lookup to join company_drives and users collections based on matching appeared_students with _id. It retrieves the company name, drive date, and names of students who appeared for each drive.



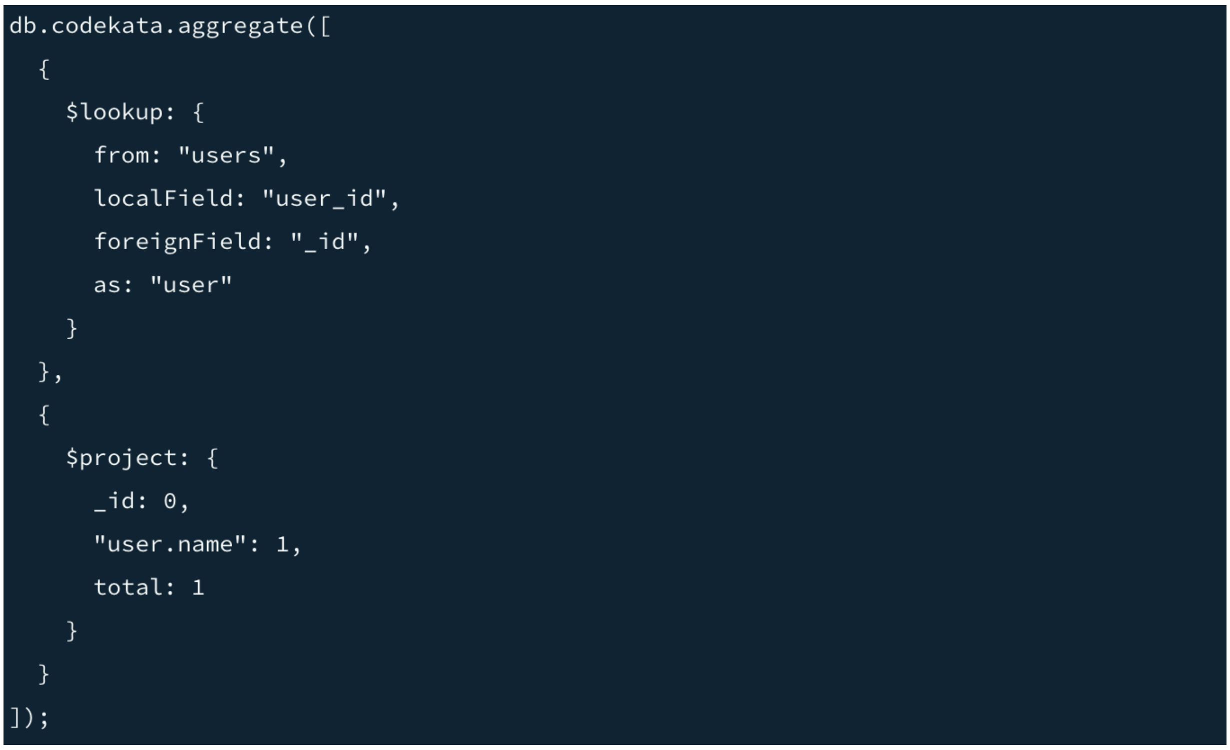
```
> db.company_drives.aggregate([
    {
      $lookup: {
        from: "users",
        localField: "appeared_students",
        foreignField: "_id",
        as: "students"
      }
    },
    {
      $project: {
        company: 1,
        drive_date: 1,
        "students.name": 1,
        "students.email": 1
      }
    }]);
```



```
{
  _id: ObjectId('6889f2e0981c54cff26ccfbb'),
  company: 'Google',
  drive_date: 2020-10-20T00:00:00.000Z,
  students: [
    {
      name: 'Anitha',
      email: 'anitha@example.com'
    },
    {
      name: 'Lakshmi',
      email: 'lakshmi@example.com'
    },
    {
      name: 'Kavya',
      email: 'kavya@example.com'
    },
    {
      name: 'Rahul',
      email: 'rahul@example.com'
    }
  ]
}
```

