CS 5200 Practicum 2 - Milestone 2 Documentation

Task 5: Advanced MongoDB Queries & Aggregation

I've implemented advanced aggregation queries on our course data model to extract valuable insights from the e-learning platform.

Query 1: Department Course Statistics

Purpose:
Aggregates courses by department to determine the number of courses and the average credits offered per department.

Business Value:
Provides insights for resource allocation and curriculum evaluation by highlighting departmental strengths and disparities in course offerings.

Implementation:

```
db.course.aggregate([
  { $match: { department: { $exists: true, $ne: null } } },
  {
    $group: {
      _id: "$department",
      courseCount: { $sum: 1 },
      avgCredits: { $avg: "$credits" }
    }
  },
  { $sort: { courseCount: -1 } }
]);
```

Sample Results:

```
{ "_id": "D134", "courseCount": 3, "avgCredits": 3 }
{ "_id": "History", "courseCount": 2, "avgCredits": 4 }
```

Query 2: Course Lesson Release Statistics

Purpose: Counts the lessons under each course based on their release status, i.e., released or unreleased.

Business Value: Helps instructors monitor course content updates and manage release scheduling effectively.

Implementation:

```
db.course.aggregate([
  { $unwind: "$lessons" },
  {
    $group: {
      _id: { courseId: "$_id", released: "$lessons.releasedOrNot" },
      lessonCount: { $sum: 1 }
    }
  },
  {
    $group: {
      _id: "$_id.courseId",
      lessonStats: { $push: { released: "$_id.released", count: "$lessonCount" } }
    }
  }
]);
```

Sample Results:

```
{
  "_id": "603dcd1f8a1234567890abcd",
  "lessonStats": [
    { "released": true, "count": 3 },
    { "released": false, "count": 2 }
  ]
}
```

Query 3: Course Enrollment Statistics

Purpose: Provides enrollment statistics for each course with status breakdown.

Business Value: Helps administrators track course popularity and enrollment trends.

Implementation:

```
async getCourseEnrollmentStatistics() {
  try {
    const result = await Enrollment.aggregate([
      {
        $group: {
          _id: "$course",
          totalEnrollments: { $sum: 1 },
          activeEnrollments: {
            $sum: { $cond: [{ $eq: ["$status", "ACTIVE"] }, 1, 0] }
          },
          completedEnrollments: {
            $sum: { $cond: [{ $eq: ["$status", "COMPLETED"] }, 1, 0] }
          },
          droppedEnrollments: {
            $sum: { $cond: [{ $eq: ["$status", "DROPPED"] }, 1, 0] }
          },
          pendingEnrollments: {
            $sum: { $cond: [{ $eq: ["$status", "PENDING"] }, 1, 0] }
          }
        }
      },
      {
        $lookup: {
          from: "course",
          localField: "_id",
          foreignField: "_id",
          as: "courseDetails"
        }
      },
      { $unwind: "$courseDetails" },
      {
        $project: {
          _id: 0,
          courseId: "$_id",
          courseName: "$courseDetails.name",
          courseNumber: "$courseDetails.number",
          department: "$courseDetails.department",
          term: "$courseDetails.term",
          totalEnrollments: 1,
          activeEnrollments: 1,
          completedEnrollments: 1,
          droppedEnrollments: 1,
          pendingEnrollments: 1,
          activePercentage: {
            $multiply: [
              { $divide: ["$activeEnrollments", "$totalEnrollments"] },
              100
```

```
            ]
          }
        }
      },
      { $sort: { totalEnrollments: -1 } }
    ]);
    return result;
  } catch (error) {
    console.error('Error in getCourseEnrollmentStatistics:', error);
    throw error;
  }
}
```

