DBMSL ASSIGNMENT – 10

Roll No.: 31446

Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB

Setup: Sample Database and Collection

```
use college_db

db.students.insertMany([
... {
... _id: 1,
... name: "Rahul Sharma",
... age: 20,
... department: "Computer Science",
... city: "Mumbai",
... cgpa: 8.5,
... semester: 6,
... subjects: ["DBMS", "AI", "Networking"],
... fees_paid: 45000
... },
... {
... _id: 2,
... name: "Priya Patel",
... age: 21,
... department: "Information Technology",
... city: "Pune",
... cgpa: 9.2,
... semester: 8,
... subjects: ["Cloud Computing","IoT","Big Data"],
... fees_paid: 50000
... },
... {
... _id: 3,
... name: "Amit Kumar",
... age: 19,
... department: "Computer Science",
... city: "Delhi",
... cgpa: 7.8,
... semester: 4,
... subjects: ["DSA", "DBMS", "OS"],
... fees_paid: 40000
... },
... {
... _id: 4,
... name: "Sneha Reddy",
... age: 22,
... department: "Electronics",
... city: "Banglore",
... cgpa: 8.9,
... semester: 8,
... subjects: ["VLSI", "Embedded Systems"],
... fees_paid: 48000
... },
... {
... _id: 5,
... name: "Vikram Singh",
... age: 20,
... department: "Information Technology",
... city: "Pune",
... cgpa: 8.1,
... semester: 6,
... subjects: ["Web Development", "DBMS", "AI"],
```

```
... fees_paid: 46000
... },
... {
... _id: 6,
... name: "Anjali Desai",
... age: 21,
... department: "Computer Science",
... city: "Mumbai",
... cgpa: 9.5,
... semester: 8,
... subjects: ["Machine Learning", "AI", "DBMS"],
... fees_paid: 52000
... }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5, '5': 6 }


college_db> db.students.find()
[
  {
    _id: 1,
    name: 'Rahul Sharma',
    age: 20,
    department: 'Computer Science',
    city: 'Mumbai',
    cgpa: 8.5,
    semester: 6,
    subjects: [ 'DBMS', 'AI', 'Networking' ],
    fees_paid: 45000
  },
  {
    _id: 2,
    name: 'Priya Patel',
    age: 21,
    department: 'Information Technology',
    city: 'Pune',
    cgpa: 9.2,
    semester: 8,
    subjects: [ 'Cloud Computing', 'IoT', 'Big Data' ],
    fees_paid: 50000
  },
  {
    _id: 3,
    name: 'Amit Kumar',
    age: 19,
    department: 'Computer Science',
    city: 'Delhi',
    cgpa: 7.8,
    semester: 4,
    subjects: [ 'DSA', 'DBMS', 'OS' ],
    fees_paid: 40000
  },
  {
    _id: 4,
    name: 'Sneha Reddy',
    age: 22,
```

```
    department: 'Electronics',
    city: 'Banglore',
    cgpa: 8.9,
    semester: 8,
    subjects: [ 'VLSI', 'Embedded Systems' ],
    fees_paid: 48000
  },
  {
    _id: 5,
    name: 'Vikram Singh',
    age: 20,
    department: 'Information Technology',
    city: 'Pune',
    cgpa: 8.1,
    semester: 6,
    subjects: [ 'Web Development', 'DBMS', 'AI' ],
    fees_paid: 46000
  },
  {
    _id: 6,
    name: 'Anjali Desai',
    age: 21,
    department: 'Computer Science',
    city: 'Mumbai',
    cgpa: 9.5,
    semester: 8,
    subjects: [ 'Machine Learning', 'AI', 'DBMS' ],
    fees_paid: 52000
  }
]
```

Part 1: MongoDB Aggregation

1. Basic Aggregation - $match and $project

Find students from Computer Science department and show only name and CGPA:

```
college_db> db.students.aggregate([
... {
... $match: { department: "Computer Science" }
... },
... {
... $project:{
... _id: 0,
... name: 1,
... cgpa: 1,
... department: 1
... }
... }
... ])
[
  { name: 'Rahul Sharma', department: 'Computer Science', cgpa: 8.5 },
  { name: 'Amit Kumar', department: 'Computer Science', cgpa: 7.8 },
  { name: 'Anjali Desai', department: 'Computer Science', cgpa: 9.5 }
]
```