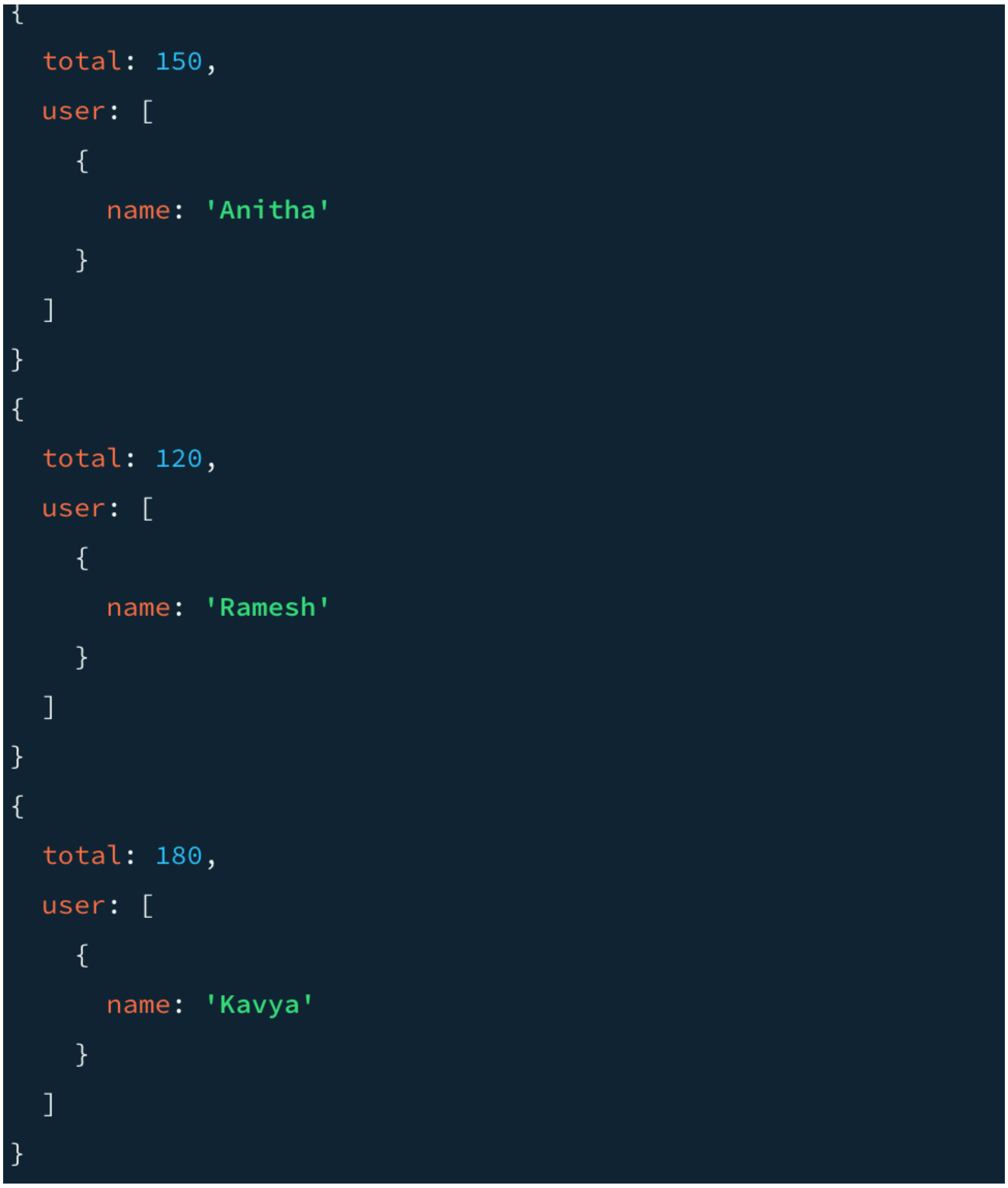
The above aggregate command is used to get this result which shows the students appeared for respective company drive.