Sample Results:


```
[
  {
    "courseId": "67f6ee5b30bcd5365624d31f",
    "courseName": "Spacecraft Design",
    "courseNumber": "RS103",
    "department": "D123",
    "term": "2025 SP",
    "totalEnrollments": 2,
    "activeEnrollments": 2,
    "completedEnrollments": 0,
    "droppedEnrollments": 0,
    "pendingEnrollments": 0,
    "activePercentage": 100
  },
  {
    "courseId": "67f6ee5b30bcd5365624d324",
    "courseName": "Foundations of Christian Ethics",
    "courseNumber": "THEO101",
    "department": "Theology",
    "term": "2025 SP",
    "totalEnrollments": 2,
    "activeEnrollments": 2,
    "completedEnrollments": 0,
    "droppedEnrollments": 0,
    "pendingEnrollments": 0,
    "activePercentage": 100
  }
]
```


The results show enrollment counts by status for each course, sorted by popularity. This helps identify high-demand courses and monitor student retention.

Query 4: Instructor Course Load

Purpose: Calculates teaching load metrics for each instructor.

Business Value: Ensures fair workload distribution among faculty.

Implementation:

```
async getInstructorCourseLoad() {
  try {
    const result = await Course.aggregate([
      { $match: { instructor: { $ne: null } } },
      {
        $group: {
          _id: "$instructor",
          coursesCount: { $sum: 1 },
          totalCredits: { $sum: "$credits" },
          courses: {
            $push: {
              courseId: "$_id",
              courseName: "$name",
              courseNumber: "$number",
              credits: "$credits",
              term: "$term"
            }
          }
        }
      },
      {
        $lookup: {
          from: "user",
          localField: "_id",
          foreignField: "_id",
          as: "instructorDetails"
        }
      },
      { $unwind: "$instructorDetails" },
      {
        $lookup: {
          from: "enrollment",
          let: { instructor_id: "$_id" },
          pipeline: [
            {
              $lookup: {
                from: "course",
                localField: "course",
                foreignField: "_id",
                as: "courseInfo"
              }
            },
            { $unwind: "$courseInfo" },
            {
              $match: {
                $expr: { $eq: ["$courseInfo.instructor", "$$instructor_id"] },
                status: "ACTIVE"
```

```
              }
            }
          ],
          as: "enrollments"
        }
      },
      {
        $project: {
          _id: 0,
          instructorId: "$_id",
          instructorName: {
            $concat: ["$instructorDetails.firstName", " ",
"$instructorDetails.lastName"]
          },
          email: "$instructorDetails.email",
          coursesCount: 1,
          totalCredits: 1,
          totalStudents: { $size: "$enrollments" },
          courses: 1
        }
      },
      { $sort: { coursesCount: -1 } }
    ]);
    return result;
  } catch (error) {
    console.error('Error in getInstructorCourseLoad:', error);
    throw error;
  }
}
```

Sample Results:

```
[
  {
    "instructorId": "67ef53e352126ab8ed55d754",
    "instructorName": "Robert Williams",
    "email": "r.williams@university.edu",
    "coursesCount": 2,
    "totalCredits": 5,
    "totalStudents": 2,
    "courses": [
      {
        "courseId": "67f6ee5b30bcd5365624d31f",
        "courseName": "Spacecraft Design",
        "courseNumber": "RS103",
        "credits": 4,
        "term": "2025 SP"
      },
      {
        "courseId": "67f71a7ab148fe0e7e432156",
        "courseName": "computer science",
        "courseNumber": "cs101",
        "credits": 1,
        "term": "2025 SP"
      }
    ]
  },
  {
    "instructorId": "67ef53e352126ab8ed55d75b",
    "instructorName": "Richard Martin",
    "email": "r.martin@university.edu",
    "coursesCount": 1,
    "totalCredits": 3,
    "totalStudents": 0,
    "courses": [
      {
        "courseId": "67f6ee5b30bcd5365624d321",
        "courseName": "Physical Chemistry",
        "courseNumber": "CH1250",
        "credits": 3,
        "term": "2025 SP"
      }
    ]
  }
]
```


The results show each instructor's teaching load, including course count, credits, and student counts, helping administrators balance faculty workload.

Query 5: Student Performance Analytics

Purpose: Analyzes student enrollment patterns and engagement levels.

Business Value: Identifies at-risk students who might need additional support.

