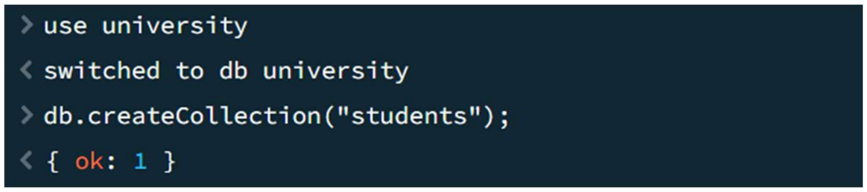
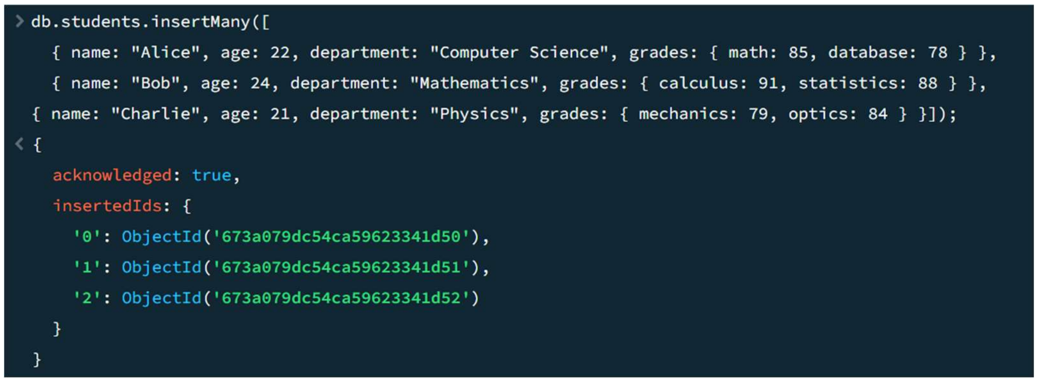
MongoDB Assignment 2

1. Create a database named university and a collection named students. Insert multiple student documents with fields: name, age, department, and grades.

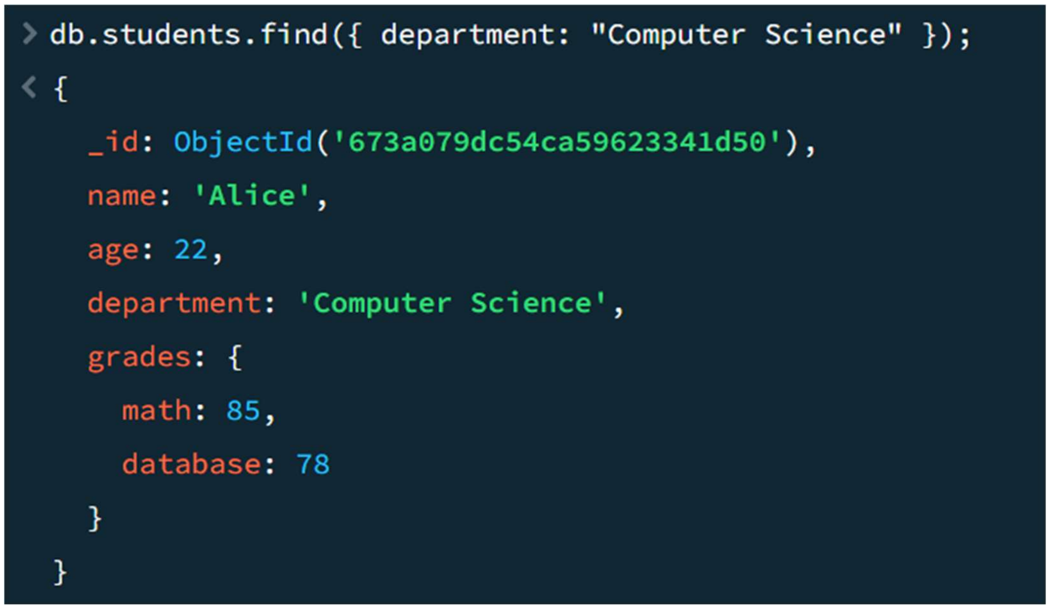


```
> use university
< switched to db university
> db.createCollection("students");
< { ok: 1 }
```



