
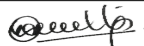


**COURSE LABORATORY MANUAL****A. LABORATORY OVERVIEW**

Degree:	<b>B.E</b>	Programme:	CD
Semester:	4	Academic Year:	2024-25
Laboratory Title:	MongoDB	Laboratory Code:	BDS456B
L-T-P-S:	0-0-2-0	Duration of SEE:	3 Hrs
Total Contact Hours:	24	SEE Marks:	50*
Credits:	01	CIE Marks:	50
Lab Manual Author:	<b>Prof.Sapna K N</b>	Sign 	Dt : 6/02/2025
Checked By:	<b>Prof. Rohith H P</b>	Sign 	Dt : 6/02/2025

\*The SEE will be conducted for 100 marks and proportionally reduced to 50 marks.

**B. DESCRIPTION****1. PREREQUISITES:**

- Principles of Programming using C (BPOPS103)
- Data Structures and Applications (BCS304)
- Data Structures Lab (BCSL305)

**2. BASE COURSE:**

- MongoDB

**3. COURSE OUTCOMES:**

At the end of the course, the student will be able to;

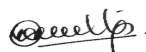
1. Make use of MongoDB commands and queries.
2. Illustrate the role of aggregate pipelines to extract data.
3. Demonstrate optimization of queries by creating indexes.
4. Develop aggregate pipelines for text search in collections.

**4. RESOURCES REQUIRED:**

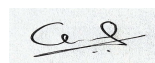
- Hardware requirements:
  - a) Processor: Pentium 4
  - b) Memory: 1 GB RAM
- Software requirements:
  - a) Operating System: Ubuntu / Windows XP/ Windows 7



Prepared by: Prof.Sapna K N



Checked by: Prof. Rohith H P



HOD

**COURSE LABORATORY MANUAL****5. RELEVANCE OF THE COURSE:**

- Database Management Systems(BCS403)
- Theory of Computation (BCS503)
- Big Data Analytics (BAD601)
- Natural Language Processing (NLP)( BAD613B )
- Introduction to DBMS (BCD755A)

**6. GENERAL INSTRUCTIONS:**

- Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage if caused is punishable.
- Students are required to carry their observation / programs book with completed exercises while entering the lab.
- Students are supposed to occupy the machines allotted to them and are not supposed to talk or make noise in the lab.
- Lab can be used in free time / lunch hours by the students who need to use the systems should take prior permission from the lab in-charge/ Lab instructors.
- Lab records need to be submitted on or before date of submission.
- Students are not supposed to use CDs/ Pen drives, etc.,

**7. CONTENTS:**

Expt No.	Title of the Experiments	RBT	CO
1	a. Illustration of Where Clause, AND,OR operations in MongoDB. b. Execute the Commands of MongoDB and operations in MongoDB : Insert, Query, Update, Delete and Projection. (Note: use any collection)	L2	CO1
2	a. Develop a MongoDB query to select certain fields and ignore some fields of the documents from any collection. b. Develop a MongoDB query to display the first 5 documents from the results obtained in a. [use of limit and find]	L3	CO1
3	a. Execute query selectors (comparison selectors, logical selectors ) and list out the results on any collection b. Execute query selectors (Geospatial selectors, Bitwise selectors ) and list out the results on any collection	L2	CO1
4	Create and demonstrate how projection operators (\$, \$elematch and \$slice) would be used in the MondoDB.	L3	CO1
5	Execute Aggregation operations (\$avg, \$min,\$max, \$push, \$addToSet etc.). students encourage to execute several queries to demonstrate various aggregation operators)	L2	CO2
6	Execute Aggregation Pipeline and its operations (pipeline must contain \$match, \$group, \$sort, \$project, \$skip etc. students encourage to execute several queries to demonstrate various aggregation operators)	L2	CO2

## COURSE LABORATORY MANUAL

7	a. Find all listings with listing_url, name, address, host_picture_url in the listings And Reviews collection that have a host with a picture url b. Using E-commerce collection write a query to display reviews summary	L2	CO3
8	a. Demonstrate creation of different types of indexes on collection (unique, sparse, compound and multikey indexes) b. Demonstrate optimization of queries using indexes.	L3	CO3
9	a. Develop a query to demonstrate Text search using catalog data collection for a given word b. Develop queries to illustrate excluding documents with certain words and phrases	L3	CO4
10	Develop an aggregation pipeline to illustrate Text search on Catalog data collection	L3	CO4
14	Open ended experiment – 1		
15	Open ended experiment – 2		

### 8. REFERENCE:

- BOOK 1:** "MongoDB: The Definitive Guide", Kristina chodorow, 2nd ed O'REILLY, 2013.
- BOOK 2:** "MongoDB in Action" by KYLE BANKER et. al. 2nd ed, Manning publication, 2016
- BOOK 3:** "MongoDB Complete Guide" by Manu Sharma 1st ed, bpb publication, 2023.
- Installation of MongoDB Video:** <https://www.youtube.com/watch?v=dEm2AS5amyA>
- video on Aggregation:** <https://www.youtube.com/watch?v=vx1C8EyTa7Y>
- MongoDB in action book Code download URL:** <https://www.manning.com/downloads/529>
- MongoDB Exercise URL:** <https://www.w3resource.com/mongodb-exercises/>

### C. EVALUATION SCHEME

For CBCS 2021 scheme:

- Laboratory Components: **30 Marks**  
 Observation Writeup – 10 Marks  
 Lab Conduction – 10 Marks  
 Record Writing – 10 Marks.  
 Total --- **30 Marks**  
 (Note: **Minimum marks** to be scored by the student to appear for SEE – **12 Marks**)
- Laboratory IA test: **20 Marks**  
 IA test shall be conducted for 100 Marks  
 Write up & Conduction---- 60 Marks,  
 Viva---- 40 Marks  
 Total---- 100 Marks , and scaled down to **20 Marks**  
 Note: **Minimum marks** to be scored by the student to appear for SEE – **8 Marks**)  
 Continuous Internal Evaluation (CIE) = **30+20 = 50 Marks.**
- SEE : 50\* Marks  
 (\*The SEE will be conducted for 100 marks and proportionally reduced to 50 marks)

## COURSE LABORATORY MANUAL

### D1. ARTICULATION MATRIX

Mapping of CO to PO

COs	POs											
	1	2	3	4	5	6	7	8	9	10	11	12
1. Make use of MangoDB commands and queries.	3	2	-	-	-	-	-	-	-	-	2	3
2. Illustrate the role of aggregate pipelines to extract data.	3	2	-	-	-	-	-	-	-	-	2	3
3. Demonstrate optimization of queries by creating indexes	3	2	2	-	-	-	-	-	-	-	2	3
4. Develop aggregate pipelines for text search in collections	3	2	2	-	-	-	-	-	-	-	2	3

*Note: Mappings in the Tables D1 (above) and D2 (below) are done by entering in the corresponding cell the Correlation Levels in terms of numbers. For Slight (Low): 1, Moderate (Medium): 2, Substantial (High): 3 and for no correlation: “ - ”.*

### D2. ARTICULATION MATRIX CO v/s PSO

Mapping of CO to PSO

COs	PSOs		
	1	2	3
1 . Make use of MangoDB commands and queries.	-	2	-
2. Illustrate the role of aggregate pipelines to extract data.	-	2	-
3 Demonstrate optimization of queries by creating indexes	-	2	-
4 Develop aggregate pipelines for text search in collections.	-	2	-

### E. EXPERIMENTS

1. EXPERIMENT NO:1

2. TITLE: a. Illustration of Where Clause, AND,OR operations in MongoDB.

b. Execute the Commands of MongoDB and operations in MongoDB : Insert, Query, Update, Delete and Projection. (Note: use any collection)

3. LEARNING OBJECTIVES:

- Understand and Apply MongoDB Query Operations
- Perform CRUD Operations in MongoDB

4. AIM:

- To demonstrate fundamental **MongoDB operations** using a sample collection.