2. Grouping and Aggregation - $group

Count students and average, min, max CGPA by department:

```
college_db> db.students.aggregate([
... {
... $group: {
... _id: "$department",
... total_students: { $sum: 1 },
... average_cgpa: { $avg: "$cgpa" },
... max_cgpa: { $max: "$cgpa" },
... min_cgpa: { $min: "$cgpa" }
... }
... },
... {
... $sort: { average_cgpa: -1 }
... }
... ])
[
  {
    _id: 'Electronics',
    total_students: 1,
    average_cgpa: 8.9,
    max_cgpa: 8.9,
    min_cgpa: 8.9
  },
  {
    _id: 'Information Technology',
    total_students: 2,
    average_cgpa: 8.649999999999999,
    max_cgpa: 9.2,
    min_cgpa: 8.1
  },
  {
    _id: 'Computer Science',
    total_students: 3,
    average_cgpa: 8.6,
    max_cgpa: 9.5,
    min_cgpa: 7.8
  }
]
```

3. Sorting and Limiting - $sort and $limit

Find top 3 students with highest CGPA:

```
college_db> db.students.aggregate([
... {
... $sort: { cgpa: -1 }
... },
... {
... $limit: 3
... },
... {
... $project: {
```

```
... _id: 0,
... name: 1,
... cgpa: 1,
... department: 1
... }
... }
... ])
[
  { name: 'Anjali Desai', department: 'Computer Science', cgpa: 9.5 },
  {
    name: 'Priya Patel',
    department: 'Information Technology',
    cgpa: 9.2
  },
  { name: 'Sneha Reddy', department: 'Electronics', cgpa: 8.9 }
```

4. Array Operations - $unwind

List all subjects being studied with student names:

```
college_db> db.students.aggregate([
... {
... $unwind: "$subjects"
... },
... {
... $project: {
... _id: 0,
... student_name: "$name",
... subject: "$subjects"
... }
... },
... {
... $sort: { subject: 1 }
... }
... ])
[
  { student_name: 'Rahul Sharma', subject: 'AI' },
  { student_name: 'Vikram Singh', subject: 'AI' },
  { student_name: 'Anjali Desai', subject: 'AI' },
  { student_name: 'Priya Patel', subject: 'Big Data' },
  { student_name: 'Priya Patel', subject: 'Cloud Computing' },
  { student_name: 'Rahul Sharma', subject: 'DBMS' },
  { student_name: 'Amit Kumar', subject: 'DBMS' },
  { student_name: 'Vikram Singh', subject: 'DBMS' },
  { student_name: 'Anjali Desai', subject: 'DBMS' },
  { student_name: 'Amit Kumar', subject: 'DSA' },
  { student_name: 'Sneha Reddy', subject: 'Embedded Systems' },
  { student_name: 'Priya Patel', subject: 'IoT' },
  { student_name: 'Anjali Desai', subject: 'Machine Learning' },
  { student_name: 'Rahul Sharma', subject: 'Networking' },
  { student_name: 'Amit Kumar', subject: 'OS' },
  { student_name: 'Sneha Reddy', subject: 'VLSI' },
  { student_name: 'Vikram Singh', subject: 'Web Development' }
]
```

Count how many students study each subject:

```
college_db> db.students.aggregate([
... {
... $unwind: "$subjects"
... },
... {
... $group: {
... _id: "$subjects",
... student_count: { $sum: 1 },
... students: { $push: "$name" }
... }
... },
... {
... $sort: { student_count: -1 }
... }
... ])
[
  {
    _id: 'DBMS',
    student_count: 4,
    students: [ 'Rahul Sharma', 'Amit Kumar', 'Vikram Singh', 'Anjali
Desai' ]
  },
  {
    _id: 'AI',
    student_count: 3,
    students: [ 'Rahul Sharma', 'Vikram Singh', 'Anjali Desai' ]
  },
  { _id: 'OS', student_count: 1, students: [ 'Amit Kumar' ] },
  {
    _id: 'Cloud Computing',
    student_count: 1,
    students: [ 'Priya Patel' ]
  },
  { _id: 'Networking', student_count: 1, students: [ 'Rahul Sharma' ] },
  { _id: 'Big Data', student_count: 1, students: [ 'Priya Patel' ] },
  { _id: 'DSA', student_count: 1, students: [ 'Amit Kumar' ] },
  { _id: 'IoT', student_count: 1, students: [ 'Priya Patel' ] },
  {
    _id: 'Machine Learning',
    student_count: 1,
    students: [ 'Anjali Desai' ]
  },
  {
    _id: 'Web Development',
    student_count: 1,
    students: [ 'Vikram Singh' ]
  },
  {
    _id: 'Embedded Systems',
    student_count: 1,
    students: [ 'Sneha Reddy' ]
  },
  { _id: 'VLSI', student_count: 1, students: [ 'Sneha Reddy' ] }
```

