BAHRIA UNIVERSITY (KARACHI CAMPUS)

Assignment-03

(Big Data Analytics)

Class: **BSE [4]-7 (B)** **(Morning)**

Course Instructor: **Dr. Salahuddin Shaikh** Submission Date: **23/12/2024**

Date: **12/12/2024)** Max Marks**:** **10 M (CLO2-3)**

Student's Name: **Muhammad Shoaib Akhter Qadri** Reg. No: **79290____**

MongoDB Task: Advanced Student Management System

Consider a MongoDB collection named **"students"** that stores information about students enrolled in a university. Each document represents a student and contains the following fields:

- **_id**: Unique identifier for the student.
- **name**: Full name of the student.
- **age**: Age of the student.
- **department**: Department the student belongs to (e.g., Computer Science, Business, Engineering).
- **gpa**: Current GPA of the student.
- **courses**: An array containing the list of courses the student is enrolled in. Each course includes:
    - **course_name**: Name of the course.
    - **grade**: Grade obtained in the course.

Tasks for Applying Advanced MongoDB Queries:

1. Insert five new student documents, ensuring at least one student has a GPA of exactly 4.0 and at least three courses. Verify the insertions by retrieving the students with the highest GPA.

Solution:

```
db.students.insertMany([
  {
    name: "Shoaib Akhter",
    age: 21,
    department: "Computer Science",
    gpa: 3.2,
    courses: [
      { course_name: "Algorithms", grade: "B" },
      { course_name: "Data Structures", grade: "B" }
    ]
  },
  {
```

```
    name: "Emily Davis",
    age: 23,
    department: "Business",
    gpa: 3.8,
    courses: [
      { course_name: "Marketing 101", grade: "A" },
      { course_name: "Accounting Basics", grade: "B" }
    ]
  },
  {
    name: "Michael Smith",
    age: 24,
    department: "Engineering",
    gpa: 3.7,
    courses: [
      { course_name: "Thermodynamics", grade: "A" },
      { course_name: "Fluid Mechanics", grade: "B" },
      { course_name: "Calculus II", grade: "B" }
    ]
  },
  {
    name: "Aisha Ali",
    age: 22,
    department: "Mathematics",
    gpa: 4.0, // exactly 4.0
    courses: [
      { course_name: "Linear Algebra", grade: "A" },
      { course_name: "Real Analysis", grade: "A" },
      { course_name: "Complex Analysis", grade: "A" }
    ]
  },
  {
    name: "David Kim",
    age: 20,
    department: "Computer Science",
    gpa: 3.8,
    courses: [
      { course_name: "Intro to Programming", grade: "B" },
      { course_name: "Operating Systems", grade: "C" },
      { course_name: "Computer Architecture", grade: "B" }
    ]
  }
]);

db.students.find().sort({ gpa: -1 }).limit(1);
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... ... ... ...
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6767b6bdc3a0070cf06b128c'),
    '1': ObjectId('6767b6bdc3a0070cf06b128d'),
    '2': ObjectId('6767b6bdc3a0070cf06b128e'),
    '3': ObjectId('6767b6bdc3a0070cf06b128f'),
    '4': ObjectId('6767b6bdc3a0070cf06b1290')
  }
}
mycompiler_mongodb>
mycompiler_mongodb>
```

```
mycompiler_mongodb> [
  {
    _id: ObjectId('6767b6bdc3a0070cf06b128f'),
    name: 'Aisha Ali',
    age: 22,
    department: 'Mathematics',
    gpa: 4,
    courses: [
      { course_name: 'Linear Algebra', grade: 'A' },
      { course_name: 'Real Analysis', grade: 'A' },
      { course_name: 'Complex Analysis', grade: 'A' }
    ]
  }
]
mycompiler_mongodb>
```

2. Retrieve all students from the "Computer Science" department who have taken more than two courses and a GPA greater than 3.5.

