Web Technology

Assignment - 9

Name: PRANESH SHARMA
Roll No: 22MC3042

1. Connect to a MongoDB server using MongoDB Compass.
2. Create a new database named "testdb" in MongoDB Compass.
3. Create a new collection named "students" in the "testdb" database.
4. Insert ten documents into the "students" collection with the following fields: name, age, and email.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function main() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");
```

```
    // Access a specific database
    const database = client.db('testdb');

    // Access a specific collection within the database
    const collection = database.collection("students");

    // Example: Inserting a document into the collection
    const result = await collection.insertOne({ name: "John", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Jinny", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "jalan", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Jonny", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "James", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Jacob", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Justin", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Jolly", age: 21, email: "123@rgipt.ac.in"
});
    console.log("Inserted document:", result.insertedId);
    const result = await collection.insertOne({ name: "Joy", age: 21, email: "123@rgipt.ac.in" });
```

```
        console.log("Inserted document:", result.insertedId);
        const result = await collection.insertOne({ name: "Jammy", age: 21, email:
"123@rgipt.ac.in" });
        console.log("Inserted document:", result.insertedId);

        // Example: Querying documents from the collection
        const queryResult = await collection.findOne({ name: "Jery" });
        console.log("Query result:", queryResult);
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the main function
main().catch(console.error);
```

5. View the contents of the "students" collection.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function viewStudentsCollection() {
    try {
        // Connect the client to the MongoDB server
        await client.connect();
        console.log("Connected to MongoDB server");
```

```
        // Access the database containing the "students" collection
        const database = client.db('<testdb>');
        const collection = database.collection('students');

        // Find all documents in the "students" collection
        const cursor = collection.find();

        // Iterate over the cursor to access each document
        await cursor.forEach(document => {
            console.log(document);
        });
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the function to view the contents of the "students" collection
viewStudentsCollection().catch(console.error);
```

6. Update the age of a specific student in the "students" collection.

```
const { MongoClient, ObjectId } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);
```

```
async function updateStudentAge(studentId, newAge) {
    try {
        // Connect the client to the MongoDB server
        await client.connect();
        console.log("Connected to MongoDB server");

        // Access the database containing the "students" collection
        const database = client.db('testdb');
        const collection = database.collection('students');

        // Update the age of the student with the specified studentId
        const filter = { _id: ObjectId(studentId) }; // Convert the studentId string to ObjectId
        const updateDoc = {
            $set: {
                age: newAge // Update the age field
            }
        };
        const result = await collection.updateOne(filter, updateDoc);

        // Check if the update was successful
        if (result.modifiedCount === 1) {
            console.log(`Successfully updated age of student with ID ${studentId}`);
        } else {
            console.log(`No student found with ID ${studentId}`);
        }
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the function to update the age of a specific student
```

```
updateStudentAge('James', 25).catch(console.error);
```

7. Delete a document from the "students" collection based on a specific condition.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function deleteStudent(condition) {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collection = database.collection('students');

    // Delete the document that matches the specified condition
    const result = await collection.deleteOne(condition);

    // Check if the deletion was successful
    if (result.deletedCount === 1) {
      console.log("Successfully deleted the document from the 'students' collection");
    } else {
      console.log("No document found matching the specified condition");
```

```
    }
  } finally {
    // Close the client connection
    await client.close();
  }
}

// Call the function to delete a document from the "students" collection based on a specific condition
deleteStudent({ name: "John" }).catch(console.error);
```

8. Use the aggregation pipeline to calculate the average age of all students in the "students" collection.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function calculateAverageAge() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collection = database.collection('students');
```

```
        // Define the aggregation pipeline
        const pipeline = [
            {
                $group: {
                    _id: null, // Group all documents together
                    averageAge: { $avg: "$age" } // Calculate the average age
                }
            }
        ];

        // Execute the aggregation pipeline
        const result = await collection.aggregate(pipeline).toArray();

        // Output the average age
        if (result.length > 0) {
            console.log("Average age of all students:", result[0].averageAge);
        } else {
            console.log("No students found in the collection");
        }
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the function to calculate the average age of all students in the "students" collection
calculateAverageAge().catch(console.error);
```