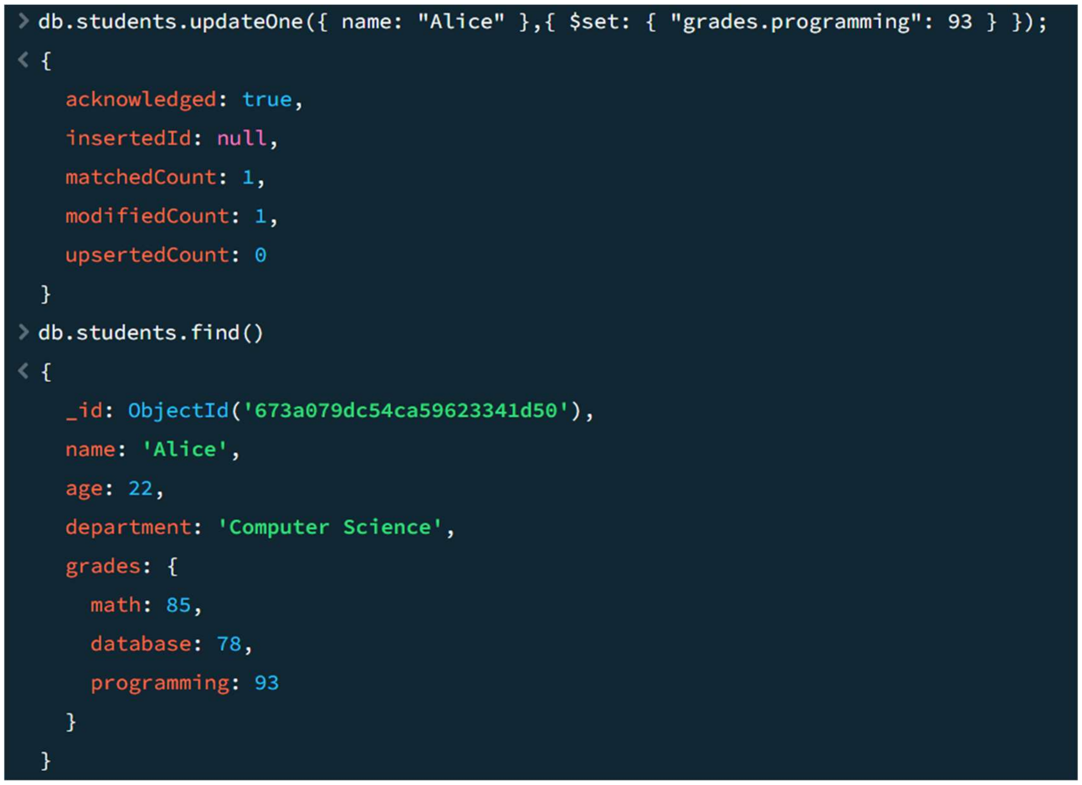
```
> db.students.insertMany([
    { name: "Alice", age: 22, department: "Computer Science", grades: { math: 85, database: 78 } },
    { name: "Bob", age: 24, department: "Mathematics", grades: { calculus: 91, statistics: 88 } },
  { name: "Charlie", age: 21, department: "Physics", grades: { mechanics: 79, optics: 84 } }]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('673a079dc54ca59623341d50'),
      '1': ObjectId('673a079dc54ca59623341d51'),
      '2': ObjectId('673a079dc54ca59623341d52')
    }
  }
```