4. Find the number of problems solved by the user in codekata



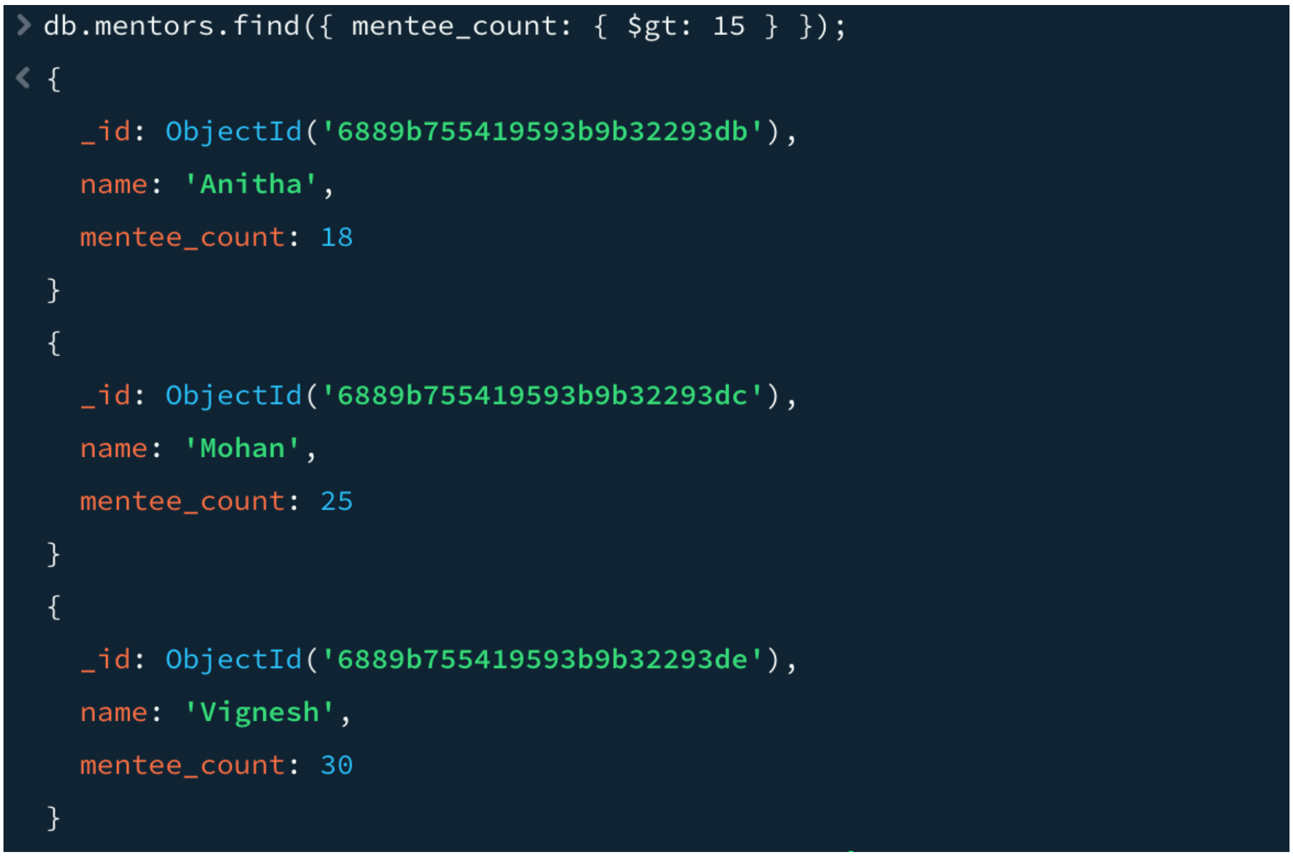
```
db.codekata.aggregate([
  {
    $lookup: {
      from: "users",
      localField: "user_id",
      foreignField: "_id",
      as: "user"
    }
  },
  {
    $project: {
      _id: 0,
      "user.name": 1,
      total: 1
    }
  }
]);
```

This query joins the codekata and users collections using $lookup to find the number of problems solved by each user. It retrieves the user's name along with the total problems solved.



```
{
  total: 150,
  user: [
    {
      name: 'Anitha'
    }
  ]
}
{
  total: 120,
  user: [
    {
      name: 'Ramesh'
    }
  ]
}
{
  total: 180,
  user: [
    {
      name: 'Kavya'
    }
  ]
}
```

