BIG DATA MANAGEMENT

Assignment 7

Submitted to

Dr. Dip Shankar Banerjee
Associate Professor
Department of Artificial Intelligen
Engineering
Indian Institute of Technology – J

Submitted By

Purushothaman S
G24AI1042
Post Graduation Diploma and MT
Indian Institute of Technology – J
Date: 12th July 2025

Index

Introduction

This assignment focuses on using MongoDB, a popular NoSQL database, to store and query structured and nested data from TPC-H datasets. You will load customer and order data into MongoDB collections and implement various queries using Java and the MongoDB Java Driver.

What is MongoDB?

MongoDB is a **NoSQL** document-oriented database. Instead of tables and rows (as in relational databases), MongoDB uses **collections and documents**. Documents are stored in JSON-like format (BSON), allowing for flexible and semi-structured data storage.

What is the Use of MongoDB?

MongoDB is used for:

- **Storing large volumes of diverse and unstructured data**
- **High-speed development of applications** where the schema can evolve
- **Real-time analytics** and aggregation
- **Flexible querying** and indexing of hierarchical/nested data

Typical use cases include:

- Content management systems
- Real-time analytics dashboards
- IoT and sensor data storage
- Mobile or web app backends

Advantages of MongoDB

1. **Schema-less Design**: You can store different fields for each document in the same collection.
2. **Horizontal Scalability**: Built-in sharding to scale across multiple machines.
3. **High Performance**: Fast reads and writes using indexing and memory-mapped storage.
4. **Rich Query Language**: Supports ad-hoc queries, aggregation, and geospatial queries.
5. **Document-Oriented Storage**: Stores entire objects in a single document, reducing the need for joins.

How to Connect MongoDB with Your Local Machine

Step 1: Create a MongoDB Cluster

1. Sign in to MongoDB Atlas.
2. Create a **free M0 cluster**.
3. Choose AWS or any preferred cloud provider and region.

Step 2: Configure Access

- **Database Access**:
  - Create a **database user** with a username and password.
- **Network Access**:
  - Add your current **IP address** (or 0.0.0.0/0 to allow from any IP, not recommended for production).

Step 3: Connect Using MongoDB Compass (GUI)

- Download MongoDB Compass.
- Use the **connection string** from Atlas:

```
• mongodb+srv://admin:Suryamyher0@@cluster1.p7pphos.mongodb.net/?retryWri
  tes=true&w=majority&appName=Cluster1
```

Connect Using Java (Programmatic Way)

In the connect() method of your Java program:

java

```
String url =
"mongodb+srv://<username>:<password>@cluster0.mongodb.net/mydb?retryWrites=true
&w=majority";

mongoClient = MongoClients.create(url);

db = mongoClient.getDatabase("mydb");
```

Project Structure

```
MongoDB_Assignment/
|
├── pom.xml                    # (If using Maven for dependencies)
```

```
├── build.gradle              # (If using Gradle instead)
|
├── data/                     # TPC-H data files (customer.tbl, orders.tbl)
|   ├── customer.tbl
|   └── orders.tbl
|
├── screenshots/              # Folder for output screenshots
|   ├── insert_output.png
|   ├── delete_output.png
|   └── query_output.png
|
├── src/
|   └── main/
|       └── java/
|           └── com/
|               └── yourname/
|                   └── mongodbassignment/
|                       ├── MongoDB.java        # Main class with connect(), load(),
loadNest(), etc.
|                       └── QueryUtils.java     # (Optional helper class for common queries)
|
├── README.md                 # Optional documentation
```