2. Write a query to display all students who are in the Computer Science department.



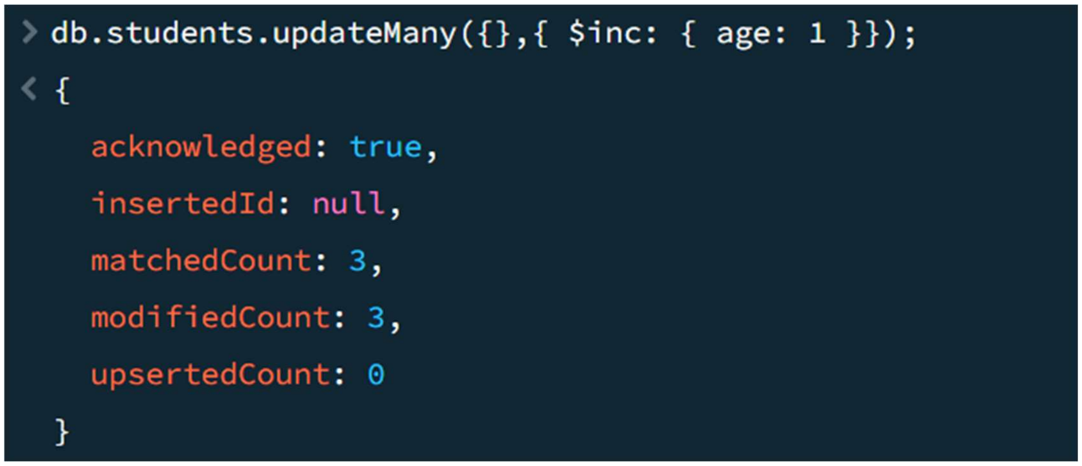
```
> db.students.find({ department: "Computer Science" });
< {
    _id: ObjectId('673a079dc54ca59623341d50'),
    name: 'Alice',
    age: 22,
    department: 'Computer Science',
    grades: {
      math: 85,
      database: 78
    }
  }
```

3. Write a query to update the grades of a student named Alice by adding a new subject programming with a grade of 93.

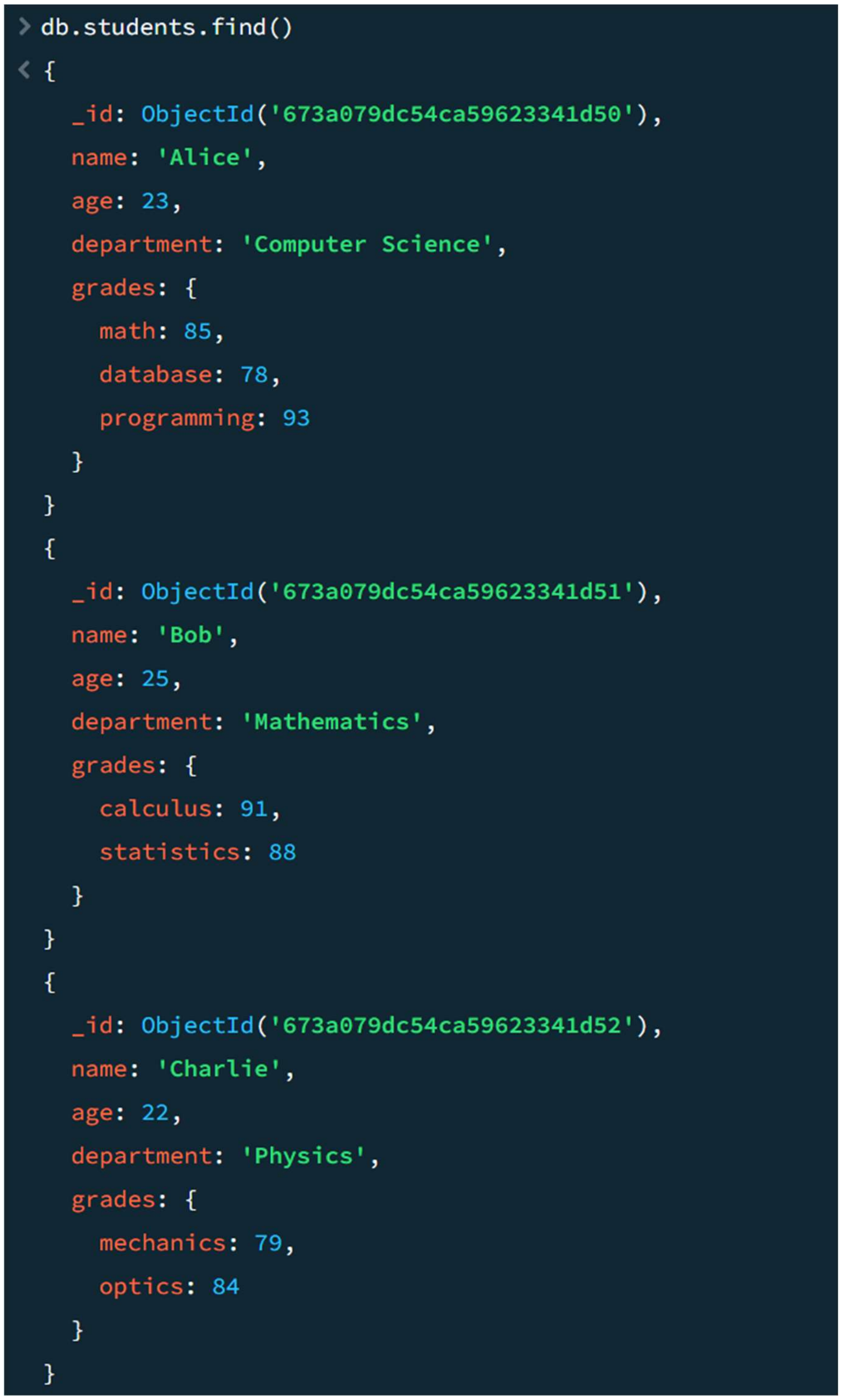


```
> db.students.updateOne({ name: "Alice" },{ $set: { "grades.programming": 93 } });
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.students.find()
< {
    _id: ObjectId('673a079dc54ca59623341d50'),
    name: 'Alice',
    age: 22,
    department: 'Computer Science',
    grades: {
      math: 85,
      database: 78,
      programming: 93
    }
  }
```