Implementation:

```
async getStudentPerformanceAnalytics() {
  try {
    const result = await Enrollment.aggregate([
      { $match: { status: "ACTIVE" } },
      {
        $lookup: {
          from: "user",
          localField: "user",
          foreignField: "_id",
          as: "studentDetails"
        }
      },
      { $unwind: "$studentDetails" },
      {
        $lookup: {
          from: "course",
          localField: "course",
          foreignField: "_id",
          as: "courseDetails"
        }
      },
      { $unwind: "$courseDetails" },
      {
        $group: {
          _id: "$user",
          studentName: {
            $first: {
              $concat: ["$studentDetails.firstName", " ",
"$studentDetails.lastName"]
            }
          },
          email: { $first: "$studentDetails.email" },
          coursesEnrolled: { $sum: 1 },
          totalCredits: { $sum: "$courseDetails.credits" },
          courses: { $push: {
              courseId: "$courseDetails._id",
              courseName: "$courseDetails.name",
              courseNumber: "$courseDetails.number",
              term: "$courseDetails.term",
              enrollmentStatus: "$status",
              enrollmentDate: "$enrollmentDate"
          }},
          lastActivity: { $max: "$lastActivity" }
        }
      },
      {
        $addFields: {
```

```
          daysActive: {
            $divide: [
              { $subtract: [new Date(), "$lastActivity"] },
              1000 * 60 * 60 * 24
            ]
          }
        }
      },
      {
        $project: {
          _id: 0,
          studentId: "$_id",
          studentName: 1,
          email: 1,
          coursesEnrolled: 1,
          totalCredits: 1,
          courses: 1,
          lastActivity: 1,
          daysActive: 1,
          engagementScore: {
            $cond: {
              if: { $lt: ["$daysActive", 7] },
              then: "High",
              else: {
                $cond: {
                  if: { $lt: ["$daysActive", 14] },
                  then: "Medium",
                  else: "Low"
                }
              }
            }
          }
        }
      },
      { $sort: { coursesEnrolled: -1 } }
    ]);
    return result;
  } catch (error) {
    console.error('Error in getStudentPerformanceAnalytics:', error);
    throw error;
  }
}
```

Sample Results:

```
[
  {
    "studentId": "67ef53e352126ab8ed55d753",
    "studentName": "Em Smith",
    "email": "emma.smith@university.edu",
    "coursesEnrolled": 2,
    "totalCredits": 7,
    "courses": [
      {
        "courseId": "67f6ee5b30bcd5365624d31f",
        "courseName": "Spacecraft Design",
        "courseNumber": "RS103",
        "term": "2025 SP",
        "enrollmentStatus": "ACTIVE",
        "enrollmentDate": "2025-04-09T00:00:00.000Z"
      },
      {
        "courseId": "67f6ee5b30bcd5365624d320",
        "courseName": "Inorganic Chemistry",
        "courseNumber": "CH1240",
        "term": "2025 SP",
        "enrollmentStatus": "ACTIVE",
        "enrollmentDate": "2025-04-10T00:00:00.000Z"
      }
    ],
    "lastActivity": "2023-11-06T10:15:45.000Z",
    "daysActive": 516.9687,
    "engagementScore": "Low"
  }
]
```


The results show student enrollment details with an engagement score based on recent activity, helping instructors identify students who may need intervention.

Query 6: Course Content Analysis

Purpose: Analyzes course structure, content distribution, and workload.

Business Value: Helps instructors balance course content and identify overloaded courses.