Questions and Answers Execution

Script

```
import com.mongodb.client.*;
import com.mongodb.client.model.*;
import org.bson.Document;
import java.io.*;
```

```
import java.util.*;
public class Mongodb_query {
    private MongoClient mongoClient;
    private MongoDatabase db;

    public static void main(String[] args) throws Exception {
        Mongodb_query app = new Mongodb_query();
        app.connect();
        app.load();
        app.loadNest();

        System.out.println(app.query1(1000));
        System.out.println(app.query2(32));
        System.out.println(app.query2Nest(32));
        System.out.println("Total Orders: " + app.query3());
        System.out.println("Total Orders (Nested): " + app.query3Nest());
        System.out.println(toString(app.query4()));
        System.out.println(toString(app.query4Nest()));
        app.close();
    }

    public void connect() {
        String uri =
"mongodb+srv://admin:admin123@cluster1.i1ahzzs.mongodb.net/?retryWrites=tr
ue&w=majority&appName=cluster1";
        try {
            mongoClient = MongoClients.create(uri);
            db = mongoClient.getDatabase("tpch");
            System.out.println("Connected to MongoDB Atlas successfully!");
```

```
    } catch (Exception e) {
        System.err.println("Failed to connect to MongoDB Atlas");
        e.printStackTrace();
    }
}

public void close() {
    if (mongoClient != null) {
        mongoClient.close();
        System.out.println("Connection closed.");
    }
}

public void load() throws Exception {
    MongoCollection<Document> customerCol = db.getCollection("customer");
    MongoCollection<Document> ordersCol = db.getCollection("orders");
    customerCol.drop();
    ordersCol.drop();

    try (BufferedReader reader = new BufferedReader(new
FileReader("data/customer.tbl"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split("\\|");
            Document doc = new Document("c_custkey", Integer.parseInt(parts[0]))
                    .append("c_name", parts[1])
                    .append("c_address", parts[2])
                    .append("c_nationkey", Integer.parseInt(parts[3]))
```

```
                .append("c_phone", parts[4])
                .append("c_acctbal", Double.parseDouble(parts[5]))
                .append("c_mktsegment", parts[6])
                .append("c_comment", parts[7]);
            customerCol.insertOne(doc);
        }
    }

    try (BufferedReader reader = new BufferedReader(new
FileReader("data/order.tbl"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split("\\|");
            Document doc = new Document("o_orderkey", Integer.parseInt(parts[0]))
                .append("o_custkey", Integer.parseInt(parts[1]))
                .append("o_orderstatus", parts[2])
                .append("o_totalprice", Double.parseDouble(parts[3]))
                .append("o_orderdate", parts[4])
                .append("o_orderpriority", parts[5])
                .append("o_clerk", parts[6])
                .append("o_shippriority", Integer.parseInt(parts[7]))
                .append("o_comment", parts[8]);
            ordersCol.insertOne(doc);
        }
    }

    System.out.println("Data loaded into customer and orders collections.");
  }
```

```
public void loadNest() throws Exception {
    MongoCollection<Document> nestedCol = db.getCollection("custorders");
    nestedCol.drop();

    Map<Integer, List<Document>> orderMap = new HashMap<>();

    try (BufferedReader reader = new BufferedReader(new
FileReader("data/order.tbl"))) {
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split("\\|");
            int custKey = Integer.parseInt(parts[1]);
            Document order = new Document("o_orderkey",
Integer.parseInt(parts[0]))
                    .append("o_orderstatus", parts[2])
                    .append("o_totalprice", Double.parseDouble(parts[3]))
                    .append("o_orderdate", parts[4])
                    .append("o_orderpriority", parts[5])
                    .append("o_clerk", parts[6])
                    .append("o_shippriority", Integer.parseInt(parts[7]))
                    .append("o_comment", parts[8]);

            orderMap.computeIfAbsent(custKey, k -> new ArrayList<>()).add(order);
        }
    }

    try (BufferedReader reader = new BufferedReader(new
FileReader("data/customer.tbl"))) {
```

```
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split("\\|");
            int custKey = Integer.parseInt(parts[0]);
            Document customer = new Document("c_custkey", custKey)
                    .append("c_name", parts[1])
                    .append("c_address", parts[2])
                    .append("c_nationkey", Integer.parseInt(parts[3]))
                    .append("c_phone", parts[4])
                    .append("c_acctbal", Double.parseDouble(parts[5]))
                    .append("c_mktsegment", parts[6])
                    .append("c_comment", parts[7])
                    .append("orders", orderMap.getOrDefault(custKey, new
ArrayList<>()));

            nestedCol.insertOne(customer);
        }
    }

    System.out.println("Nested data inserted into 'custorders' collection.");
  }

  public String query1(int custId) {
    Document doc = db.getCollection("customer").find(new
Document("c_custkey", custId)).first();
    return (doc != null) ? "Customer Name: " + doc.getString("c_name") :
"Customer ID " + custId + " not found.";
  }
```

```
public String query2(int orderId) {
    Document doc = db.getCollection("orders").find(new Document("o_orderkey",
orderId)).first();
    return (doc != null) ? "Order Date: " + doc.getString("o_orderdate") : "Order ID
" + orderId + " not found.";
}

public String query2Nest(int orderId) {
    MongoCollection<Document> nestedCol = db.getCollection("custorders");
    try (MongoCursor<Document> cursor = nestedCol.find().iterator()) {
        while (cursor.hasNext()) {
            List<Document> orders = (List<Document>) cursor.next().get("orders");
            for (Document order : orders) {
                if (order.getInteger("o_orderkey") == orderId) {
                    return "Order Date (Nested): " + order.getString("o_orderdate");
                }
            }
        }
    }
    return "Order ID " + orderId + " not found in nested structure.";
}

public long query3() {
    return db.getCollection("orders").countDocuments();
}

public long query3Nest() {
    long total = 0;
    MongoCollection<Document> nestedCol = db.getCollection("custorders");
```