Solution:

```
db.students.find({
  department: "Computer Science",
  gpa: { $gt: 3.5 },
  $expr: { $gt: [{ $size: "$courses" }, 2] }
});
```

Output:

```
mycompiler_mongodb> ... ... ... ... [
  {
    _id: ObjectId('6767b8b73dc2c970006b1290'),
    name: 'David Kim',
    age: 20,
    department: 'Computer Science',
    gpa: 3.8,
    courses: [
      { course_name: 'Intro to Programming', grade: 'B' },
      { course_name: 'Operating Systems', grade: 'C' },
      { course_name: 'Computer Architecture', grade: 'B' }
    ]
  }
]
```

3. Update all students aged 22 or below by increasing their GPA by 0.3 **and** adding a new course titled "Ethics" with a grade of "A".

Solution:

```
db.students.updateMany(
  { age: { $lte: 22 } },
  {
    $inc: { gpa: 0.3 },
    $push: { courses: { course_name: "Ethics", grade: "A" } }
  }
);
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... {
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>

[Execution complete with exit code 0]
```

4. Remove the lowest grade from the courses array **and** reduce the GPA by 0.1 for students who have more than three courses.

Solution:

```
const studentsWithManyCourses = db.students.find({
  $expr: { $gt: [{ $size: "$courses" }, 3] }
}).toArray();
```

```
const gradeMap = { A: 4, B: 3, C: 2, D: 1, F: 0 };

studentsWithManyCourses.forEach(student => {
  let lowestCourse = null;
  let lowestScore = Infinity;

  student.courses.forEach(crs => {
    const score = gradeMap[crs.grade] ?? 0;
    if (score < lowestScore) {
      lowestScore = score;
      lowestCourse = crs;
    }
  });

  if (lowestCourse) {
    db.students.updateOne(
      { _id: student._id },
      {
        $pull: { courses: { course_name: lowestCourse.course_name, grade:
lowestCourse.grade } },
        $inc: { gpa: -0.1 }
      }
    );
  }
});
```

Output:

```
mycompiler_mongodb> {
  _id: ObjectId('6767c663b2c8d54b7c6b128c'),
  name: 'Shoaib Akhter',
  age: 21,
  department: 'Computer Science',
  gpa: 3.2,
  courses: [
    {
      course_name: 'Algorithms',
      grade: 'B'
    },
    {
      course_name: 'Data Structures',
      grade: 'B'
    }
  ]
```

```
}
{
  _id: ObjectId('6767c663b2c8d54b7c6b128d'),
  name: 'Emily Davis',
  age: 23,
  department: 'Business',
  gpa: 3.8,
  courses: [
    {
      course_name: 'Marketing 101',
      grade: 'C'
    },
    {
      course_name: 'Accounting Basics',
      grade: 'C'
    }
  ]
}
{
  _id: ObjectId('6767c663b2c8d54b7c6b128e'),
  name: 'Michael Smith',
  age: 35,
  department: 'Engineering',
  gpa: 3.9,
  courses: [
    {
      course_name: 'Thermodynamics',
      grade: 'A'
    },
    {
      course_name: 'Fluid Mechanics',
      grade: 'A'
    }
  ]
}
{
  _id: ObjectId('6767c663b2c8d54b7c6b128f'),
  name: 'Aisha Ali',
  age: 22,
  department: 'Mathematics',
  gpa: 4,
  courses: [
    {
      course_name: 'Linear Algebra',
      grade: 'A'
    },
    {
      course_name: 'Real Analysis',
      grade: 'A'
```

```
  ]
}
{
  _id: ObjectId('6767c664b2c8d54b7c6b1290'),
  name: 'David Kim',
  age: 25,
  department: 'Computer Science',
  gpa: 3.8,
  courses: [
    {
      course_name: 'Intro to Programming',
      grade: 'A'
    },
    {
      course_name: 'Math',
      grade: 'C'
    },
    {
      course_name: 'Computer Architecture',
      grade: 'B'
    }
  ]
}
{
  _id: ObjectId('6767c664b2c8d54b7c6b1291'),
  name: 'Sarah Lee',
  age: 22,
  department: 'Business',
  gpa: 2.9,
  courses: [
```

```
  _id: ObjectId('6767c664b2c8d54b7c6b1291'),
  name: 'Sarah Lee',
  age: 22,
  department: 'Business',
  gpa: 2.9,
  courses: [
    {
      course_name: 'Business Law',
      grade: 'F'
    },
    {
      course_name: 'Business Ethics',
      grade: 'D'
    }
  ]
}
```

5. Find the student with the highest total grades across all courses **and** add a new course titled "Leadership" with a grade of "A".