5. MATERIAL / EQUIPMENT REQUIRED:

6. THEORY / HYPOTHESIS:

- In MongoDB, there is no direct WHERE clause like in SQL, but similar operations can be performed using query filters such as AND and OR.. MongoDB uses different ommands for Insert, Query, Update, Delete, and Projection

7. FORMULA / CALCULATIONS:

**COURSE LABORATORY MANUAL****8. PROCEDURE / PROGRAMME / ACTIVITY:****1A.**

use newDB

`db.createCollection("ProgrammingBooks")``db.ProgrammingBooks.insertMany([``{ title: "Clean Code", author: "Robert C. Martin", category: "Software Development", year: 2008 },``{ title: "JavaScript: The Good Parts", author: "Douglas Crockford", category: "JavaScript", year: 2008 },``{ title: "Design Patterns", author: "Erich Gamma", category: "Software Design", year: 1994 },``{ title: "Introduction to Algorithms", author: "Thomas H. Cormen", category: "Algorithms", year: 2009 },``{ title: "Python Crash Course", author: "Eric Matthes", category: "Python", year: 2015 }``]);`

//WHERE clause equivalent

`db.ProgrammingBooks.find({ year: 2008 }).pretty()`

//Using the \$and Operator

`db.ProgrammingBooks.find({` `$and: [` `{ category: "Software Development" },` `{ year: 2008 }` `]``}).pretty()`

//Using the \$or Operator

`db.ProgrammingBooks.find({` `$or: [` `{ category: "JavaScript" },` `{ year: 2015 }` `]``}).pretty()`

//Combining \$and and \$or Operators

`db.ProgrammingBooks.find({` `$or: [` `{` `$and: [` `{ category: "Software Development" },` `{ year: { $gt: 2007 } }` `]` `},` `{ category: "Python" }` `]``}).pretty()`

## **COURSE LABORATORY MANUAL**

1b.

use ProgBooksDB

```
db.createCollection("ProgrammingBooks")
```

```
// insert a new document into the ProgrammingBooks collection:
```

```
db.ProgrammingBooks.insertOne({  
  title: "The Pragmatic Programmer: Your Journey to Mastery",  
  author: "David Thomas, Andrew Hunt",  
  category: "Software Development",  
  year: 1999  
})
```

```
db.ProgrammingBooks.insertMany([  
  {  
    title: "Clean Code: A Handbook of Agile Software Craftsmanship",  
    author: "Robert C. Martin",  
    category: "Software Development",  
    year: 2008  
  },  
  {  
    title: "JavaScript: The Good Parts",  
    author: "Douglas Crockford",  
    category: "JavaScript",  
    year: 2008  
  },  
  {  
    title: "Design Patterns: Elements of Reusable Object-Oriented Software",  
    author: "Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides",  
    category: "Software Design",  
    year: 1994  
  },  
  {  
    title: "Introduction to Algorithms",  
    author: "Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein",  
    category: "Algorithms",  
    year: 1990  
  },  
  {  
    title: "Python Crash Course: A Hands-On, Project-Based Introduction to Programming",  
    author: "Eric Matthes",  
    category: "Python",  
    year: 2015  
  }  
])
```

```
//Find All Documents
```

```
db.ProgrammingBooks.find().pretty()
```

**COURSE LABORATORY MANUAL**

```
//Find Documents Matching a Condition
```

```
db.ProgrammingBooks.find({ year: { $gt: 2000 } }).pretty()
```

```
//Update a Single Document
```

```
db.ProgrammingBooks.updateOne(  
  { title: "Clean Code: A Handbook of Agile Software Craftsmanship" },  
  { $set: { author: "Robert C. Martin (Uncle Bob)" } }  
)
```

```
//verify by displaying books published in year 2008
```

```
db.ProgrammingBooks.find({ year: { $eq: 2008 } }).pretty()
```

```
//another way to verify
```

```
db.ProgrammingBooks.find({ author: { $regex: "Robert*" } }).pretty()
```

```
// Update Multiple Documents
```

```
db.ProgrammingBooks.updateMany(  
  { year: { $lt: 2010 } },  
  { $set: { category: "Classic Programming Books" } }  
)
```

```
//Delete a Single Document
```

```
db.ProgrammingBooks.deleteOne({ title: "JavaScript: The Good Parts" })  
{ acknowledged: true, deletedCount: 1 }
```

```
//Verify to see document is deleted
```

```
db.ProgrammingBooks.find({ title: "JavaScript: The Good Parts" }).pretty()
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

•

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

•

11. GRAPHS / OUTPUTS:

switched to db newDB

```
{ "ok" : 1 }
```

```
{
```

```
  "acknowledged" : true,
```

```
  "insertedIds" : [
```

```
    ObjectId("679741b869061db4760ca3ca"),
```

```
    ObjectId("679741b869061db4760ca3cb"),
```

```
    ObjectId("679741b869061db4760ca3cc"),
```

```
    ObjectId("679741b869061db4760ca3cd"),
```

```
    ObjectId("679741b869061db4760ca3ce")
```

```
  ]
```

```
}
```

```
{
```

```
  "_id" : ObjectId("679741b869061db4760ca3ca"),
```

```
  "title" : "Clean Code",
```



## **COURSE LABORATORY MANUAL**

```
"author" : "Robert C. Martin",
"category" : "Software Development",
"year" : 2008
}
{
  "_id" : ObjectId("679741b869061db4760ca3cb"),
  "title" : "JavaScript: The Good Parts",
  "author" : "Douglas Crockford",
  "category" : "JavaScript",
  "year" : 2008
}
{
  "_id" : ObjectId("679741b869061db4760ca3ca"),
  "title" : "Clean Code",
  "author" : "Robert C. Martin",
  "category" : "Software Development",
  "year" : 2008
}
{
  "_id" : ObjectId("679741b869061db4760ca3cb"),
  "title" : "JavaScript: The Good Parts",
  "author" : "Douglas Crockford",
  "category" : "JavaScript",
  "year" : 2008
}
{
  "_id" : ObjectId("679741b869061db4760ca3ce"),
  "title" : "Python Crash Course",
  "author" : "Eric Matthes",
  "category" : "Python",
  "year" : 2015
}
{
  "_id" : ObjectId("679741b869061db4760ca3ca"),
  "title" : "Clean Code",
  "author" : "Robert C. Martin",
  "category" : "Software Development",
  "year" : 2008
}
{
  "_id" : ObjectId("679741b869061db4760ca3ce"),
  "title" : "Python Crash Course",
  "author" : "Eric Matthes",
  "category" : "Python",
  "year" : 2015
}
```

## 12. RESULTS & CONCLUSIONS:

-



## **COURSE LABORATORY MANUAL**

13. LEARNING OUTCOMES :

- 

14. APPLICATION AREAS:

- 

15. REMARKS:

- 

1. EXPERIMENT NO:2

2. TITLE: 2. a. Develop a MongoDB query to select certain fields and ignore some fields of the documents from any collection.

b. Develop a MongoDB query to display the first 5 documents from the results obtained in a. [use of limit and find]

3. LEARNING OBJECTIVES:

- Understand MongoDB Query Projection
- Use of `find()` Method
- Apply `limit()` to Control Output
- -

4. AIM:

- The aim of these MongoDB queries is to **efficiently retrieve specific data** from a collection by selecting only the required fields and limiting the number of documents returned.

5. MATERIAL / EQUIPMENT REQUIRED:

- 

6. THEORY / HYPOTHESIS:

- It allows querying documents efficiently using the `find()` method, along with projection and limit functionalities.
- 

7. FORMULA / CALCULATIONS:

- -
- -

8. PROCEDURE / PROGRAMME / ACTIVITY:

2a.

use MoviesDB

```
db.createCollection("Movies")
```

```
db.Movies.insertMany([
```

```
  { title: "Inception", director: "Christopher Nolan", genre: "Science Fiction", year: 2010, ratings: { imdb: 8.8, rottenTomatoes: 87 } },
```

```
  { title: "The Matrix", director: "Wachowskis", genre: "Science Fiction", year: 1999, ratings: { imdb: 8.7, rottenTomatoes: 87 } },
```

```
  { title: "The Godfather", director: "Francis Ford Coppola", genre: "Crime", year: 1972, ratings: { imdb: 9.2, rottenTomatoes: 97 } } ]);
```

```
//To select only the title and director fields from the Movies collection
```

```
db.Movies.find({}, { title: 1, director: 1, _id: 0 })
```

```
//To exclude the ratings field from the results:
```

## COURSE LABORATORY MANUAL

```
db.Movies.find({}, { ratings: 0 })
```

```
//Combining Filter and Projection
```

```
db.Movies.find({ director: "Christopher Nolan" }, { title: 1, year: 1, _id: 0 })
```

2b.