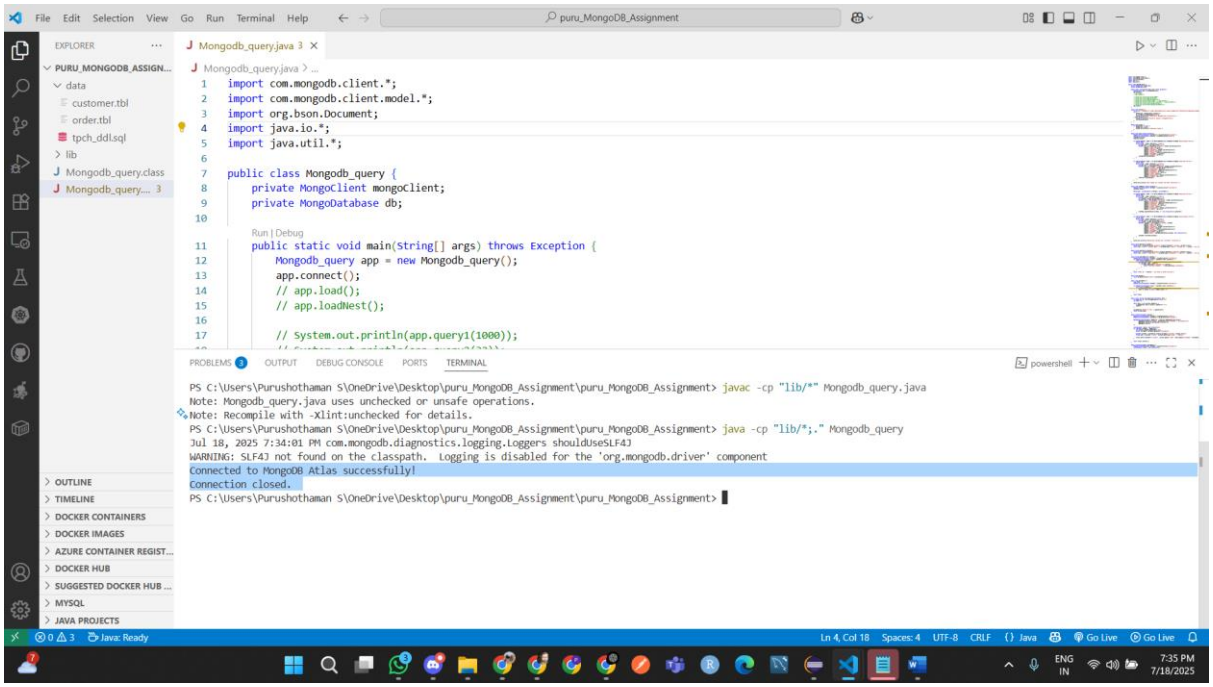
```
    try (MongoCursor<Document> cursor = nestedCol.find().iterator()) {
        while (cursor.hasNext()) {
            List<Document> orders = (List<Document>) cursor.next().get("orders");
            total += (orders != null) ? orders.size() : 0;
        }
    }

    return total;
}

public static String toString(Iterator<Document> docs) {
    StringBuilder sb = new StringBuilder("Rows:\n");
    int count = 0;

    while (docs != null && docs.hasNext()) {
        sb.append(docs.next().toJson()).append("\n");
        count++;
    }

    sb.append("Number of rows: ").append(count);
    return sb.toString();
}

public Iterator<Document> query4() {
    MongoCollection<Document> ordersCol = db.getCollection("orders");
    MongoCollection<Document> customerCol = db.getCollection("customer");
```

```
        AggregateIterable<Document> aggResults = ordersCol.aggregate(Arrays.asList(
            Aggregates.group("$o_custkey", Accumulators.sum("totalSpent",
"$o_totalprice")),
            Aggregates.sort(Sorts.descending("totalSpent")),
            Aggregates.limit(5)
        ));

        List<Document> result = new ArrayList<>();
        for (Document groupDoc : aggResults) {
            int custId = groupDoc.getInteger("_id");
            double totalSpent = groupDoc.getDouble("totalSpent");

            Document customer = customerCol.find(new Document("c_custkey",
custId)).first();
            String name = (customer != null) ? customer.getString("c_name") :
"Unknown";

            result.add(new Document("c_custkey", custId).append("c_name",
name).append("totalSpent", totalSpent));
        }

        return result.iterator();
    }

    public Iterator<Document> query4Nest() {
        MongoCollection<Document> nestedCol = db.getCollection("custorders");
        List<Document> result = new ArrayList<>();

        try (MongoCursor<Document> cursor = nestedCol.find().iterator()) {
```