Implementation:

```
async getCourseContentAnalysis() {
  try {
    const result = await Course.aggregate([
      {
        $project: {
          _id: 1,
          name: 1,
          number: 1,
          term: 1,
          department: 1,
          credits: 1,
          lessonCount: { $size: { $ifNull: ["$lessons", []] } }
        }
      },
      {
        $lookup: {
          from: "assignment",
          let: { course_number: "$number" },
          pipeline: [
            {
              $match: {
                $expr: {
                  $eq: ["$course_number", "$$course_number"]
                }
              }
            }
          ],
          as: "assignments"
        }
      },
      {
        $lookup: {
          from: "quiz",
          let: { course_number: "$number" },
          pipeline: [
            {
              $match: {
                $expr: {
                  $eq: ["$courses", "$$course_number"]
                }
              }
            }
          ],
          as: "quizzes"
        }
      },
      {
```

```
  $addFields: {
    assignmentCount: { $size: "$assignments" },
    quizCount: { $size: "$quizzes" },
    totalAssignmentPoints: { $sum: "$assignments.points" },
    assignmentTypes: {
      released: {
        $size: {
          $filter: {
            input: "$assignments",
            as: "assignment",
            cond: { $eq: ["$$assignment.releasedOrNot", true] }
          }
        }
      },
      upcoming: {
        $size: {
          $filter: {
            input: "$assignments",
            as: "assignment",
            cond: { $eq: ["$$assignment.releasedOrNot", false] }
          }
        }
      }
    },
    quizTypes: {
      exams: {
        $size: {
          $filter: {
            input: "$quizzes",
            as: "quiz",
            cond: { $eq: ["$$quiz.assignmentGroup", "EXAMS"] }
          }
        }
      },
      quizzes: {
        $size: {
          $filter: {
            input: "$quizzes",
            as: "quiz",
            cond: { $eq: ["$$quiz.assignmentGroup", "QUIZZES"] }
          }
        }
      }
    }
  }
},
{
```

```
        $project: {
          _id: 0,
          courseId: "$_id",
          name: 1,
          number: 1,
          term: 1,
          department: 1,
          credits: 1,
          contentSummary: {
            lessonCount: "$lessonCount",
            assignmentCount: "$assignmentCount",
            quizCount: "$quizCount"
          },
          assignmentAnalysis: {
            totalPoints: "$totalAssignmentPoints",
            releasedCount: "$assignmentTypes.released",
            upcomingCount: "$assignmentTypes.upcoming"
          },
          quizAnalysis: {
            examCount: "$quizTypes.exams",
            quizCount: "$quizTypes.quizzes"
          },
          contentDensity: {
            $divide: [
              { $add: ["$lessonCount", "$assignmentCount", "$quizCount"] },
              "$credits"
            ]
          }
        }
      },
      { $sort: { "contentDensity": -1 } }
    ]);
    return result;
  } catch (error) {
    console.error('Error in getCourseContentAnalysis:', error);
    throw error;
  }
}
```

Sample Results:

```
[
  {
    "courseId": "67f6ee5b30bcd5365624d332",
    "name": "Quantum Computing Fundamentals",
    "number": "CS4500",
    "term": "2025 SP",
    "department": "Computer Science",
    "credits": 4,
    "contentSummary": {
      "lessonCount": 2,
      "assignmentCount": 7,
      "quizCount": 4
    },
    "assignmentAnalysis": {
      "totalPoints": 680,
      "releasedCount": 5,
      "upcomingCount": 2
    },
    "quizAnalysis": {
      "examCount": 2,
      "quizCount": 2
    },
    "contentDensity": 3.25
  }
]
```


This query analyzes course content structure, showing lesson counts, assignment distribution, and quiz types. The contentDensity metric helps identify courses that may be overloaded relative to their credit value.

Query 7: Department Analytics

Purpose: Provides department-level metrics on courses, students, and faculty.

Business Value: Helps administrators assess resource allocation across departments.