```
use MoviesDB
```

```
db.createCollection("Movies")
```

```
db.Movies.insertMany([
```

```
  { title: "Inception", director: "Christopher Nolan", genre: "Science Fiction", year: 2010, ratings:
```

```
  { imdb: 8.8, rottenTomatoes: 87 } },
```

```
  { title: "The Matrix", director: "Wachowskis", genre: "Science Fiction", year: 1999, ratings:
```

```
  { imdb: 8.7, rottenTomatoes: 87 } },
```

```
  { title: "The Godfather", director: "Francis Ford Coppola", genre: "Crime", year: 1972, ratings:
```

```
  { imdb: 9.2, rottenTomatoes: 97 } },
```

```
  { title: "Pulp Fiction", director: "Quentin Tarantino", genre: "Crime", year: 1994, ratings: { imdb:
```

```
8.9, rottenTomatoes: 92 } },
```

```
  { title: "The Shawshank Redemption", director: "Frank Darabont", genre: "Drama", year: 1994,
```

```
ratings: { imdb: 9.3, rottenTomatoes: 91 } },
```

```
  { title: "The Dark Knight", director: "Christopher Nolan", genre: "Action", year: 2008, ratings:
```

```
  { imdb: 9.0, rottenTomatoes: 94 } },
```

```
  { title: "Fight Club", director: "David Fincher", genre: "Drama", year: 1999, ratings: { imdb: 8.8,
```

```
rottenTomatoes: 79 } }]
```

```
);
```

```
//Query with Projection and Limit
```

```
db.Movies.find({}, { title: 1, director: 1, year: 1, _id: 0 }).limit(5)
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

•

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

•

11. GRAPHS / OUTPUTS:

- MoviesDB> { ok: 1 }

- MoviesDB> ... .. {

acknowledged: true,

insertedIds: {

'0': ObjectId('679c5d3aa2d9428c7d6b128c'),

'1': ObjectId('679c5d3aa2d9428c7d6b128d'),

'2': ObjectId('679c5d3aa2d9428c7d6b128e')

}

}

- MoviesDB> [

{ title: 'Inception', director: 'Christopher Nolan' },

{ title: 'The Matrix', director: 'Wachowskis' },

## COURSE LABORATORY MANUAL

```
{ title: 'The Godfather', director: 'Francis Ford Coppola' }
```

```
]
```

- MoviesDB> [

```
{
```

```
  _id: ObjectId('679c5d3aa2d9428c7d6b128c'),
```

```
  title: 'Inception',
```

```
  director: 'Christopher Nolan',
```

```
  genre: 'Science Fiction',
```

```
  year: 2010
```

```
},
```

```
{
```

```
  _id: ObjectId('679c5d3aa2d9428c7d6b128d'),
```

```
  title: 'The Matrix',
```

```
  director: 'Wachowskis',
```

```
  genre: 'Science Fiction',
```

```
  year: 1999
```

```
},
```

```
{
```

```
  _id: ObjectId('679c5d3aa2d9428c7d6b128e'),
```

```
  title: 'The Godfather',
```

```
  director: 'Francis Ford Coppola',
```

```
  genre: 'Crime',
```

```
  year: 1972
```

```
}
```

```
]
```

```
MoviesDB> [ { title: 'Inception', year: 2010 } ]
```

- 
- 

### 12. RESULTS & CONCLUSIONS:

- 

### 13. LEARNING OUTCOMES :

- 

### 14. APPLICATION AREAS:

- 

### 15. REMARKS:

- 

### 1. EXPERIMENT NO:3a and 3b

2. TITLE: **3 a.** Execute query selectors (comparison selectors, logical selectors ) and list out the results on any collection

**b.** Execute query selectors (Geospatial selectors, Bitwise selectors ) and list out the results on any collection

### 3. LEARNING OBJECTIVES:

- Understand the purpose of **query selectors** in MongoDB. Understand the purpose of **geospatial selectors** and when to use them. Understand **bitwise selectors** and their role in querying bitwise data. By executing **comparison, logical, geospatial, and bitwise selectors**, learners will Understand Querying in MongoDB

**COURSE LABORATORY MANUAL**

•

## 4. AIM:

- The aim of this program is to develop proficiency in using **MongoDB query selectors** to retrieve, filter, and manipulate data efficiently.

•

## 5. MATERIAL / EQUIPMENT REQUIRED:

•

## 6. THEORY / HYPOTHESIS:

- Comparison operators in MongoDB allow filtering documents based on specific values.
- Geospatial queries help in working with location-based data
- Bitwise selectors allow querying documents based on binary values

## 7. FORMULA / CALCULATIONS:

•

## 8. PROCEDURE / PROGRAMME / ACTIVITY:

3A.

## USE COMPANYDB

//Create the Employees Collection and Insert Documents

db.Employees.insertMany([

{ name: "Alice", age: 30, department: "HR", salary: 50000, joinDate: new Date("2015-01-15") },

{ name: "Bob", age: 24, department: "Engineering", salary: 70000, joinDate: new Date("2019-03-10") },

{ name: "Charlie", age: 29, department: "Engineering", salary: 75000, joinDate: new Date("2017-06-23") },

{ name: "David", age: 35, department: "Marketing", salary: 60000, joinDate: new Date("2014-11-01") },

{ name: "Eve", age: 28, department: "Finance", salary: 80000, joinDate: new Date("2018-08-19") } ])

])

// \$eq

db.Employees.find({ department: { \$eq: "Engineering" } }).pretty()

// \$ne

db.Employees.find({ department: { \$ne: "HR" } }).pretty()

// \$gt

db.Employees.find({ age: { \$gt: 30 } }).pretty()

// \$lt

db.Employees.find({ salary: { \$lt: 70000 } }).pretty()

// \$gte

db.Employees.find({ joinDate: { \$gte: new Date("2018-01-01") } }).pretty()

// \$lte

db.Employees.find({ age: { \$lte: 28 } }).pretty()

//Queries Using Logical Selectors

## **COURSE LABORATORY MANUAL**

```
// $and
db.Employees.find({
  $and: [
    { department: "Engineering" },
    { salary: { $gt: 70000 } }
  ]
}).pretty()

// $or
db.Employees.find({
  $or: [
    { department: "HR" },
    { salary: { $lt: 60000 } }
  ]
}).pretty()

// $not
db.Employees.find({
  department: {
    $not: { $eq: "Engineering" }
  }
}).pretty()

// $nor
db.Employees.find({
  $nor: [
    { department: "HR" },
    { salary: { $gt: 75000 } }
  ]
}).pretty()

3B.

  • -use geoDatabase

db.Places.insertMany([
  { name: "Central Park", location: { type: "Point", coordinates: [-73.9654, 40.7829] } },
  { name: "Times Square", location: { type: "Point", coordinates: [-73.9851, 40.7580] } },
  { name: "Brooklyn Bridge", location: { type: "Point", coordinates: [-73.9969, 40.7061] } },
  { name: "Empire State Building", location: { type: "Point", coordinates: [-73.9857, 40.7488] } },
  { name: "Statue of Liberty", location: { type: "Point", coordinates: [-74.0445, 40.6892] } }
])

// Create a geospatial index
db.Places.createIndex({ location: "2dsphere" })

// Find places near a specific coordinate, for example, near Times Square.
db.Places.find({
  location: {
```

## **COURSE LABORATORY MANUAL**

```
$near: {
  $geometry: {
    type: "Point",
    coordinates: [-73.9851, 40.7580]
  },
  $maxDistance: 5000 // distance in meters
}
}).pretty()

// $geoWithin
db.Places.find({
  location: {
    $geoWithin: {
      $geometry: {
        type: "Polygon",
        coordinates: [
          [
            [-70.016, 35.715],
            [-74.014, 40.717],
            [-73.990, 40.730],
            [-73.990, 40.715],
            [-70.016, 35.715]
          ]
        ]
      }
    }
  }
}).pretty()

//Bitwise Selectors

use techDB

db.Devices.insertMany([
  { name: "Device A", status: 5 }, // Binary: 0101
  { name: "Device B", status: 3 }, // Binary: 0011
  { name: "Device C", status: 12 }, // Binary: 1100
  { name: "Device D", status: 10 }, // Binary: 1010
  { name: "Device E", status: 7 } // Binary: 0111
])

//$bitsAllSet
//Find devices where the binary status has both the 1st and 3rd bits set

db.Devices.find({
  status: { $bitsAllSet: [0, 2] }
}).pretty()
```