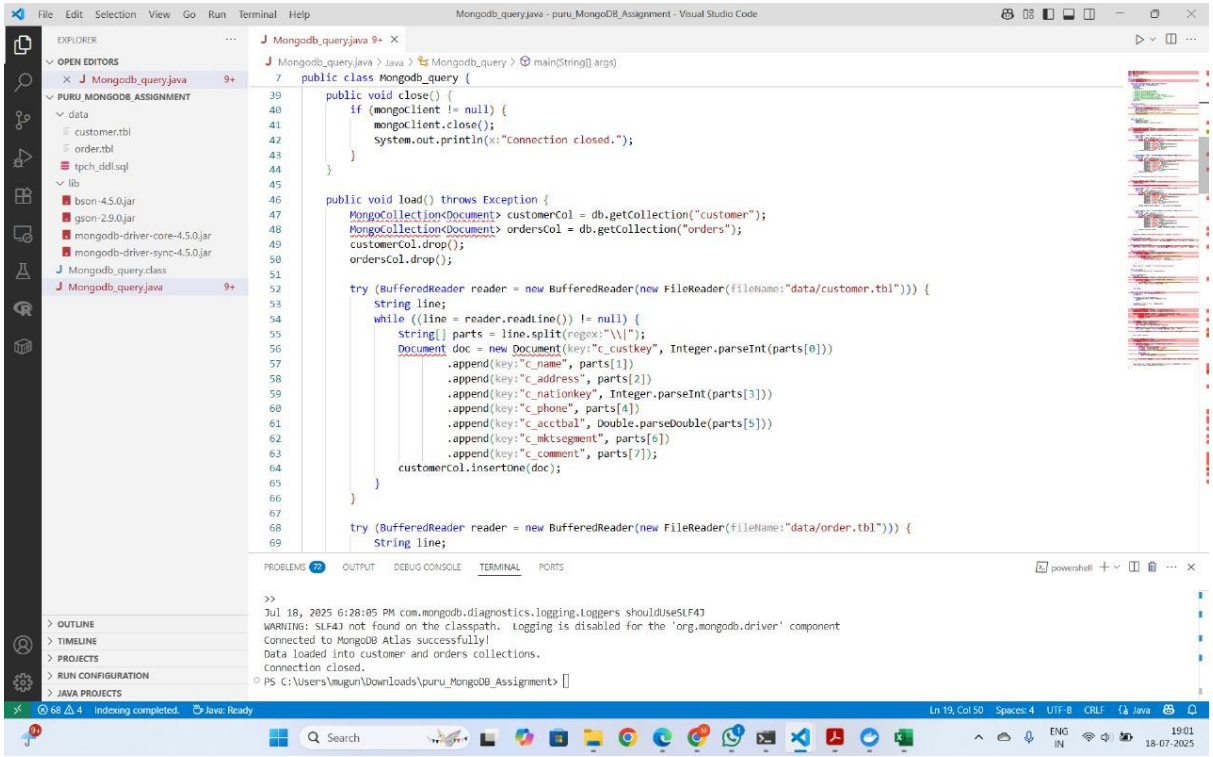
```
        while (cursor.hasNext()) {
            Document customer = cursor.next();
            int custKey = customer.getInteger("c_custkey");
            String name = customer.getString("c_name");
            List<Document> orders = (List<Document>) customer.get("orders");

            double total = 0;
            for (Document order : orders) {
                total += order.getDouble("o_totalprice");
            }

            result.add(new Document("c_custkey", custKey).append("c_name",
name).append("totalSpent", total));
        }
    }

    result.sort((a, b) -> Double.compare(b.getDouble("totalSpent"),
a.getDouble("totalSpent")));
    return result.stream().limit(5).iterator();
  }
}
```