Solution:

```
const pipeline = [
  {
    $addFields: {
      totalGradePoints: {
        $sum: {
          $map: {
            input: "$courses",
            as: "crs",
            in: {
              $switch: {
                branches: [
                  { case: { $eq: ["$$crs.grade", "A"] }, then: 4 },
                  { case: { $eq: ["$$crs.grade", "B"] }, then: 3 },
                  { case: { $eq: ["$$crs.grade", "C"] }, then: 2 },
                  { case: { $eq: ["$$crs.grade", "D"] }, then: 1 },
                ],
                default: 0
              }
            }
          }
        }
      }
    }
  },
  { $sort: { totalGradePoints: -1 } },
  { $limit: 1 }
```

```
];

const topStudent = db.students.aggregate(pipeline).toArray()[0];
if (topStudent) {
  db.students.updateOne(
    { _id: topStudent._id },
    {
      $push: { courses: { course_name: "Leadership", grade: "A" } }
    }
  );
}
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ... {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mycompiler_mongodb>
mycompiler_mongodb>
```

6. Calculate the average GPA of students grouped by their department **and** include the total number of students in each department.

Solution:

```
db.students.aggregate([
  {
    $group: {
      _id: "$department",
      averageGPA: { $avg: "$gpa" },
      totalStudents: { $sum: 1 }
    }
  }
]);
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ... ... [
  { _id: 'Engineering', averageGPA: 3.7, totalStudents: 1 },
  { _id: 'Computer Science', averageGPA: 3.5, totalStudents: 2 },
  { _id: 'Mathematics', averageGPA: 4, totalStudents: 1 },
  { _id: 'Business', averageGPA: 3.8, totalStudents: 1 }
]
mycompiler_mongodb>
```

7. Retrieve the names of students enrolled in exactly two courses **and** sort them by GPA in descending order.

Solution:

```
db.students.find(
  {
    $expr: { $eq: [{ $size: "$courses" }, 2] }
  },
  { name: 1, gpa: 1, _id: 0 }
).sort({ gpa: -1 });
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... [
  { name: 'Emily Davis', gpa: 3.8 },
  { name: 'Shoaib Akhter', gpa: 3.2 }
]
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
```

8. Find and update the student with the lowest GPA by adding a new course titled "Programming Basics" with a grade of "A" **and** setting their GPA to at least 2.5 if it's currently below that.

Solution:

```
const lowestGPAStudent = db.students.find().sort({ gpa: 1 }).limit(1).toArray()[0];

if (lowestGPAStudent) {
  db.students.updateOne(
    { _id: lowestGPAStudent._id },
    {
      $push: { courses: { course_name: "Programming Basics", grade: "A" } },
      $set: {
        gpa: Math.max(lowestGPAStudent.gpa, 2.5)
      }
    }
  );
}
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ...
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mycompiler_mongodb>
mycompiler_mongodb>
```

9. Delete students from the "Business" department whose average course grade is below "C" **and** have fewer than three courses.