## COURSE LABORATORY MANUAL

```
// $bitsAnySet
// Find devices where the binary status has at least the 2nd bit set
db.Devices.find({
  status: { $bitsAnySet: [1] }
}).pretty()
```

```
// $bitsAllClear
// Find devices where the binary status has both the 2nd and 4th bits clear
db.Devices.find({
  status: { $bitsAllClear: [1, 3] }
}).pretty()
```

```
// $bitsAnyClear
// Find devices where the binary status has at least the 1st bit clear
db.Devices.find({
  status: { $bitsAnyClear: [0] }
}).pretty()
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

11. GRAPHS / OUTPUTS:

3a. Output

- mycompiler\_mongodb> ... .. {
 acknowledged: true,
 insertedIds: {
 '0': ObjectId('679c6a68cf105230266b128c'),
 '1': ObjectId('679c6a68cf105230266b128d'),
 '2': ObjectId('679c6a68cf105230266b128e'),
 '3': ObjectId('679c6a68cf105230266b128f'),
 '4': ObjectId('679c6a68cf105230266b1290')
 }
 }
 }
 • mycompiler\_mongodb> [
 {
 \_id: ObjectId('679c6a68cf105230266b128d'),
 name: 'Bob',
 age: 24,
 department: 'Engineering',
 salary: 70000,
 joinDate: ISODate('2019-03-10T00:00:00.000Z')
 },
 {
 \_id: ObjectId('679c6a68cf105230266b128e'),
 name: 'Charlie',
 age: 29,
 department: 'Engineering',
 salary: 75000,
 joinDate: ISODate('2017-06-23T00:00:00.000Z')
 }
 ]



## COURSE LABORATORY MANUAL

```
}
]

mycompiler_mongodb> [
{
  _id: ObjectId('679c6a68cf105230266b128d'),
  name: 'Bob',
  age: 24,
  department: 'Engineering',
  salary: 70000,
  joinDate: ISODate('2019-03-10T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128e'),
  name: 'Charlie',
  age: 29,
  department: 'Engineering',
  salary: 75000,
  joinDate: ISODate('2017-06-23T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128f'),
  name: 'David',
  age: 35,
  department: 'Marketing',
  salary: 60000,
  joinDate: ISODate('2014-11-01T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b1290'),
  name: 'Eve',
  age: 28,
  department: 'Finance',
  salary: 80000,
  joinDate: ISODate('2018-08-19T00:00:00.000Z')
}
]

mycompiler_mongodb> [
{
  _id: ObjectId('679c6a68cf105230266b128f'),
  name: 'David',
  age: 35,
  department: 'Marketing',
  salary: 60000,
  joinDate: ISODate('2014-11-01T00:00:00.000Z')
}
]
```

## **COURSE LABORATORY MANUAL**

```
mycompiler_mongodb> [
{
  _id: ObjectId('679c6a68cf105230266b128c'),
  name: 'Alice',
  age: 30,
  department: 'HR',
  salary: 50000,
  joinDate: ISODate('2015-01-15T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128f'),
  name: 'David',
  age: 35,
  department: 'Marketing',
  salary: 60000,
  joinDate: ISODate('2014-11-01T00:00:00.000Z')
}
]
```

```
mycompiler_mongodb> [
{
  _id: ObjectId('679c6a68cf105230266b128d'),
  name: 'Bob',
  age: 24,
  department: 'Engineering',
  salary: 70000,
  joinDate: ISODate('2019-03-10T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b1290'),
  name: 'Eve',
  age: 28,
  department: 'Finance',
  salary: 80000,
  joinDate: ISODate('2018-08-19T00:00:00.000Z')
}
]
```

```
mycompiler_mongodb> [
{
  _id: ObjectId('679c6a68cf105230266b128d'),
  name: 'Bob',
  age: 24,
  department: 'Engineering',
  salary: 70000,
  joinDate: ISODate('2019-03-10T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b1290'),
  name: 'Eve',

```

## **COURSE LABORATORY MANUAL**

```
age: 28,
department: 'Finance',
salary: 80000,
joinDate: ISODate('2018-08-19T00:00:00.000Z')
}
]

mycompiler_mongodb> ... .. [
{
  _id: ObjectId('679c6a68cf105230266b128e'),
  name: 'Charlie',
  age: 29,
  department: 'Engineering',
  salary: 75000,
  joinDate: ISODate('2017-06-23T00:00:00.000Z')
}
]

mycompiler_mongodb> ... .. [
{
  _id: ObjectId('679c6a68cf105230266b128c'),
  name: 'Alice',
  age: 30,
  department: 'HR',
  salary: 50000,
  joinDate: ISODate('2015-01-15T00:00:00.000Z')
}
]

mycompiler_mongodb>
mycompiler_mongodb>
mycompiler_mongodb> ... .. [
{
  _id: ObjectId('679c6a68cf105230266b128c'),
  name: 'Alice',
  age: 30,
  department: 'HR',
  salary: 50000,
  joinDate: ISODate('2015-01-15T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128f'),
  name: 'David',
  age: 35,
  department: 'Marketing',
  salary: 60000,
  joinDate: ISODate('2014-11-01T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b1290'),
```

## **COURSE LABORATORY MANUAL**

```
name: 'Eve',
age: 28,
department: 'Finance',
salary: 80000,
joinDate: ISODate('2018-08-19T00:00:00.000Z')
}
]
```

```
mycompiler_mongodb> ... .. [
{
  _id: ObjectId('679c6a68cf105230266b128d'),
  name: 'Bob',
  age: 24,
  department: 'Engineering',
  salary: 70000,
  joinDate: ISODate('2019-03-10T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128e'),
  name: 'Charlie',
  age: 29,
  department: 'Engineering',
  salary: 75000,
  joinDate: ISODate('2017-06-23T00:00:00.000Z')
},
{
  _id: ObjectId('679c6a68cf105230266b128f'),
  name: 'David',
  age: 35,
  department: 'Marketing',
  salary: 60000,
  joinDate: ISODate('2014-11-01T00:00:00.000Z')
}
]
```

3b output:

```
geoDatabase> ... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c6e8956ec7ca0e46b128c'),
    '1': ObjectId('679c6e8956ec7ca0e46b128d'),
    '2': ObjectId('679c6e8956ec7ca0e46b128e'),
    '3': ObjectId('679c6e8956ec7ca0e46b128f'),
    '4': ObjectId('679c6e8956ec7ca0e46b1290')
  }
}
geoDatabase>
```

## **COURSE LABORATORY MANUAL**

```
geoDatabase> location_2dsphere
```

```
geoDatabase> ... .. [
{
  _id: ObjectId('679c6e8956ec7ca0e46b128d'),
  name: 'Times Square',
  location: { type: 'Point', coordinates: [ -73.9851, 40.758 ] }
},
{
  _id: ObjectId('679c6e8956ec7ca0e46b128f'),
  name: 'Empire State Building',
  location: { type: 'Point', coordinates: [ -73.9857, 40.7488 ] }
},
{
  _id: ObjectId('679c6e8956ec7ca0e46b128c'),
  name: 'Central Park',
  location: { type: 'Point', coordinates: [ -73.9654, 40.7829 ] }
}
]
```

```
geoDatabase> ... ..
[
{
  _id: ObjectId('679c6e8956ec7ca0e46b128e'),
  name: 'Brooklyn Bridge',
  location: { type: 'Point', coordinates: [ -73.9969, 40.7061 ] }
}
]
```

```
geoDatabase> switched to db techDB
```

```
techDB>
```

```
techDB> ... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c6e8b56ec7ca0e46b1291'),
    '1': ObjectId('679c6e8b56ec7ca0e46b1292'),
    '2': ObjectId('679c6e8b56ec7ca0e46b1293'),
    '3': ObjectId('679c6e8b56ec7ca0e46b1294'),
    '4': ObjectId('679c6e8b56ec7ca0e46b1295')
  }
}
>
```

```
techDB> ... .. [
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1291'),
  name: 'Device A',
  status: 5
},
]
```

## **COURSE LABORATORY MANUAL**

```
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1295'),
  name: 'Device E',
  status: 7
}
]
```

```
techDB> ... .. [
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1292'),
  name: 'Device B',
  status: 3
},
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1294'),
  name: 'Device D',
  status: 10
},
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1295'),
  name: 'Device E',
  status: 7
}
]
```