5. Filter Arrays - $filter

Get only the subjects containing "DBMS" for each student.

```
college_db> db.students.aggregate([
... {
... $project: {
... name: 1,
... department: 1,
... subject_with_DBMS: {
... $filter: {
... input: "$subjects",
... as: "subject",
... cond: { $eq: ["$$subject", "DBMS"]}
... }
... }
... }
... }
... ])
[
  {
    _id: 1,
    name: 'Rahul Sharma',
    department: 'Computer Science',
    subject_with_DBMS: [ 'DBMS' ]
  },
  {
    _id: 2,
    name: 'Priya Patel',
    department: 'Information Technology',
    subject_with_DBMS: []
  },
  {
    _id: 3,
    name: 'Amit Kumar',
    department: 'Computer Science',
    subject_with_DBMS: [ 'DBMS' ]
  },
  {
    _id: 4,
    name: 'Sneha Reddy',
    department: 'Electronics',
    subject_with_DBMS: []
  },
  {
    _id: 5,
    name: 'Vikram Singh',
    department: 'Information Technology',
    subject_with_DBMS: [ 'DBMS' ]
  },
  {
    _id: 6,
    name: 'Anjali Desai',
    department: 'Computer Science',
    subject_with_DBMS: [ 'DBMS' ]
  }
```

6. Complex Aggregation - Multiple Stages

Find departments with average CGPA > 8.5 and total fees collected:

```
college_db> db.students.aggregate([
... {
... $group: {
... _id: "$department",
... avg_cgpa: { $avg: "$cgpa" },
... total_fees: { $sum: "$fees_paid" },
... student_count: { $sum: 1 }
... }
... },
... {
... $match: {
... avg_cgpa: { $gt: 8.5 }
... }
... },
... {
... $project: {
... department: "_id",
... _id: 0,
... avg_cgpa: { $round: ["$avg_cgpa", 2] },
... total_fees: 1,
... student_count: 1
... }
... },
... {
... $sort: { avg_cgpa: -1 }
... }
... ])
[
  {
    total_fees: 48000,
    student_count: 1,
    department: '_id',
    avg_cgpa: 8.9
  },
  {
    total_fees: 96000,
    student_count: 2,
    department: '_id',
    avg_cgpa: 8.65
  },
  {
    total_fees: 137000,
    student_count: 3,
    department: '_id',
    avg_cgpa: 8.6
  }
]
```

7. Lookup (Join) - $lookup

First, create a courses collection:

```
college_db> db.courses.insertMany([
... { course_code: "CS101", course_name: "DBMS", credits: 4 },
... { course_code: "CS102", course_name: "AI", credits: 4 },
... { course_code: "CS103", course_name: "Networking", credits: 3 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68fe50281758a0cc8acebea4'),
    '1': ObjectId('68fe50281758a0cc8acebea5'),
    '2': ObjectId('68fe50281758a0cc8acebea6')
  }
}
```

Join students with courses (simplified example):

```
college_db> db.students.aggregate([
... {
... $lookup: {
... from: "courses",
... localField: "subjects",
... foreignField: "course_name",
... as: "course_details"
... }
... },
... {
... $project: {
... name: 1,
... subjects: 1,
... course_details: 1
... }
... },
... {
... $limit: 2
... }
... ])
[
  {
    _id: 1,
    name: 'Rahul Sharma',
    subjects: [ 'DBMS', 'AI', 'Networking' ],
    course_details: [
      {
        _id: ObjectId('68fe50281758a0cc8acebea4'),
        course_code: 'CS101',
        course_name: 'DBMS',
        credits: 4
      },
      {
        _id: ObjectId('68fe50281758a0cc8acebea5'),
        course_code: 'CS102',
        course_name: 'AI',
        credits: 4
      },
```

```
      {
        _id: ObjectId('68fe50281758a0cc8acebea6'),
        course_code: 'CS103',
        course_name: 'Networking',
        credits: 3
      }
    ]
  },
  {
    _id: 2,
    name: 'Priya Patel',
    subjects: [ 'Cloud Computing', 'IoT', 'Big Data' ],
    course_details: []
  }
]
```

Part 2: MongoDB Indexing

1. View Existing Indexes

```
college_db> db.students.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

Output: Shows default _id index

2. Create Single Field Index

Create index on department field:

```
college_db> db.students.createIndex({ department: 1 })
department_1
```

Explanation: 1 for ascending, -1 for descending

3. Create Compound Index

Create compound index on department and cgpa:

```
college_db> db.students.createIndex({ department: 1, cgpa: -1 })
department_1_cgpa_-1
```

4. Create Unique Index

Create unique index on email (add email field first):

```
// Add email field to documents
db.students.updateMany({}, [
  {
    $set: {
      email: {
        $concat: [
          { $toLower: { $replaceAll: { input: "$name", find: " ",
replacement: "." } } },
          "@college.edu"
        ]
      }
```

```
    }
  }
])

// Create unique index

db.students.createIndex({ email: 1 }, { unique: true })
email_1
```

5. Create Text Index

Create text index for text search:

```
college_db> db.students.createIndex({ name: "text", department: "text" })
```

Search using text index:

```
college_db> db.students.find({ $text: { $search: "Computer Science" } })
```

6. Create Multikey Index (for arrays)

```
college_db> db.students.createIndex({ subjects: 1 })
```

Benefits: Efficient queries on array elements

7. Check Query Performance - explain()

Without index:

```
college_db> db.students.find({ cgpa: { $gt: 8.5 }
}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'college_db.students',
    parsedQuery: { cgpa: { '$gt': 8.5 } },
    indexFilterSet: false,
    queryHash: 'D7E0032F',
    planCacheShapeHash: 'D7E0032F',
    planCacheKey: 'FF018592',
    optimizationTimeMillis: 2,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { cgpa: { '$gt': 8.5 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 3,
    executionTimeMillis: 6,
    totalKeysExamined: 0,
```

```
    totalDocsExamined: 6,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { cgpa: { '$gt': 8.5 } },
      nReturned: 3,
      executionTimeMillisEstimate: 0,
      works: 7,
      advanced: 3,
      needTime: 3,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 6
    }
  },
  queryShapeHash:
'1AB66686AF78BC046FAA1D8EDD7A0DF29ECD126B8B521D1E7620AD70FD7038E9',
  command: {
    find: 'students',
    filter: { cgpa: { '$gt': 8.5 } },
    '$db': 'college_db'
  },
  serverInfo: {
    host: 'Asus_ExpertBook',
    port: 27017,
    version: '8.2.1',
    gitVersion: '3312bdcf28aa65f5930005e21c2cb130f648b8c3'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
```

With index:

```
college_db> db.students.createIndex({ cgpa: 1 })
cgpa_1

db.students.find({ cgpa: { $gt: 8.5 } }).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'college_db.students',
    parsedQuery: { cgpa: { '$gt': 8.5 } },
    indexFilterSet: false,
```

```
    queryHash: 'D7E0032F',
    planCacheShapeHash: 'D7E0032F',
    planCacheKey: '9291F2C7',
    optimizationTimeMillis: 2,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { cgpa: 1 },
        indexName: 'cgpa_1',
        isMultiKey: false,
        multiKeyPaths: { cgpa: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { cgpa: [ '(8.5, inf]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 3,
    executionTimeMillis: 4,
    totalKeysExamined: 3,
    totalDocsExamined: 3,
    executionStages: {
      isCached: false,
      stage: 'FETCH',
      nReturned: 3,
      executionTimeMillisEstimate: 0,
      works: 4,
      advanced: 3,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 3,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 3,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 3,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
```

```
        keyPattern: { cgpa: 1 },
        indexName: 'cgpa_1',
        isMultiKey: false,
        multiKeyPaths: { cgpa: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { cgpa: [ '(8.5, inf]' ] },
        keysExamined: 3,
        seeks: 1,
        dupsTested: 0,
        dupsDropped: 0
      }
    }
  },
  queryShapeHash:
'1AB66686AF78BC046FAA1D8EDD7A0DF29ECD126B8B521D1E7620AD70FD7038E9',
  command: {
    find: 'students',
    filter: { cgpa: { '$gt': 8.5 } },
    '$db': 'college_db'
  },
  serverInfo: {
    host: 'Asus_ExpertBook',
    port: 27017,
    version: '8.2.1',
    gitVersion: '3312bdcf28aa65f5930005e21c2cb130f648b8c3'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
```

Compare: Look at executionTimeMillis and totalDocsExamined

8. Drop Index

```
// Drop specific index
db.students.dropIndex({ department: 1 })

// Or by name
db.students.dropIndex("department_1")

// Drop all indexes except _id
db.students.dropIndexes()
```