Solution:

```
db.students.insertMany([
  {
    name: "Shoaib Akhter",
    age: 21,
    department: "Computer Science",
    gpa: 3.2,
    courses: [
      { course_name: "Algorithms", grade: "B" },
      { course_name: "Data Structures", grade: "B" }
    ]
  },
  {
    name: "Emily Davis",
```

```
  age: 23,
  department: "Business",
  gpa: 3.8,
  courses: [
    { course_name: "Marketing 101", grade: "C" },
    { course_name: "Accounting Basics", grade: "C" }
  ]
},
{
  name: "Michael Smith",
  age: 35,
  department: "Engineering",
  gpa: 4.0,
  courses: [
    { course_name: "Thermodynamics", grade: "A" },
    { course_name: "Fluid Mechanics", grade: "A" },
    { course_name: "Calculus II", grade: "A" },
    { course_name: "Material Science", grade: "B" }
  ]
},
{
  name: "Aisha Ali",
  age: 22,
  department: "Mathematics",
  gpa: 4.0,
  courses: [
    { course_name: "Linear Algebra", grade: "A" },
    { course_name: "Real Analysis", grade: "A" },
    { course_name: "Complex Analysis", grade: "A" }
  ]
},
{
  name: "David Kim",
  age: 25,
  department: "Computer Science",
  gpa: 3.8,
  courses: [
    { course_name: "Intro to Programming", grade: "A" },
    { course_name: "Math", grade: "C" },
    { course_name: "Computer Architecture", grade: "B" }
  ]
},
{
  name: "Sarah Lee",
  age: 22,
  department: "Business",
  gpa: 2.9,
  courses: [
    { course_name: "Business Law", grade: "F" },
    { course_name: "Business Ethics", grade: "D" }
  ]
```

```
  }
]);
const gradeMap = { A: 4, B: 3, C: 2, D: 1, F: 0 };
const studentsToDelete = db.students.aggregate([
  { $match: { department: "Business" } },
  {
    $addFields: {
      numericGrades: {
        $map: {
          input: "$courses",
          as: "course",
          in: {
            $switch: {
              branches: [
                { case: { $eq: ["$$course.grade", "A"] }, then: 4 },
                { case: { $eq: ["$$course.grade", "B"] }, then: 3 },
                { case: { $eq: ["$$course.grade", "C"] }, then: 2 },
                { case: { $eq: ["$$course.grade", "D"] }, then: 1 },
                { case: { $eq: ["$$course.grade", "F"] }, then: 0 }
              ],
              default: 0
            }
          }
        }
      },
      courseCount: { $size: "$courses" }
    }
  },
  {
    $addFields: {
      avgNumericGrade: { $avg: "$numericGrades" }
    }
  },
  {
    $match: {
      avgNumericGrade: { $lt: 2 },
      courseCount: { $lt: 3 }
    }
  }
]).toArray(); // Convert the result to an array for processing

// Step 3: Delete the matching students
studentsToDelete.forEach(student => {
  db.students.deleteOne({ _id: student._id });
});
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> ... ...
mycompiler_mongodb>
```

10. Retrieve students aged between 20 and 25 who have taken at least one course containing the word "Math" in its name, sorted by age in ascending order.

Solution:

```
db.students.find(
  {
    age: { $gte: 20, $lte: 25 },
    "courses.course_name": /Math/i  // regex to match "Math" (case-insensitive)
  }
).sort({ age: 1 });
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... [
  {
    _id: ObjectId('6767bdf756ad7ab9e26b128f'),
    name: 'Aisha Ali',
    age: 22,
    department: 'Mathematics',
    gpa: 4,
    courses: [
      { course_name: 'Linear Algebra', grade: 'A' },
      { course_name: 'Math', grade: 'A' },
      { course_name: 'Complex Analysis', grade: 'A' }
    ]
  },
  {
    _id: ObjectId('6767bdf756ad7ab9e26b1290'),
    name: 'David Kim',
    age: 25,
    department: 'Computer Science',
    gpa: 3.8,
    courses: [
      { course_name: 'Intro to Programming', grade: 'B' },
      { course_name: 'Math', grade: 'C' },
      { course_name: 'Computer Architecture', grade: 'B' }
    ]
  }
]
```

11. Update all students in the "Engineering" department by increasing their GPA by 0.2 **and** removing any courses with grades below "B".

Solution:

```
db.students.updateMany(
  { department: "Engineering" },
  [
    {
      $set: {
        gpa: { $add: ["$gpa", 0.2] },
        courses: {
          $filter: {
            input: "$courses",
            as: "crs",
            cond: {
              $or: [
                { $eq: ["$$crs.grade", "A"] },
                { $eq: ["$$crs.grade", "B"] }
              ]
            }
          }
        }
      }
    }
  ]
);
```

Output:

```
mycompiler_mongodb> ... ... ... ... .
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mycompiler_mongodb>
```

12. Calculate the total number of courses taken by all students grouped by department **and** find the department with the maximum courses.

Solution:

```
db.students.aggregate([
  {
    $group: {
```

```
    _id: "$department",
    totalCourses: { $sum: { $size: "$courses" } }
   }
  },
  {
   $sort: { totalCourses: -1 }
  },
]);
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... ... ... [
  { _id: 'Computer Science', totalCourses: 5 },
  { _id: 'Engineering', totalCourses: 3 },
  { _id: 'Mathematics', totalCourses: 3 },
  { _id: 'Business', totalCourses: 2 }
]
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb>
```

13. Retrieve the top three students with the highest GPA **and** display their names, GPAs, and the total number of courses they are enrolled in.