```
techDB> ... .. [
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1291'),
  name: 'Device A',
  status: 5
}
]
```

```
techDB> ... .. [
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1293'),
  name: 'Device C',
  status: 12
},
{
  _id: ObjectId('679c6e8b56ec7ca0e46b1294'),
  name: 'Device D',
  status: 10
}
]
techDB>
```

### 12. RESULTS & CONCLUSIONS:

- 

### 13. LEARNING OUTCOMES :

- 

### 14. APPLICATION AREAS:

## COURSE LABORATORY MANUAL

• 15. REMARKS: •
------------------------

1. EXPERIMENT NO:4
2. TITLE: Create and demonstrate how projection operators (\$, \$elemMatch and \$slice) would be used in the MondoDB
3. LEARNING OBJECTIVES: <ul style="list-style-type: none"> <li>Learn what projection is and why it is used in MongoDB queries. Learn how to using the \$ Projection Operator , \$elemMatch Projection Operator , \$slice Projection Operator</li> </ul>
4. AIM: <ul style="list-style-type: none"> <li>-The aim of this program is to <b>understand and apply projection operators (\$, \$elemMatch, and \$slice) in MongoDB</b> to efficiently retrieve specific portions of data from documents</li> <li>-</li> </ul>
5.
6. THEORY / HYPOTHESIS:
7. FORMULA / CALCULATIONS:
8. PROCEDURE / PROGRAMME / ACTIVITY: <p style="margin-left: 40px;">use retailDB</p> <pre> db.Products.insertMany([ {   name: "Laptop",   brand: "BrandA",   features: [     { name: "Processor", value: "Intel i7" },     { name: "RAM", value: "16GB" },     { name: "Storage", value: "512GB SSD" }   ],   reviews: [     { user: "Alice", rating: 5, comment: "Excellent!" },     { user: "Bob", rating: 4, comment: "Very good" },     { user: "Charlie", rating: 3, comment: "Average" }   ] }, {   name: "Smartphone",   brand: "BrandB",   features: [     { name: "Processor", value: "Snapdragon 888" },     { name: "RAM", value: "8GB" },     { name: "Storage", value: "256GB" }   ] }, </pre>



## **COURSE LABORATORY MANUAL**

```
reviews: [
  { user: "Dave", rating: 4, comment: "Good phone" },
  { user: "Eve", rating: 2, comment: "Not satisfied" }
]
}
```

//The \$ Projection Operator

```
db.Products.find(
  { name: "Laptop", "reviews.user": "Alice" },
  { "reviews.$": 1 }
).pretty()
```

//The \$elemMatch Projection Operator

```
db.Products.find(
  { name: "Laptop" },
  { reviews: { $elemMatch: { rating: { $gt: 4 } } } }
).pretty()
```

//The \$slice Projection Operator

```
db.Products.find(
  { name: "Smartphone" },
  { reviews: { $slice: 1 } }
).pretty()
```

//Multiple Projection Operators

```
db.Products.find(
  { name: "Laptop" },
  {
    name: 1,
    features: { $slice: 2 },
    reviews: { $elemMatch: { rating: 5 } }
  }
).pretty()
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

11. GRAPHS / OUTPUTS:

- retailDB>

```
retailDB> ..... {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c70f818e5e77f676b128c'),
    '1': ObjectId('679c70f818e5e77f676b128d')
  }
}
```

**COURSE LABORATORY MANUAL**

```
retailDB> ... .. [
{
  _id: ObjectId('679c70f818e5e77f676b128c'),
  reviews: [ { user: 'Alice', rating: 5, comment: 'Excellent!' } ]
}
]

retailDB> ... .. [
{
  _id: ObjectId('679c70f818e5e77f676b128c'),
  reviews: [ { user: 'Alice', rating: 5, comment: 'Excellent!' } ]
}
]

retailDB> ... .. [
{
  _id: ObjectId('679c70f818e5e77f676b128d'),
  name: 'Smartphone',
  brand: 'BrandB',
  features: [
    { name: 'Processor', value: 'Snapdragon 888' },
    { name: 'RAM', value: '8GB' },
    { name: 'Storage', value: '256GB' }
  ],
  reviews: [ { user: 'Dave', rating: 4, comment: 'Good phone' } ]
}
]

retailDB> ... .. [
{
  _id: ObjectId('679c70f818e5e77f676b128c'),
  name: 'Laptop',
  features: [
    { name: 'Processor', value: 'Intel i7' },
    { name: 'RAM', value: '16GB' }
  ],
  reviews: [ { user: 'Alice', rating: 5, comment: 'Excellent!' } ]
}
]
```

**12. RESULTS & CONCLUSIONS:**

- 

**13. LEARNING OUTCOMES :**

- 

**14. APPLICATION AREAS:**

- 

**15. REMARKS:****1. EXPERIMENT NO:5**

## COURSE LABORATORY MANUAL

2. TITLE: Execute Aggregation operations (\$avg, \$min, \$max, \$push, \$addToSet etc.). students encourage to execute several queries to demonstrate various aggregation operators)

### 3. LEARNING OBJECTIVES:

- Learn the **purpose and benefits** of the aggregation framework.
- Understand how **pipeline stages** work in MongoDB aggregation.
- Use Statistical Aggregation Operators such as \$avg, \$min, \$max, \$push, \$addToSet etc.

### 4. AIM:

- To **understand and execute aggregation operations in MongoDB** using various operators such as \$avg, \$min, \$max, \$push, and \$addToSet.

### 5. MATERIAL / EQUIPMENT REQUIRED:

- 

### 6. THEORY / HYPOTHESIS:

### 7. FORMULA / CALCULATIONS:

- 

### 8. PROCEDURE / PROGRAMME / ACTIVITY:

//create the Sales collection and insert sample documents.

use salesDB

```
db.Sales.insertMany([
  { date: new Date("2024-01-01"), product: "Laptop", price: 1200, quantity: 1, customer: "Amar" },
  { date: new Date("2024-01-02"), product: "Laptop", price: 1200, quantity: 2, customer: "Babu" },
  { date: new Date("2024-01-03"), product: "Mouse", price: 25, quantity: 5, customer: "Chandra" },
  { date: new Date("2024-01-04"), product: "Keyboard", price: 45, quantity: 3, customer: "Amar" },
  { date: new Date("2024-01-05"), product: "Monitor", price: 300, quantity: 1, customer: "Babu" },
  { date: new Date("2024-01-06"), product: "Laptop", price: 1200, quantity: 1, customer: "Deva" }
])
```

//Calculate the average price of each product.

```
db.Sales.aggregate([
  {
    $group: {
      _id: "$product",
      averagePrice: { $avg: "$price" }
    }
  }
]).pretty()
```

//Find the minimum price of each product.

```
db.Sales.aggregate([
  {
```

## **COURSE LABORATORY MANUAL**

```
$group: {
  _id: "$product",
  minPrice: { $min: "$price" }
}
}
]).pretty()

//Find the maximum price of each product.

db.Sales.aggregate([
{
  $group: {
    _id: "$product",
    maxPrice: { $max: "$price" }
  }
}
]).pretty()

//Group sales by customer and push each purchased product into an array.

db.Sales.aggregate([
{
  $group: {
    _id: "$customer",
    products: { $push: "$product" }
  }
}
]).pretty()

//Group sales by customer and add each unique purchased product to an array.

db.Sales.aggregate([
{
  $group: {
    _id: "$customer",
    uniqueProducts: { $addToSet: "$product" }
  }
}
]).pretty()

//Combining Aggregation Operations

db.Sales.aggregate([
{
  $group: {
    _id: "$product",
    totalQuantity: { $sum: "$quantity" },
    totalSales: { $sum: { $multiply: ["$price", "$quantity"] } },
    customers: { $addToSet: "$customer" }
  }
}
]).pretty()
```

## COURSE LABORATORY MANUAL

```
}
}
]).pretty()
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

•

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

11. GRAPHS / OUTPUTS:

• mycompiler\_mongodb> switched to db salesDB

```
salesDB> ... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c75d4589aca9a456b128c'),
    '1': ObjectId('679c75d4589aca9a456b128d'),
    '2': ObjectId('679c75d4589aca9a456b128e'),
    '3': ObjectId('679c75d4589aca9a456b128f'),
    '4': ObjectId('679c75d4589aca9a456b1290'),
    '5': ObjectId('679c75d4589aca9a456b1291')
  }
}
```