4. Write a query to increment the age of all students by 1.

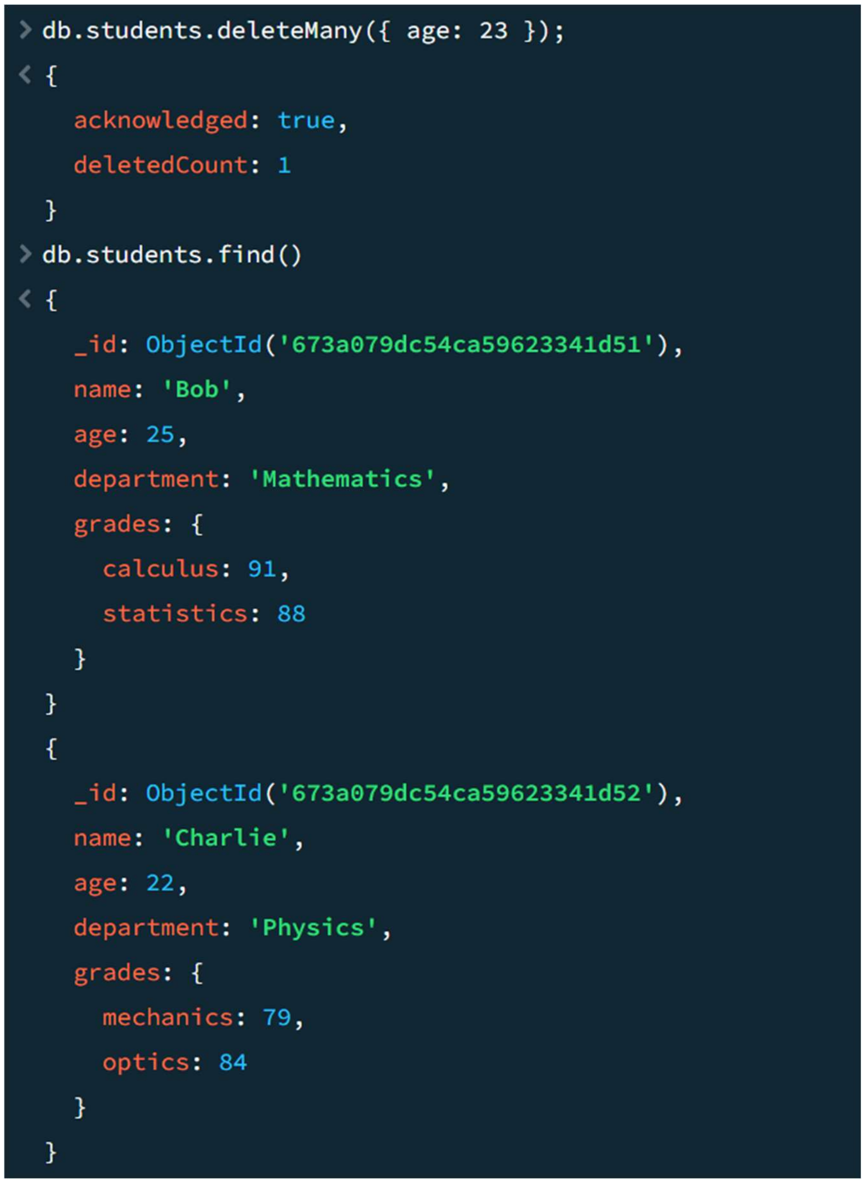


```
> db.students.updateMany({},{ $inc: { age: 1 }});
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 3,
    modifiedCount: 3,
    upsertedCount: 0
  }
```