Solution:

```
db.students.aggregate([
  {
   $project: {
    name: 1,
    gpa: 1,
    totalCourses: { $size: "$courses" }
   }
  },
  { $sort: { gpa: -1 } },
  { $limit: 3 }
]);
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... [
  {
    _id: ObjectId('6767bec25b31ae5b266b128f'),
    name: 'Aisha Ali',
    gpa: 4,
    totalCourses: 3
  },
  {
    _id: ObjectId('6767bec25b31ae5b266b1290'),
    name: 'David Kim',
    gpa: 3.8,
    totalCourses: 3
  },
  {
    _id: ObjectId('6767bec25b31ae5b266b128d'),
    name: 'Emily Davis',
    gpa: 3.8,
    totalCourses: 2
  }
]
```

14. Remove students who are not enrolled in any courses **and** sort the remaining students by age in descending order.

Solution:

```
db.students.deleteMany({
  $expr: { $eq: [{ $size: "$courses" }, 0] }
});
db.students.find().sort({ age: -1 });
```

Output:

```
mycompiler_mongodb> ... ... { acknowledged: true, deletedCount: 0 }
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> [
  {
    _id: ObjectId('6767befcf9aa99912e6b128e'),
    name: 'Michael Smith',
    age: 35,
    department: 'Engineering',
    gpa: 3.7,
    courses: [
      { course_name: 'Thermodynamics', grade: 'A' },
      { course_name: 'Fluid Mechanics', grade: 'B' },
      { course_name: 'Calculus II', grade: 'B' }
    ]
  },
```

```
  {
    _id: ObjectId('6767befcf9aa99912e6b1290'),
    name: 'David Kim',
    age: 25,
    department: 'Computer Science',
    gpa: 3.8,
    courses: [
      { course_name: 'Intro to Programming', grade: 'B' },
      { course_name: 'Math', grade: 'C' },
      { course_name: 'Computer Architecture', grade: 'B' }
    ]
  },
{
  _id: ObjectId('6767befcf9aa99912e6b128d'),
  name: 'Emily Davis',
  age: 23,
  department: 'Business',
  gpa: 3.8,
  courses: [
    { course_name: 'Marketing 101', grade: 'A' },
    { course_name: 'Accounting Basics', grade: 'B' }
  ]
},
  {
    _id: ObjectId('6767befcf9aa99912e6b128f'),
    name: 'Aisha Ali',
    age: 22,
    department: 'Mathematics',
    gpa: 4,
    courses: [
      { course_name: 'Linear Algebra', grade: 'A' },
      { course_name: 'Math', grade: 'A' },
      { course_name: 'Complex Analysis', grade: 'A' }
    ]
  },
  {
    _id: ObjectId('6767befcf9aa99912e6b128c'),
    name: 'Shoaib Akhter',
    age: 21,
    department: 'Computer Science',
    gpa: 3.2,
    courses: [
      { course_name: 'Algorithms', grade: 'B' },
      { course_name: 'Data Structures', grade: 'B' }
    ]
  }
]
```

15. Find students with "Ali" in their name who are in the "Mathematics" department **and** have taken at least one course, then update their age to 21.

Solution:

```
db.students.updateMany(
  {
    name: /Ali/i,
    department: "Mathematics",
    $expr: { $gt: [{ $size: "$courses" }, 0] }
  },
  { $set: { age: 21 } }
);
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mycompiler_mongodb>
```

16. Calculate the average GPA **and** the average number of courses taken by students in each department.

Solution:

```
db.students.aggregate([
  {
    $group: {
      _id: "$department",
      averageGPA: { $avg: "$gpa" },
      averageCourses: { $avg: { $size: "$courses" } }
    }
  }
]);
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... [
  { _id: 'Business', averageGPA: 3.8, averageCourses: 2 },
  { _id: 'Engineering', averageGPA: 3.7, averageCourses: 3 },
  { _id: 'Computer Science', averageGPA: 3.5, averageCourses: 2.5 },
  { _id: 'Mathematics', averageGPA: 4, averageCourses: 3 }
]
mycompiler_mongodb>
mycompiler_mongodb>
```

17. Find and delete the course with the lowest grade for the student with the highest number of courses **and** update their GPA by decreasing it by 0.2.