```
salesDB> ... .. [
  { _id: 'Laptop', averagePrice: 1200 },
  { _id: 'Monitor', averagePrice: 300 },
  { _id: 'Keyboard', averagePrice: 45 },
  { _id: 'Mouse', averagePrice: 25 }
]
```

```
salesDB> ... .. [
  { _id: 'Laptop', minPrice: 1200 },
  { _id: 'Monitor', minPrice: 300 },
  { _id: 'Keyboard', minPrice: 45 },
  { _id: 'Mouse', minPrice: 25 }
]
```

```
salesDB> ... .. [
  { _id: 'Keyboard', maxPrice: 45 },
  { _id: 'Monitor', maxPrice: 300 },
  { _id: 'Laptop', maxPrice: 1200 },
  { _id: 'Mouse', maxPrice: 25 }
]
```

```
salesDB> ... .. [
  { _id: 'Amar', products: [ 'Laptop', 'Keyboard' ] },
  { _id: 'Deva', products: [ 'Laptop' ] },
  { _id: 'Babu', products: [ 'Laptop', 'Monitor' ] },
  { _id: 'Chandra', products: [ 'Mouse' ] }
]
```

```
salesDB> ... .. [
  { _id: 'Babu', uniqueProducts: [ 'Laptop', 'Monitor' ] },
  { _id: 'Amar', uniqueProducts: [ 'Laptop', 'Keyboard' ] },
  { _id: 'Deva', uniqueProducts: [ 'Laptop' ] },
]
```

**COURSE LABORATORY MANUAL**

```
{ _id: 'Chandra', uniqueProducts: [ 'Mouse' ] }  
]
```

```
salesDB> ... .. [
```

```
{  
  _id: 'Laptop',  
  totalQuantity: 4,  
  totalSales: 4800,  
  customers: [ 'Babu', 'Amar', 'Deva' ]  
},  
{  
  _id: 'Monitor',  
  totalQuantity: 1,  
  totalSales: 300,  
  customers: [ 'Babu' ]  
},  
{  
  _id: 'Keyboard',  
  totalQuantity: 3,  
  totalSales: 135,  
  customers: [ 'Amar' ]  
},  
{  
  _id: 'Mouse',  
  totalQuantity: 5,  
  totalSales: 125,  
  customers: [ 'Chandra' ]  
}  
]
```

12. RESULTS &amp; CONCLUSIONS:

13. LEARNING OUTCOMES :

14. APPLICATION AREAS:

15. REMARKS:

1. EXPERIMENT NO:6

2. TITLE: Execute Aggregation Pipeline and its operations (pipeline must contain \$match, \$group, \$sort, \$project, \$skip etc. students encourage to execute several queries to demonstrate various aggregation operators)

3. LEARNING OBJECTIVES:

- Understand Aggregation Pipelines such as \$match, \$group, \$sort, \$project, \$skip etc.
- 

4. AIM:

- To equip students with the skills to execute and understand MongoDB aggregation pipelines that combine various stages such as \$match, \$group, \$sort, \$project, and \$skip.

5. MATERIAL / EQUIPMENT REQUIRED:

## COURSE LABORATORY MANUAL

•

### 6. THEORY / HYPOTHESIS:

- In MongoDB, **aggregation** is the process of transforming data from a collection into a form that is more useful for analysis or reporting.
- -

### 7. FORMULA / CALCULATIONS:

•

### 8. PROCEDURE / PROGRAMME / ACTIVITY:

```
// Switch to the restaurantDB database
use restaurantDB
```

```
// Insert sample documents into the restaurants collection
```

```
db.restaurants.insertMany([
{
  name: "Biryani House",
  cuisine: "Indian",
  location: "Jayanagar",
  reviews: [
    { user: "Aarav", rating: 5, comment: "Amazing biryani!" },
    { user: "Bhavana", rating: 4, comment: "Great place!" }
  ]
},
{
  name: "Burger Joint",
  cuisine: "American",
  location: "Koramangala",
  reviews: [
    { user: "Chirag", rating: 3, comment: "Average burger" },
    { user: "Devika", rating: 4, comment: "Good value" }
  ]
},
{
  name: "Pasta House",
  cuisine: "Italian",
  location: "Rajajinagar",
  reviews: [
    { user: "Esha", rating: 5, comment: "Delicious pasta!" },
    { user: "Farhan", rating: 4, comment: "Nice ambiance" }
  ]
},
{
  name: "Curry Palace",
  cuisine: "Indian",
  location: "Jayanagar",
  reviews: [
    { user: "Gaurav", rating: 4, comment: "Spicy and tasty!" },
    { user: "Harini", rating: 5, comment: "Best curry in town!" }
  ]
}
])
```



**COURSE LABORATORY MANUAL**

```
},
{
  name: "Taco Stand",
  cuisine: "Mexican",
  location: "Jayanagar",
  reviews: [
    { user: "Ishaan", rating: 5, comment: "Fantastic tacos!" },
    { user: "Jaya", rating: 4, comment: "Very authentic" }
  ]
}
])

// Run the aggregation pipeline query to display reviews summary
db.restaurants.aggregate([
{
  $match: {
    location: "Jayanagar"
  }
},
{
  $unwind: "$reviews"
},
{
  $group: {
    _id: "$name",
    averageRating: { $avg: "$reviews.rating" },
    totalReviews: { $sum: 1 }
  }
},
{
  $sort: {
    averageRating: -1
  }
},
{
  $project: {
    _id: 0,
    restaurant: "$_id",
    averageRating: 1,
    totalReviews: 1
  }
},
{
  $skip: 1
}
]).pretty()
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

## COURSE LABORATORY MANUAL

### 10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

### 11. GRAPHS / OUTPUTS:

- restaurantDB> ..... {

acknowledged: true,

insertedIds: {

'0': ObjectId('679c7b0500c1c98f696b128c'),

'1': ObjectId('679c7b0500c1c98f696b128d'),

'2': ObjectId('679c7b0500c1c98f696b128e'),

'3': ObjectId('679c7b0500c1c98f696b128f'),

'4': ObjectId('679c7b0500c1c98f696b1290')

}

}

restaurantDB> ..... [

{ averageRating: 4.5, totalReviews: 2, restaurant: 'Curry Palace' },

{ averageRating: 4.5, totalReviews: 2, restaurant: 'Taco Stand' } }

]

### 12. RESULTS & CONCLUSIONS:

- 

### 13. LEARNING OUTCOMES :

- 

### 14. APPLICATION AREAS:

- 

### 15. REMARKS:

- 

### 1. EXPERIMENT NO:7

2. TITLE: a.Find all listings with listing\_url, name, address, host\_picture\_url in the listings And Reviews collection that have a host with a picture url

b. Using E-commerce collection write a query to display reviews summary

### 3. LEARNING OBJECTIVES:

- Learn how to query specific fields from a collection using MongoDB's find() method.
- Learn how to use **projection operators** to retrieve specific fields, such as listing\_url, name, address, and host\_picture\_url from a collection.
- Learn how to use **aggregation operators** such as \$group, \$avg, \$sum, and \$project to summarize and transform data.

### 4. AIM:

- Understand how to **filter listings based on conditions**. Master **projection** to retrieve specific fields like listing\_url, name, address, and host\_picture\_url from a collection.

### 5. MATERIAL / EQUIPMENT REQUIRED:

## **COURSE LABORATORY MANUAL**

### 6. THEORY / HYPOTHESIS:

•

### 7. FORMULA / CALCULATIONS:

•

### 8. PROCEDURE / PROGRAMME / ACTIVITY:

7A.

```
// SWITCH TO THE VACATIONRENTALS DATABASE
```

```
use vacationRentals
```

```
// Insert sample documents into the listingsAndReviews collection
```

```
db.listingsAndReviews.insertMany([
{
  listing_url: "http://www.example.com/listing/123456",
  name: "Beautiful Apartment",
  address: {
    street: "123 Main Street",
    suburb: "Central",
    city: "Metropolis",
    country: "Wonderland"
  },
  host: {
    name: "Alice",
    picture_url: "http://www.example.com/images/host/host123.jpg"
  }
},
{
  listing_url: "http://www.example.com/listing/654321",
  name: "Cozy Cottage",
  address: {
    street: "456 Another St",
    suburb: "North",
    city: "Smallville",
    country: "Wonderland"
  },
  host: {
    name: "Bob",
    picture_url: ""
  }
},
{
  listing_url: "http://www.example.com/listing/789012",
  name: "Modern Condo",
  address: {
    street: "789 Side Road",
    suburb: "East",
    city: "Gotham",
    country: "Wonderland"
  }
},
])
```

## **COURSE LABORATORY MANUAL**

```
host: {
  name: "Charlie",
  picture_url: "http://www.example.com/images/host/host789.jpg"
}
}
```

// Query to Find Listings with Host Picture URLs

```
db.listingsAndReviews.find(
{
  "host.picture_url": { $exists: true, $ne: "" }
},
{
  listing_url: 1,
  name: 1,
  address: 1,
  "host.picture_url": 1
}
).pretty()
```

7B.