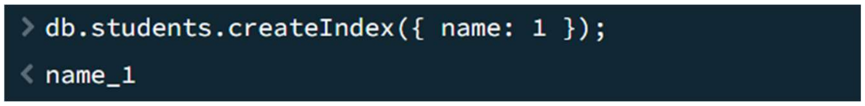
```
> db.students.find()
< {
    _id: ObjectId('673a079dc54ca59623341d50'),
    name: 'Alice',
    age: 23,
    department: 'Computer Science',
    grades: {
      math: 85,
      database: 78,
      programming: 93
    }
  }
  {
    _id: ObjectId('673a079dc54ca59623341d51'),
    name: 'Bob',
    age: 25,
    department: 'Mathematics',
    grades: {
      calculus: 91,
      statistics: 88
    }
  }
  {
    _id: ObjectId('673a079dc54ca59623341d52'),
    name: 'Charlie',
    age: 22,
    department: 'Physics',
    grades: {
      mechanics: 79,
      optics: 84
    }
  }
```

5. Write a query to delete all students who are 23 years old.



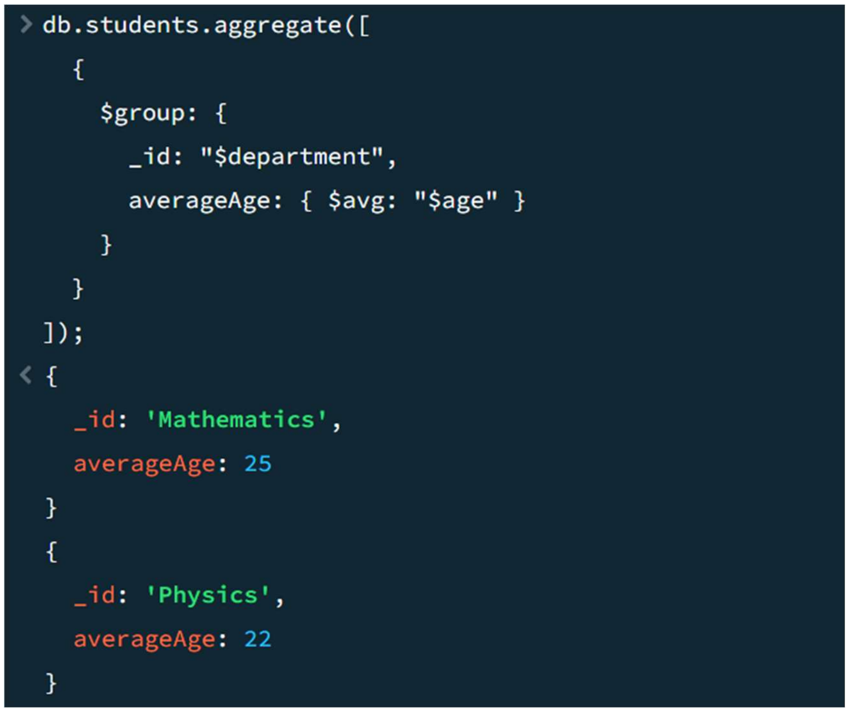
```
> db.students.deleteMany({ age: 23 });
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.students.find()
< {
    _id: ObjectId('673a079dc54ca59623341d51'),
    name: 'Bob',
    age: 25,
    department: 'Mathematics',
    grades: {
      calculus: 91,
      statistics: 88
    }
  }
  {
    _id: ObjectId('673a079dc54ca59623341d52'),
    name: 'Charlie',
    age: 22,
    department: 'Physics',
    grades: {
      mechanics: 79,
      optics: 84
    }
  }
```