Solution:

```
db.students.insertMany([
 {
  name: "Shoaib Akhter",
  age: 21,
  department: "Computer Science",
  gpa: 3.2,
  courses: [
   { course_name: "Algorithms", grade: "B" },
   { course_name: "Data Structures", grade: "B" }
  ]
 },
 {
  name: "Emily Davis",
  age: 23,
  department: "Business",
  gpa: 3.8,
  courses: [
   { course_name: "Marketing 101", grade: "A" },
   { course_name: "Accounting Basics", grade: "B" }
  ]
 },
 {
  name: "Michael Smith",
  age: 35,
  department: "Engineering",
  gpa: 4.0,
  courses: [
   { course_name: "Thermodynamics", grade: "A" },
   { course_name: "Fluid Mechanics", grade: "A" },
   { course_name: "Calculus II", grade: "A" },
   { course_name: "Material Science", grade: "B" }
  ]
 },
 {
  name: "Aisha Ali",
```

```
    age: 22,
    department: "Mathematics",
    gpa: 4.0,
    courses: [
      { course_name: "Linear Algebra", grade: "A" },
      { course_name: "Real Analysis", grade: "A" },
      { course_name: "Complex Analysis", grade: "A" }
    ]
  },
  {
    name: "David Kim",
    age: 25,
    department: "Computer Science",
    gpa: 3.8,
    courses: [
      { course_name: "Intro to Programming", grade: "A" },
      { course_name: "Math", grade: "C" },
      { course_name: "Computer Architecture", grade: "B" }
    ]
  }
]);
const studentWithMaxCourses = db.students.aggregate([
  {
    $addFields: {
      courseCount: { $size: "$courses" }
    }
  },
  {
    $sort: { courseCount: -1 }
  },
  {
    $limit: 1
  }
]).toArray()[0]; // Get the first document
if (studentWithMaxCourses) {
  const gradeMap = { A: 4, B: 3, C: 2, D: 1, F: 0 };
  let lowestCourse = null;
  let lowestGradeValue = Infinity;
  studentWithMaxCourses.courses.forEach(course => {
    const gradeValue = gradeMap[course.grade] || 0;
    if (gradeValue < lowestGradeValue) {
      lowestGradeValue = gradeValue;
      lowestCourse = course;
    }
  });
  if (lowestCourse) {
    db.students.updateOne(
      { _id: studentWithMaxCourses._id },
      {
        $pull: { courses: { course_name: lowestCourse.course_name, grade:
lowestCourse.grade } },
```

```
      $inc: { gpa: -0.2 } // Step 4: Decrease GPA by 0.2
    }
  );
 }
}
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ...
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mycompiler_mongodb>
```

18. Retrieve the names and GPAs of students aged 20 or above who have a total GPA greater than 10 when summed across all their courses.

Solution:

```
db.students.insertMany([
 {
  name: "Shoaib Akhter",
  age: 21,
  department: "Computer Science",
  gpa: 3.2,
  courses: [
   { course_name: "Algorithms", grade: "B" },
   { course_name: "Data Structures", grade: "B" }
  ]
 },
 {
  name: "Emily Davis",
  age: 23,
  department: "Business",
  gpa: 3.8,
  courses: [
   { course_name: "Marketing 101", grade: "A" },
   { course_name: "Accounting Basics", grade: "B" }
  ]
 },
 {
  name: "Michael Smith",
  age: 35,
  department: "Engineering",
  gpa: 4,
  courses: [
```

```
      { course_name: "Thermodynamics", grade: "A" },
      { course_name: "Fluid Mechanics", grade: "A" },
      { course_name: "Calculus II", grade: "A" }
    ]
  },
  {
    name: "Aisha Ali",
    age: 22,
    department: "Mathematics",
    gpa: 4.0,
    courses: [
      { course_name: "Linear Algebra", grade: "A" },
      { course_name: "Math", grade: "A" },
      { course_name: "Complex Analysis", grade: "A" }
    ]
  },
  {
    name: "David Kim",
    age: 25,
    department: "Computer Science",
    gpa: 3.8,
    courses: [
      { course_name: "Intro to Programming", grade: "A" },
      { course_name: "Math", grade: "C" },
      { course_name: "Computer Architecture", grade: "B" }
    ]
  }
]);


db.students.aggregate([
  {
    $match: {
      age: { $gte: 20 }
    }
  },
  {
    $addFields: {
      totalCourseGPA: {
        $sum: {
          $map: {
            input: "$courses",
            as: "crs",
            in: {
              $switch: {
                branches: [
                  { case: { $eq: ["$$crs.grade", "A"] }, then: 4 },
                  { case: { $eq: ["$$crs.grade", "B"] }, then: 3 },
                  { case: { $eq: ["$$crs.grade", "C"] }, then: 2 },
                  { case: { $eq: ["$$crs.grade", "D"] }, then: 1 }
                ],
```

```
                default: 0
              }
            }
          }
        }
      }
    }
  },
  {
    $match: {
      totalCourseGPA: { $gt: 10 }
    }
  },
  {
    $project: {
      _id: 0,
      name: 1,
      gpa: 1,
      totalCourseGPA: 1
    }
  }
]);
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... .
  { name: 'Michael Smith', gpa: 4, totalCourseGPA: 12 },
  { name: 'Aisha Ali', gpa: 4, totalCourseGPA: 12 }
]
mycompiler_mongodb>
mycompiler_mongodb>
```