9. Create an index on the "name" field in the "students" collection.

```
const { MongoClient } = require('mongodb');
```

```
// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function createNameIndex() {
    try {
        // Connect the client to the MongoDB server
        await client.connect();
        console.log("Connected to MongoDB server");

        // Access the database containing the "students" collection
        const database = client.db('testdb');
        const collection = database.collection('students');

        // Create an index on the "name" field
        const result = await collection.createIndex({ name: 1 });

        // Output the index creation result
        console.log("Index created:", result);
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the function to create an index on the "name" field in the "students" collection
createNameIndex().catch(console.error);
```

10. Export the contents of the "students" collection to a JSON file.

```
const { MongoClient } = require('mongodb');
const fs = require('fs');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function exportStudentsToJSON() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collection = database.collection('students');

    // Find all documents in the "students" collection
    const cursor = collection.find();

    // Convert cursor to array of documents
    const documents = await cursor.toArray();

    // Write documents to JSON file
    fs.writeFileSync('students.json', JSON.stringify(documents, null, 2));
    console.log("Exported documents to students.json");
  } finally {
    // Close the client connection
```

```
    await client.close();
  }
}

// Call the function to export the contents of the "students" collection to a JSON file
exportStudentsToJSON().catch(console.error);
```

11. Perform a complex aggregation operation to find the top 5 oldest students in the "students" collection.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function findTopOldestStudents() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collection = database.collection('students');

    // Define the aggregation pipeline
    const pipeline = [
      {
```

```
            $sort: { age: -1 } // Sort documents by age in descending order
        },
        {
            $limit: 5 // Limit the result to 5 documents
        }
    ];

    // Execute the aggregation pipeline
    const result = await collection.aggregate(pipeline).toArray();

    // Output the top 5 oldest students
    console.log("Top 5 oldest students:");
    result.forEach((student, index) => {
        console.log(`${index + 1}. Name: ${student.name}, Age: ${student.age}`);
    });
  } finally {
    // Close the client connection
    await client.close();
  }
}

// Call the function to find the top 5 oldest students in the "students" collection
findTopOldestStudents().catch(console.error);
```

12. Create a geospatial index on a field representing the location of students.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";
```

```
// Create a new MongoClient
const client = new MongoClient(uri);

async function createGeospatialIndex() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collection = database.collection('students');

    // Create a geospatial index on the "location" field
    const result = await collection.createIndex({ location: "2dsphere" });

    // Output the index creation result
    console.log("Geospatial index created:", result);
  } finally {
    // Close the client connection
    await client.close();
  }
}

// Call the function to create a geospatial index on the "location" field in the "students"
collection
createGeospatialIndex().catch(console.error);
```

13. Use MongoDB Compass to visualize the data distribution in the "students" collection.
14. Set up a data validation rule to ensure that documents in the "students" collection must have a non-empty name field.

```
const { MongoClient } = require('mongodb');

// Connection URI
const uri = "mongodb://localhost:27017/";

// Create a new MongoClient
const client = new MongoClient(uri);

async function setUpDataValidationRule() {
  try {
    // Connect the client to the MongoDB server
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database containing the "students" collection
    const database = client.db('testdb');
    const collectionName = 'students';
    const collectionOptions = {
      validator: {
        $jsonSchema: {
          bsonType: "object",
          required: ["name"],
          properties: {
            name: {
              bsonType: "string",
              minLength: 1, // Ensures name field is non-empty
              description: "must be a non-empty string"
```

```
                }
              }
            }
          }
        };

        // Create or update the "students" collection with data validation rule
        await database.createCollection(collectionName, collectionOptions);
        console.log("Data validation rule set up for the 'students' collection");
    } finally {
        // Close the client connection
        await client.close();
    }
}

// Call the function to set up the data validation rule for the "students" collection
setUpDataValidationRule().catch(console.error);
```