// Switch to the ecommerce database

use ecommerce

// Insert sample documents into the products collection

```
db.products.insertMany([
{
  product_id: 1,
  name: "Laptop",
  category: "Electronics",
  price: 1200,
  reviews: [
    { user: "Alice", rating: 5, comment: "Excellent!" },
    { user: "Bob", rating: 4, comment: "Very good" },
    { user: "Charlie", rating: 3, comment: "Average" }
  ]
},
{
  product_id: 2,
  name: "Smartphone",
  category: "Electronics",
  price: 800,
  reviews: [
    { user: "Dave", rating: 4, comment: "Good phone" },
    { user: "Eve", rating: 2, comment: "Not satisfied" },
    { user: "Frank", rating: 5, comment: "Amazing!" }
  ]
}]
```

## COURSE LABORATORY MANUAL

```

},
{
  product_id: 3,
  name: "Headphones",
  category: "Accessories",
  price: 150,
  reviews: [
    { user: "Grace", rating: 5, comment: "Great sound" },
    { user: "Heidi", rating: 3, comment: "Okay" }
  ]
}
])

```

// Run the aggregation query to display reviews summary

```

db.products.aggregate([
  {
    $unwind: "$reviews"
  },
  {
    $group: {
      _id: "$name",
      totalReviews: { $sum: 1 },
      averageRating: { $avg: "$reviews.rating" },
      comments: { $push: "$reviews.comment" }
    }
  },
  {
    $project: {
      _id: 0,
      product: "$_id",
      totalReviews: 1,
      averageRating: 1,
      comments: 1
    }
  }
]).pretty()

```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

11. GRAPHS / OUTPUTS:

12. 7a. Output

```

vacationRentals> ... .. {
... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c8d7ef39c0662ef6b128c'),
    '1': ObjectId('679c8d7ef39c0662ef6b128d'),

```

## **COURSE LABORATORY MANUAL**

```
'2': ObjectId('679c8d7ef39c0662ef6b128e')
}
}

vacationRentals> ... .. [
{
  _id: ObjectId('679c8d7ef39c0662ef6b128c'),
  listing_url: 'http://www.example.com/listing/123456',
  name: 'Beautiful Apartment',
  address: {
    street: '123 Main Street',
    suburb: 'Central',
    city: 'Metropolis',
    country: 'Wonderland'
  },
  host: { picture_url: 'http://www.example.com/images/host/host123.jpg' }
},
{
  _id: ObjectId('679c8d7ef39c0662ef6b128e'),
  listing_url: 'http://www.example.com/listing/789012',
  name: 'Modern Condo',
  address: {
    street: '789 Side Road',
    suburb: 'East',
    city: 'Gotham',
    country: 'Wonderland'
  },
  host: { picture_url: 'http://www.example.com/images/host/host789.jpg' }
}
]
```

### 7b. output

```
mycompiler_mongodb> switched to db ecommerce
ecommerce>
ecommerce>
ecommerce> ... .. {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679c8ecff728e48ea16b128c'),
    '1': ObjectId('679c8ecff728e48ea16b128d'),
    '2': ObjectId('679c8ecff728e48ea16b128e')
  }
}
ecommerce>
ecommerce>
ecommerce> ... .. [
{
  totalReviews: 3,
```

## COURSE LABORATORY MANUAL

<pre> averageRating: 4, comments: [ 'Excellent!', 'Very good', 'Average' ], product: 'Laptop' }, {   totalReviews: 3,   averageRating: 3.6666666666666665,   comments: [ 'Good phone', 'Not satisfied', 'Amazing!' ],   product: 'Smartphone' }, {   totalReviews: 2,   averageRating: 4,   comments: [ 'Great sound', 'Okay' ],   product: 'Headphones' } ] </pre>
12. RESULTS & CONCLUSIONS:
•
13. LEARNING OUTCOMES :
•
14. APPLICATION AREAS:
•
15. REMARKS:
•

1. EXPERIMENT NO:8
2. TITLE: a. Demonstrate creation of different types of indexes on collection (unique, sparse, compound and multikey indexes) b. Demonstrate optimization of queries using indexes.
3. LEARNING OBJECTIVES: <ul style="list-style-type: none"> <li>Students will be learned with the skills to create various indexes, optimize their queries for performance, and understand the trade-offs involved in indexing.</li> </ul>
4. AIM: <ul style="list-style-type: none"> <li>The aim of this program is to <b>teach students how to optimize MongoDB queries</b> by creating and using different types of indexes effectively.</li> <li>-</li> </ul>
5. MATERIAL / EQUIPMENT REQUIRED: <ul style="list-style-type: none"> <li></li> </ul>
6. THEORY / HYPOTHESIS: <ul style="list-style-type: none"> <li></li> </ul>
7. FORMULA / CALCULATIONS:
8. PROCEDURE / PROGRAMME / ACTIVITY:
8A



## **COURSE LABORATORY MANUAL**

```
// Switch to the restaurantDB database
use restaurantDB

// Insert sample documents into the restaurants collection
db.restaurants.insertMany([
  {
    name: "Biryani House",
    cuisine: "Indian",
    location: "Downtown",
    reviews: [
      { user: "Aarav", rating: 5, comment: "Amazing biryani!" },
      { user: "Bhavana", rating: 4, comment: "Great place!" }
    ],
    contact: { phone: "1234567890", email: "contact@biryanihouse.com" }
  },
  {
    name: "Curry Palace",
    cuisine: "Indian",
    location: "Downtown",
    reviews: [
      { user: "Gaurav", rating: 4, comment: "Spicy and tasty!" },
      { user: "Harini", rating: 5, comment: "Best curry in town!" }
    ],
    contact: { phone: "0987654321", email: "contact@currypalace.com" }
  },
  {
    name: "Taco Stand",
    cuisine: "Mexican",
    location: "Downtown",
    reviews: [
      { user: "Ishaan", rating: 5, comment: "Fantastic tacos!" },
      { user: "Jaya", rating: 4, comment: "Very authentic" }
    ],
    contact: { phone: "1122334455", email: "contact@tacostand.com" }
  }
])

// Create a unique index on the contact.email field
db.restaurants.createIndex({ "contact.email": 1 }, { unique: true })

// Create a sparse index on the location field
db.restaurants.createIndex({ location: 1 }, { sparse: true })

// Create a compound index on the name and location fields
db.restaurants.createIndex({ name: 1, location: 1 })

// Create a multikey index on the reviews field
```

## **COURSE LABORATORY MANUAL**

```
db.restaurants.createIndex({ reviews: 1 })
```

```
// Verify the created indexes
```

```
db.restaurants.getIndexes()
```

8b.

```
db.products.insertOne({ "name": "Laptop", "category": "Electronics", "price": 799, "tags": ["sale",  
"electronics", "laptop"], "stock": 50, "sku": "ABC123" })
```

```
//Using a Compound Index
```

```
db.products.find({ "category": "Electronics", "price": { $lt: 1000 } })
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

11. GRAPHS / OUTPUTS:

8a. output

```
mycompiler_mongoddb> switched to db restaurantDB
```

```
restaurantDB> ... .. {  
  acknowledged: true,  
  insertedIds: {  
    '0': ObjectId('679c941c63262d5ccd6b128c'),  
    '1': ObjectId('679c941c63262d5ccd6b128d'),  
    '2': ObjectId('679c941c63262d5ccd6b128e')  
  }  
}
```