19. Add a new student named "Sara Khan" to the "Mathematics" department, aged 22, with a GPA of 3.7 and three courses. Then, retrieve all students sorted by their total number of courses in ascending order.

Solution:

```
db.students.insertOne({
  name: "Sara Khan",
  age: 22,
  department: "Mathematics",
  gpa: 3.7,
  courses: [
    { course_name: "Calculus III", grade: "A" },
    { course_name: "Statistics", grade: "B" },
    { course_name: "Discrete Math", grade: "B" }
  ]
});
```

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      age: 1,
      department: 1,
      gpa: 1,
      totalCourses: { $size: "$courses" }
    }
  },
  { $sort: { totalCourses: 1 } }
]);
```

Output:

```
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... {
  acknowledged: true,
  insertedId: ObjectId('6767c04179679353096b1291')
}
mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... ... ... ... [
  {
    _id: ObjectId('6767c04079679353096b128c'),
    name: 'Shoaib Akhter',
    age: 21,
    department: 'Computer Science',
    gpa: 3.2,
    totalCourses: 2
  },
  {
    _id: ObjectId('6767c04079679353096b128d'),
    name: 'Emily Davis',
    age: 23,
    department: 'Business',
    gpa: 3.8,
    totalCourses: 2
  },
  {
    _id: ObjectId('6767c04079679353096b128e'),
    name: 'Michael Smith',
    age: 35,
    department: 'Engineering',
    gpa: 3.7,
    totalCourses: 3
  },
```

```
  {
    _id: ObjectId('6767c04079679353096b128f'),
    name: 'Aisha Ali',
    age: 22,
    department: 'Mathematics',
    gpa: 4,
    totalCourses: 3
  },
  {
    _id: ObjectId('6767c04079679353096b1290'),
    name: 'David Kim',
    age: 25,
    department: 'Computer Science',
    gpa: 3.8,
    totalCourses: 3
  },
    {
      _id: ObjectId('6767c04179679353096b1291'),
      name: 'Sara Khan',
      age: 22,
      department: 'Mathematics',
      gpa: 3.7,
      totalCourses: 3
    }
  ]
mycompiler_mongodb>
mycompiler_mongodb>
```

20. Retrieve students, grouping them by department, and calculate the **maximum GPA** and **total number of students** within each department.

Solution:

```
db.students.aggregate([
  {
    $group: {
      _id: "$department",
      maxGPA: { $max: "$gpa" },
      totalStudents: { $sum: 1 }
    }
  }
]);
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> ... ... ... ... ... ... ... ... [
  { _id: 'Business', maxGPA: 3.8, totalStudents: 1 },
  { _id: 'Mathematics', maxGPA: 4, totalStudents: 1 },
  { _id: 'Engineering', maxGPA: 3.7, totalStudents: 1 },
  { _id: 'Computer Science', maxGPA: 3.8, totalStudents: 2 }
]
mycompiler_mongodb>
mycompiler_mongodb>
```

Submission Deadline: 23rd December 2024