Implementation:

```
async getDepartmentAnalytics() {
  try {
    const result = await Course.aggregate([
      {
        $group: {
          _id: "$department",
          courseCount: { $sum: 1 },
          totalCredits: { $sum: "$credits" },
          courses: {
            $push: {
              courseId: "$_id",
              name: "$name",
              number: "$number",
              credits: "$credits"
            }
          },
          instructors: { $addToSet: "$instructor" }
        }
      },
      {
        $lookup: {
          from: "enrollment",
          let: { dept_courses: "$courses.courseId" },
          pipeline: [
            {
              $match: {
                $expr: {
                  $in: ["$course", "$$dept_courses"]
                },
                status: "ACTIVE"
              }
            }
          ],
          as: "enrollments"
        }
      },
      {
        $addFields: {
          studentCount: { $size: "$enrollments" },
          instructorCount: { $size: "$instructors" },
          avgCoursesPerInstructor: {
            $cond: [
              { $eq: [{ $size: "$instructors" }, 0] },
              0,
              { $divide: ["$courseCount", { $size: "$instructors" }] }
            ]
          },
```

```
            avgStudentsPerCourse: {
              $cond: [
                { $eq: ["$courseCount", 0] },
                0,
                { $divide: [{ $size: "$enrollments" }, "$courseCount"] }
              ]
            }
          }
        },
        {
          $project: {
            _id: 0,
            department: "$_id",
            courseCount: 1,
            totalCredits: 1,
            studentCount: 1,
            instructorCount: 1,
            avgCoursesPerInstructor: 1,
            avgStudentsPerCourse: 1,
            studentToFacultyRatio: {
              $cond: [
                { $eq: ["$instructorCount", 0] },
                "N/A",
                { $divide: ["$studentCount", "$instructorCount"] }
              ]
            }
          }
        },
        { $sort: { studentCount: -1 } }
      ]);
      return result;
    } catch (error) {
      console.error('Error in getDepartmentAnalytics:', error);
      throw error;
    }
  }
```

Sample Results:

```
[
  {
    "department": "D123",
    "courseCount": 1,
    "totalCredits": 4,
    "studentCount": 2,
    "instructorCount": 1,
    "avgCoursesPerInstructor": 1,
    "avgStudentsPerCourse": 2,
    "studentToFacultyRatio": 2
  },
  {
    "department": "D134",
    "courseCount": 3,
    "totalCredits": 9,
    "studentCount": 1,
    "instructorCount": 2,
    "avgCoursesPerInstructor": 1.5,
    "avgStudentsPerCourse": 0.3333333333333333,
    "studentToFacultyRatio": 0.5
  }
]
```


The results show department-level metrics including student-to-faculty ratios, helping administrators identify departments that may need resource adjustments.

Task 6: Query Optimization & Indexing Strategy

Indexing Implementation

I created the following indexes to optimize query performance:

```
// Index creation code from indexOptimization.js
async createAllIndexes() {
  try {
    console.log('Creating database indexes...');

    // Index 1: Optimize user lookup by role
    await User.collection.createIndex({ role: 1 });
    console.log('Created index on User.role');

    // Index 2: Optimize course queries by department and instructor
    await Course.collection.createIndex({ department: 1, instructor: 1 });
    console.log('Created composite index on Course.department and
Course.instructor');

    // Index 3: Optimize enrollment queries by user and status
    await Enrollment.collection.createIndex({ user: 1, status: 1 });
    console.log('Created composite index on Enrollment.user and
Enrollment.status');

    // Index 4: Optimize enrollment queries by course
    await Enrollment.collection.createIndex({ course: 1, status: 1 });
    console.log('Created composite index on Enrollment.course and
Enrollment.status');

    // Index 5: Optimize assignment queries by course number and due date
    await Assignment.collection.createIndex({ course_number: 1, dueDate: 1 });
    console.log('Created composite index on Assignment.course_number and
Assignment.dueDate');

    // Index 6: Text index for course search functionality
    await Course.collection.createIndex(
      { name: "text", description: "text" },
      { weights: { name: 10, description: 5 } }
    );
    console.log('Created text index on Course.name and Course.description');

    console.log('All indexes created successfully');
  } catch (error) {
    console.error('Error creating indexes:', error);
    throw error;
  }
}
```

Performance Benchmarks

I tested three key queries with and without indexes:

Query 1: Finding Users by Role

Before Indexing:

- Execution time: 32.56 ms
- Documents examined: 20
- Execution plan: COLLSCAN (full collection scan)

After Indexing:

- Execution time: 4.73 ms
- Documents examined: 10
- Execution plan: IXSCAN (index scan)