6. Write a query to create an index on the name field of the students collection.



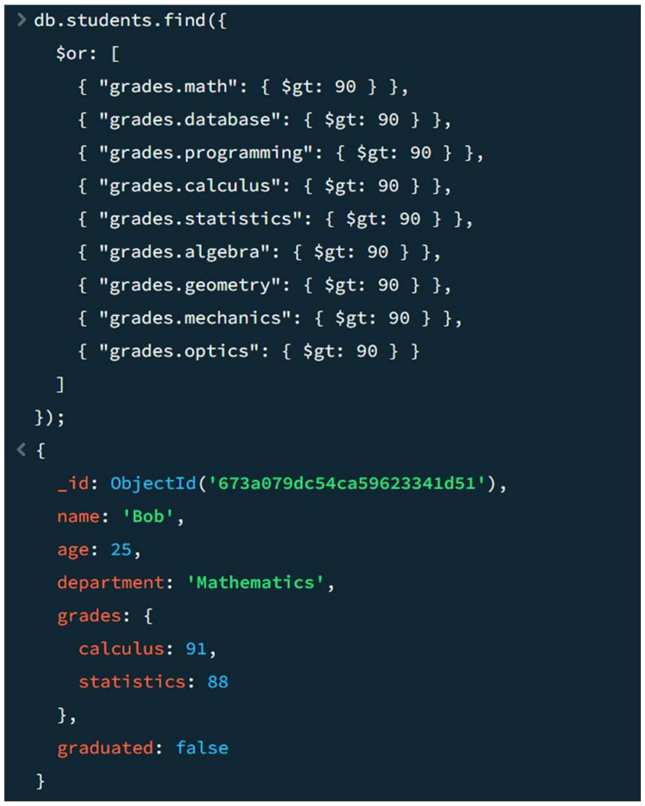
```
> db.students.createIndex({ name: 1 });
< name_1
```

7. Write an aggregation query to group students by their department and calculate the average age in each department.



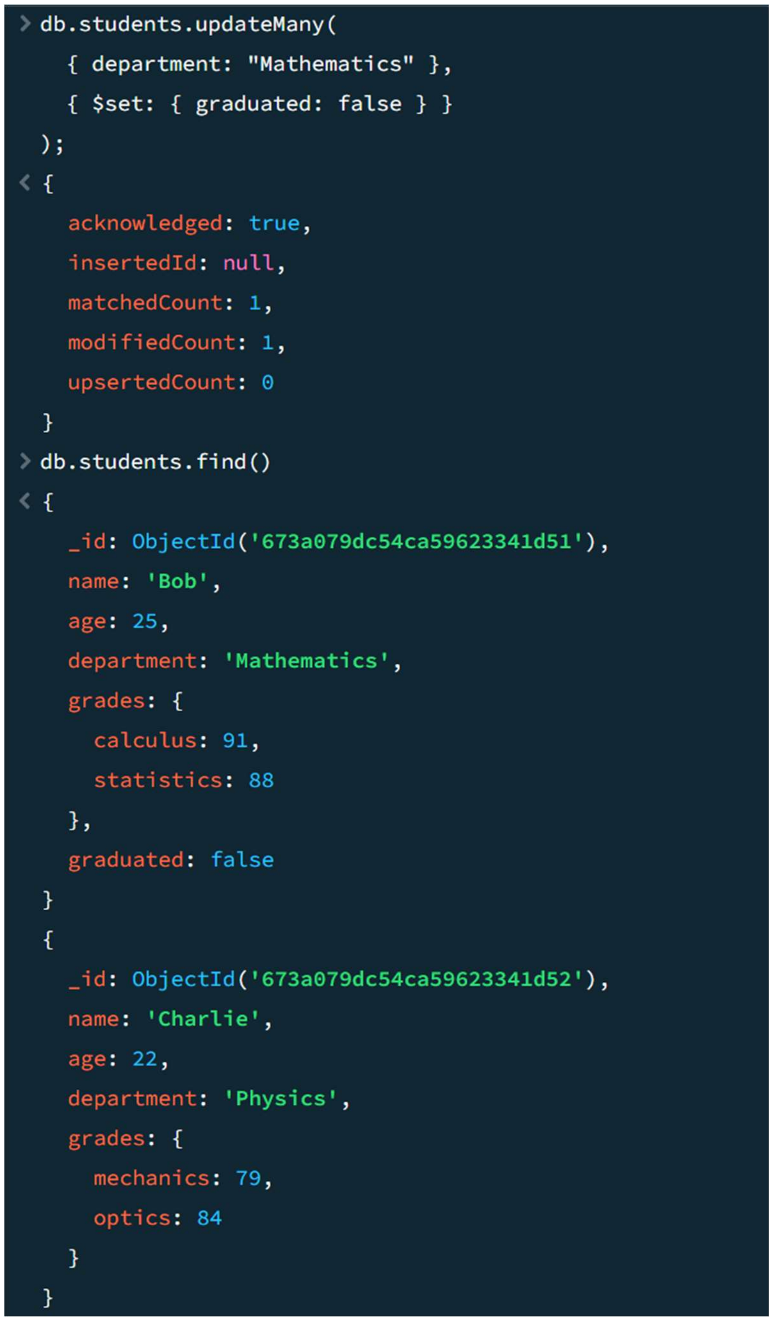
```
> db.students.aggregate([
    {
      $group: {
        _id: "$department",
        averageAge: { $avg: "$age" }
      }
    }
  ]);
< {
    _id: 'Mathematics',
    averageAge: 25
  }
  {
    _id: 'Physics',
    averageAge: 22
  }
```