5. Find all the mentors with who has the mentee's count more than 15

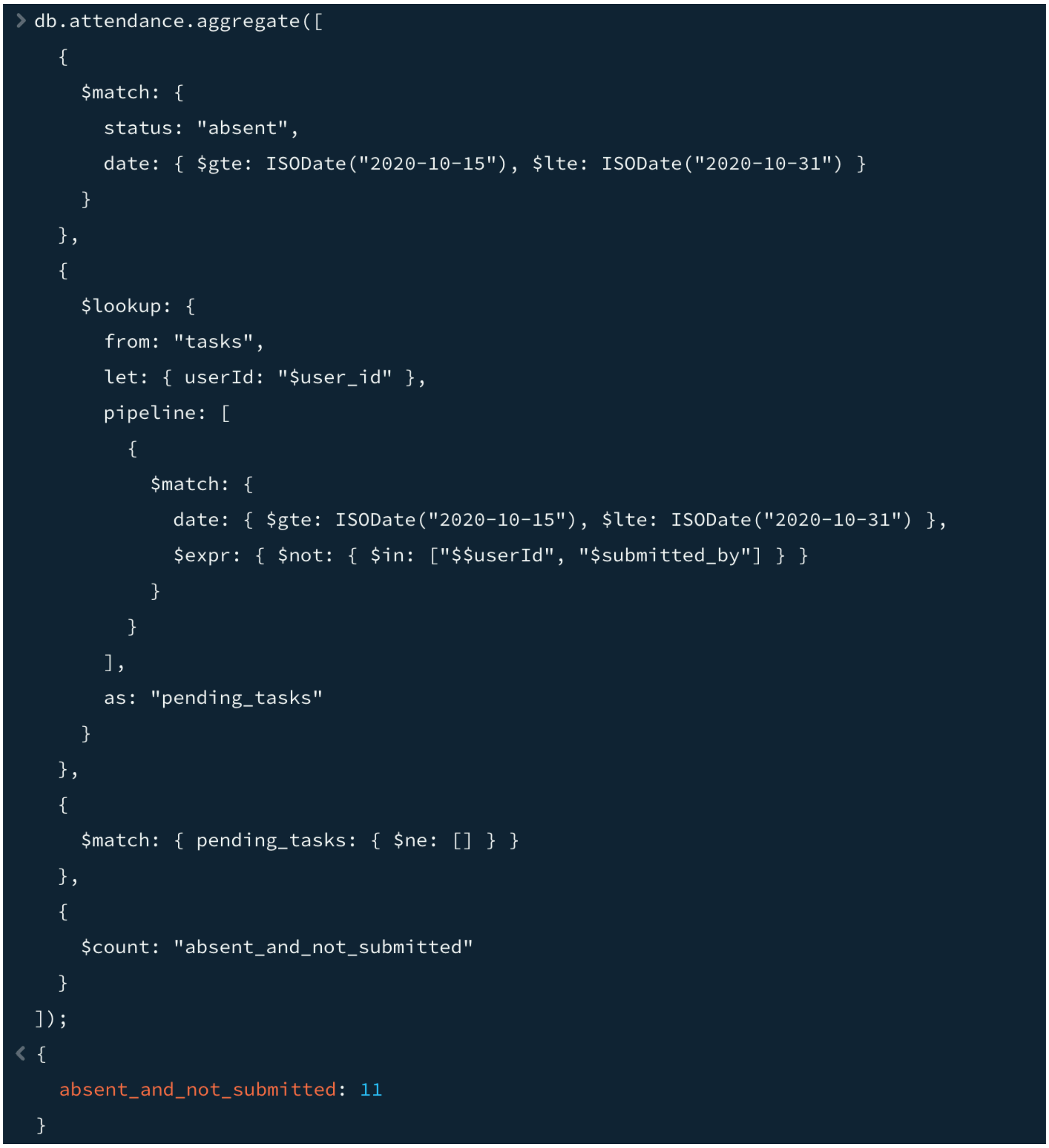


```
> db.mentors.find({ mentee_count: { $gt: 15 } });
< {
    _id: ObjectId('6889b755419593b9b32293db'),
    name: 'Anitha',
    mentee_count: 18
  }
  {
    _id: ObjectId('6889b755419593b9b32293dc'),
    name: 'Mohan',
    mentee_count: 25
  }
  {
    _id: ObjectId('6889b755419593b9b32293de'),
    name: 'Vignesh',
    mentee_count: 30
  }
```

This query retrieves all mentors from the mentors collection whose mentee_count is greater than 15 using the $gt operator.

6. Find the number of users who are absent and task is not submitted between 15 oct-2020 and 31-oct-2020

This query finds the number of students who were absent and did not submit their tasks between 15-Oct-2020 and 31-Oct-2020. It checks the attendance collection for absentees within the date range and verifies in the tasks collection that these users are not listed in the submitted_by array. Finally, it counts those users.



```
> db.attendance.aggregate([
    {
      $match: {
        status: "absent",
        date: { $gte: ISODate("2020-10-15"), $lte: ISODate("2020-10-31") }
      }
    },
    {
      $lookup: {
        from: "tasks",
        let: { userId: "$user_id" },
        pipeline: [
          {
            $match: {
              date: { $gte: ISODate("2020-10-15"), $lte: ISODate("2020-10-31") },
              $expr: { $not: { $in: ["$$userId", "$submitted_by"] } }
            }
          }
        ],
        as: "pending_tasks"
      }
    },
    {
      $match: { pending_tasks: { $ne: [] } }
    },
    {
      $count: "absent_and_not_submitted"
    }
  ]);
< {
    absent_and_not_submitted: 11
  }
```