Improvement: 85.5% faster execution time

Explain Analysis:


```
{
  "queryPlanner": {
    "plannerVersion": 1,
    "namespace": "elearning.user",
    "indexFilterSet": false,
    "parsedQuery": { "role": { "$eq": "STUDENT" } },
    "winningPlan": {
      "stage": "FETCH",
      "inputStage": {
        "stage": "IXSCAN",
        "keyPattern": { "role": 1 },
        "indexName": "role_1",
        "direction": "forward",
        "indexBounds": { "role": ["[\"STUDENT\", \"STUDENT\"]"] }
      }
    },
    "rejectedPlans": []
  },
  "executionStats": {
    "executionSuccess": true,
    "nReturned": 10,
    "executionTimeMillis": 4.73,
    "totalKeysExamined": 10,
    "totalDocsExamined": 10
  }
}
```


Query 2: Finding Active Enrollments for a Course

Before Indexing:

- Execution time: 28.31 ms
- Documents examined: 14
- Execution plan: COLLSCAN (full collection scan)

After Indexing:

- Execution time: 3.57 ms
- Documents examined: 2
- Execution plan: IXSCAN (index scan)

Improvement: 87.4% faster execution time

Explain Analysis:


```
{
  "queryPlanner": {
    "namespace": "elearning.enrollment",
    "parsedQuery": {
      "$and": [
        { "course": { "$eq": ObjectId("67f6ee5b30bcd5365624d31f") } },
        { "status": { "$eq": "ACTIVE" } }
      ]
    },
    "winningPlan": {
      "stage": "FETCH",
      "inputStage": {
        "stage": "IXSCAN",
        "keyPattern": { "course": 1, "status": 1 },
        "indexName": "course_1_status_1",
        "direction": "forward"
      }
    }
  },
  "executionStats": {
    "executionSuccess": true,
    "nReturned": 2,
    "executionTimeMillis": 3.57,
    "totalKeysExamined": 2,
    "totalDocsExamined": 2
  }
}
```


Query 3: Finding Upcoming Assignments for a Course

Before Indexing:

- Execution time: 24.18 ms
- Documents examined: 20

- Execution plan: COLLSCAN (full collection scan)

After Indexing:

- Execution time: 5.32 ms
- Documents examined: 3
- Execution plan: IXSCAN (index scan)

Improvement: 78.0% faster execution time

Explain Analysis:


```
{
  "queryPlanner": {
    "namespace": "elearning.assignment",
    "parsedQuery": {
      "$and": [
        { "course_number": { "$eq": "CS4500" } },
        { "dueDate": { "$gt": ISODate("2025-04-10T00:00:00Z") } },
        { "releasedOrNot": { "$eq": true } }
      ]
    },
    "winningPlan": {
      "stage": "FETCH",
      "filter": { "releasedOrNot": { "$eq": true } },
      "inputStage": {
        "stage": "IXSCAN",
        "keyPattern": { "course_number": 1, "dueDate": 1 },
        "indexName": "course_number_1_dueDate_1"
      }
    }
  },
  "executionStats": {
    "executionSuccess": true,
    "nReturned": 3,
    "executionTimeMillis": 5.32,
    "totalKeysExamined": 3,
    "totalDocsExamined": 3
  }
}
```


Performance Bottleneck Analysis

Based on the benchmarking results, I identified and addressed these bottlenecks:

1. **Role-based filtering:** Created a simple index on the `role` field, reducing query time by 85.5%.

2. **Course enrollment lookups:** Added a compound index on `{ course: 1, status: 1 }`, improving performance by 87.4%.
3. **Assignment filtering:** Created a compound index on `{ course_number: 1, dueDate: 1 }`, reducing execution time by 78.0%.
4. **Course search:** Added a text index on course names and descriptions with appropriate weights.

The indexing strategy combines simple indexes for equality filters, compound indexes for multi-field queries, and text indexes for search functionality. These optimizations resulted in an average performance improvement of 83.6% across tested queries.