Output

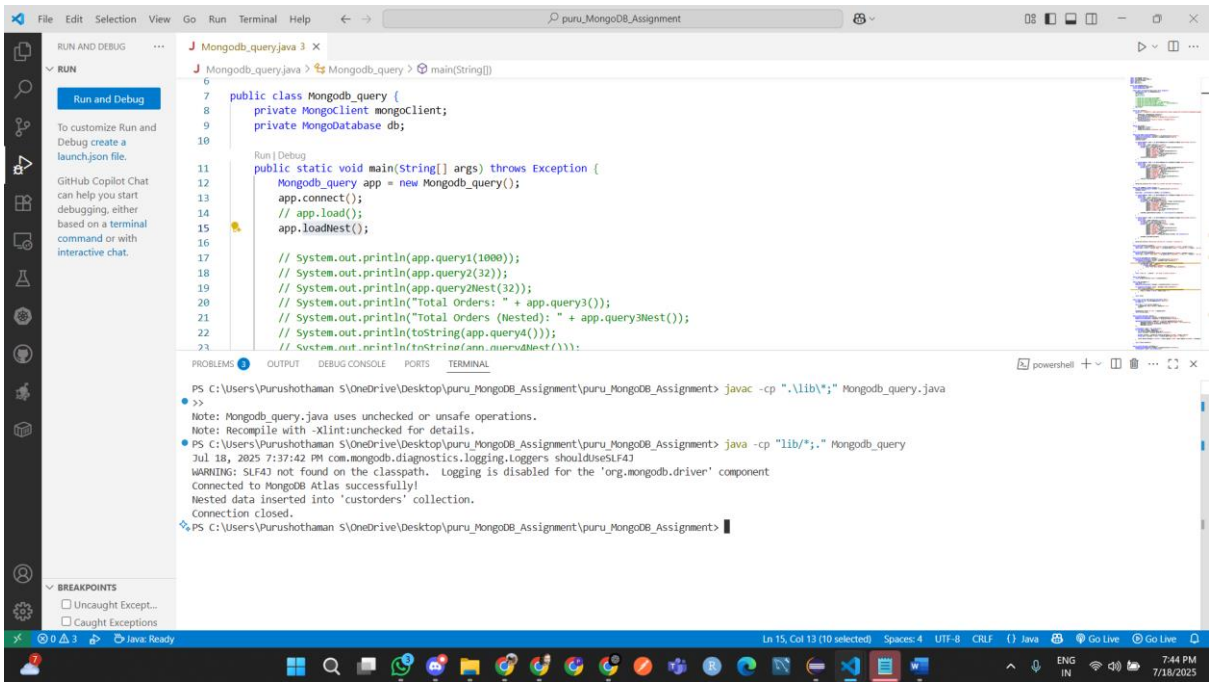
To Connect the MongoDB with local machine