```
restaurantDB> contact.email_1
```

```
restaurantDB> location_1
```

```
restaurantDB> name_1_location_1
```

```
restaurantDB> reviews_1
```

```
restaurantDB> [  
  { v: 2, key: { _id: 1 }, name: '_id_' },  
  {  
    v: 2,  
    key: { 'contact.email': 1 },  
    name: 'contact.email_1',  
    unique: true  
  },  
  { v: 2, key: { location: 1 }, name: 'location_1', sparse: true },
```

## COURSE LABORATORY MANUAL

<pre>{ v: 2, key: { name: 1, location: 1 }, name: 'name_1_location_1' }, { v: 2, key: { reviews: 1 }, name: 'reviews_1' } ]</pre>
12. RESULTS & CONCLUSIONS:
13. LEARNING OUTCOMES :
14. APPLICATION AREAS:
15. REMARKS:

1. EXPERIMENT NO: 9
2. TITLE: a. Develop a query to demonstrate Text search using catalog data collection for a given word b. Develop queries to illustrate excluding documents with certain words and phrases
3. LEARNING OBJECTIVES: <ul style="list-style-type: none"> <li>Learn how <b>text search</b> works in MongoDB, particularly for searching textual content within a collection.</li> <li>Learn to develop <b>queries for searching</b> a given word in a collection that has a text index.</li> <li>Learn how to <b>exclude documents</b> containing certain words or phrases from search results.</li> </ul>
4. AIM: <ul style="list-style-type: none"> <li>Aim of this program is to perform <b>text search operations</b> and exclude unwanted results, making them proficient in querying MongoDB collections with text-heavy data like catalog listings, product descriptions, and more.</li> </ul>
5. MATERIAL / EQUIPMENT REQUIRED:
6. THEORY / HYPOTHESIS:
7. FORMULA / CALCULATIONS:
8. PROCEDURE / PROGRAMME / ACTIVITY: <p>9A.</p> <pre>db.recipes.insertMany([ {"name": "Cafecito", "description": "A sweet and rich Cuban hot coffee made by topping an espresso shot with a thick sugar cream foam."}, {"name": "New Orleans Coffee", "description": "Cafe Noir from New Orleans is a spiced, nutty coffee made with chicory."}, {"name": "Affogato", "description": "An Italian sweet dessert coffee made with fresh-brewed espresso and vanilla ice cream."}, {"name": "Maple Latte", "description": "A wintertime classic made with espresso and steamed</pre>

**COURSE LABORATORY MANUAL**

milk and sweetened with some maple syrup."},

```
{ "name": "Pumpkin Spice Latte", "description": "It wouldn't be autumn without pumpkin spice lattes made with espresso, steamed milk, cinnamon spices, and pumpkin puree." }
```

```
}}
```

**Create a Text Index**

```
db.recipes.createIndex({ "name": "text", "description": "text" });
```

To search for a given word (e.g., "spiced"), use the \$text operator:

```
db.recipes.find({ $text: { $search: "spiced" } });
```

```
db.recipes.find({ $text: { $search: "spiced espresso" } });
```

9b.

```
db.recipes.find({ $text: { $search: "espresso -milk" } });
```

You can also exclude full phrases

```
db.recipes.find({ $text: { $search: "espresso -\"ice cream\"" } });
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

11. GRAPHS / OUTPUTS:

9a. Output

```
db.recipes.find({ $text: { $search: "spiced" } });
```

```
{ "_id" : ObjectId("61895d2787f246b334ece915"), "name" : "Pumpkin Spice Latte", "description" : "It wouldn't be autumn without pumpkin spice lattes made with espresso, steamed milk, cinnamon spices, and pumpkin puree." }
```

```
{ "_id" : ObjectId("61895d2787f246b334ece912"), "name" : "New Orleans Coffee", "description" : "Cafe Noir from New Orleans is a spiced, nutty coffee made with chicory." }
```

```
db.recipes.find({ $text: { $search: "spiced espresso" } });
```

```
{ "_id" : ObjectId("61895d2787f246b334ece914"), "name" : "Maple Latte", "description" : "A wintertime classic made with espresso and steamed milk and sweetened with some maple syrup." }
```

```
{ "_id" : ObjectId("61895d2787f246b334ece913"), "name" : "Affogato", "description" : "An Italian sweet dessert coffee made with fresh-brewed espresso and vanilla ice cream." }
```

```
{ "_id" : ObjectId("61895d2787f246b334ece911"), "name" : "Cafecito", "description" : "A sweet and rich Cuban hot coffee made by topping an espresso shot with a thick sugar cream foam." }
```

```
{ "_id" : ObjectId("61895d2787f246b334ece915"), "name" : "Pumpkin Spice Latte", "description" : "It wouldn't be autumn without pumpkin spice lattes made with espresso, steamed milk, cinnamon spices, and pumpkin puree." }
```

```
{ "_id" : ObjectId("61895d2787f246b334ece912"), "name" : "New Orleans Coffee", "description" : "Cafe Noir from New Orleans is a spiced, nutty coffee made with chicory." }
```

- 

- -

## COURSE LABORATORY MANUAL

<ul style="list-style-type: none"> <li>9b.Output</li> <li>db.recipes.find({ \$text: { \$search: "espresso -milk" } });</li> </ul> <pre>{ "_id" : ObjectId("61895d2787f246b334ece913"), "name" : "Affogato", "description" : "An Italian sweet dessert coffee made with fresh-brewed espresso and vanilla ice cream." }</pre> <pre>{ "_id" : ObjectId("61895d2787f246b334ece911"), "name" : "Cafecito", "description" : "A sweet and rich Cuban hot coffee made by topping an espresso shot with a thick sugar cream foam." }</pre> <ul style="list-style-type: none"> <li>db.recipes.find({ \$text: { \$search: "espresso -\\ice cream\\\" } });</li> </ul> <pre>{ "_id" : ObjectId("61d48c31a285f8250c8dd5e6"), "name" : "Maple Latte", "description" : "A wintertime classic made with espresso and steamed milk and sweetened with some maple syrup." }</pre> <pre>{ "_id" : ObjectId("61d48c31a285f8250c8dd5e7"), "name" : "Pumpkin Spice Latte", "description" : "It wouldn't be autumn without pumpkin spice lattes made with espresso, steamed milk, cinnamon spices, and pumpkin puree." }</pre> <pre>{ "_id" : ObjectId("61d48c31a285f8250c8dd5e3"), "name" : "Cafecito", "description" : "A sweet and rich Cuban hot coffee made by topping an espresso shot with a thick sugar cream foam." }</pre> <ul style="list-style-type: none"> <li>-</li> </ul>
12. RESULTS & CONCLUSIONS:
13. LEARNING OUTCOMES :
14. APPLICATION AREAS:
15. REMARKS:

1. EXPERIMENT NO:10
2. TITLE:Develop an aggregation pipeline to illustrate Text search on Catalog data collection.
3. LEARNING OBJECTIVES: <ul style="list-style-type: none"> <li>Students will be able to <b>efficiently perform text searches</b> on large catalog datasets, filter search results dynamically, and optimize MongoDB queries for performance.</li> </ul>
4. AIM: <ul style="list-style-type: none"> <li>The aim of this program is to <b>develop an aggregation pipeline that demonstrates text search</b> on a <b>Catalog data collection</b> in MongoDB</li> </ul>
5. MATERIAL / EQUIPMENT REQUIRED:
6. THEORY / HYPOTHESIS:
7. FORMULA / CALCULATIONS:
8. PROCEDURE / PROGRAMME / ACTIVITY: <pre>db.people.insertMany([   { "name": "Nikhil", "pet": "Dog" },   { "name": "Anil", "pet": "Dog" },   { "name": "Rohit", "pet": "Cat" },   { "name": "Vikash", "pet": "Cat" },   { "name": "Suresh", "pet": "Dog" } ])</pre>

## **COURSE LABORATORY MANUAL**

]);

```
db.people.createIndex({name:"text",pet:"text"})
```

```
db.people.aggregate([{$match: {$text: {$search:"Cat"}}},  
  {$group: {_id:null,total: {$sum:1}}}]])
```

```
db.people.aggregate([{$match: {$text: {$search:"Dog"}}},  
  {$group: {_id:null,total: {$sum:1}}}]])
```

9. BLOCK / CIRCUIT / MODEL DIAGRAM / REACTION EQUATION:

- 

10. OBSERVATION TABLE / LOOKUP TABLE / TRUTH TABLE:

- 

11. GRAPHS / OUTPUTS:

- -

--

- -

--

12. RESULTS & CONCLUSIONS:

- 

13. LEARNING OUTCOMES :

- 

14. APPLICATION AREAS:

- 

15. REMARKS:

-