Contents

Experiment 1

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

→
```
use ProgBooksDB

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()

db.dropDatabase()
```

b. Execute the Commands of MongoDB and operations in MongoDB :
Insert, Query, Update, Delete and Projection.

```
→ 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()

  // 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"
})

// Verify to see document is deleted
db.ProgrammingBooks.find({
        title : "JavaScript : The Good Parts"
}).pretty()

db.dropDatabase()
```

Experiment 2

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

```
→ use MoviesDB

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

  // Combining Filter and Projection
  db.Movies.find({
          director : "Christopher Nolan"
  },{
          title : 1,
          year : 1,
          _id : 0
  })

  db.dropDatabase()
```

b. Develop a MongoDB query to display the first 5 documents from the results obtained in **a.**

```
→ use MoviesDB

  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)

db.dropDatabase()
```

Experiment 3

a. Execute query selectors (comparison selectors, logical selectors) and list out the results on any collection

→ `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()

// $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()
db.dropDatabase()
```

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

→ 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 : {
                $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()

db.dropDatabase()

// 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()

// $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()

db.dropDatabase()
```

Experiment 4

4. Create and demonstrate how projection operators (`$`,`$elemMatch` and `$slice`) would be used in the MondoDB.

```
→ use retailDB

  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"
                  }],
                  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()

db.dropDatabase()
```

Experiment 5

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

```
→ 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([{
        $group: {
                _id : "$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()

db.dropDatabase()
```

Experiment 6

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

```
→ 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!"
                }]
        },{
                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()

db.dropDatabase()
```