8. Write a query to find all students who have scored more than 90 in any subject.



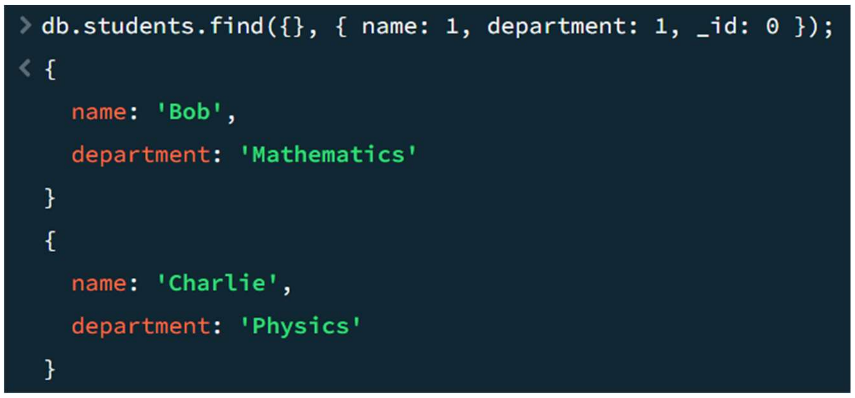
```
> db.students.find({
    $or: [
      { "grades.math": { $gt: 90 } },
      { "grades.database": { $gt: 90 } },
      { "grades.programming": { $gt: 90 } },
      { "grades.calculus": { $gt: 90 } },
      { "grades.statistics": { $gt: 90 } },
      { "grades.algebra": { $gt: 90 } },
      { "grades.geometry": { $gt: 90 } },
      { "grades.mechanics": { $gt: 90 } },
      { "grades.optics": { $gt: 90 } }
    ]
  });
< {
    _id: ObjectId('673a079dc54ca59623341d51'),
    name: 'Bob',
    age: 25,
    department: 'Mathematics',
    grades: {
      calculus: 91,
      statistics: 88
    },
    graduated: false
  }
```

9. Write a query to add a new field graduated set to false for all students who are in the Mathematics department.



```
> db.students.updateMany(
    { department: "Mathematics" },
    { $set: { graduated: false } }
  );
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.students.find()
< {
    _id: ObjectId('673a079dc54ca59623341d51'),
    name: 'Bob',
    age: 25,
    department: 'Mathematics',
    grades: {
      calculus: 91,
      statistics: 88
    },
    graduated: false
  }
  {
    _id: ObjectId('673a079dc54ca59623341d52'),
    name: 'Charlie',
    age: 22,
    department: 'Physics',
    grades: {
      mechanics: 79,
      optics: 84
    }
  }
```

10. How can you retrieve only the name and department fields for all students, excluding the _id field?



```
> db.students.find({}, { name: 1, department: 1, _id: 0 });
< {
    name: 'Bob',
    department: 'Mathematics'
  }
  {
    name: 'Charlie',
    department: 'Physics'
  }
```