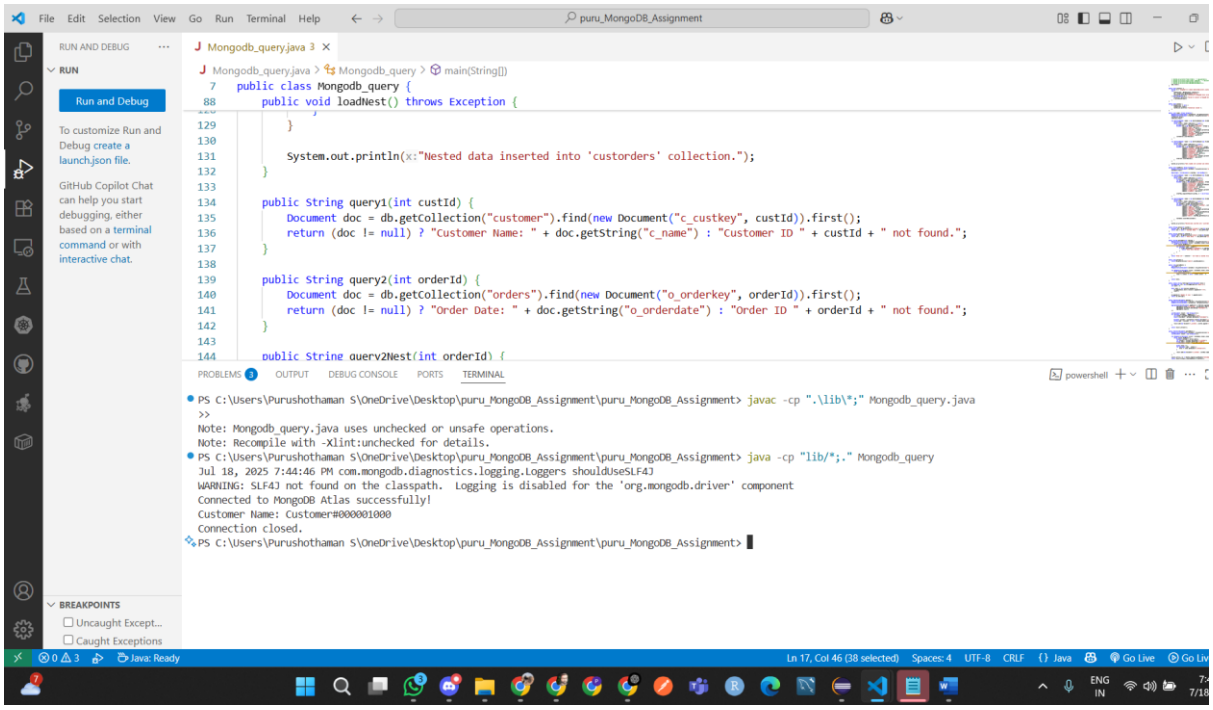
To Load



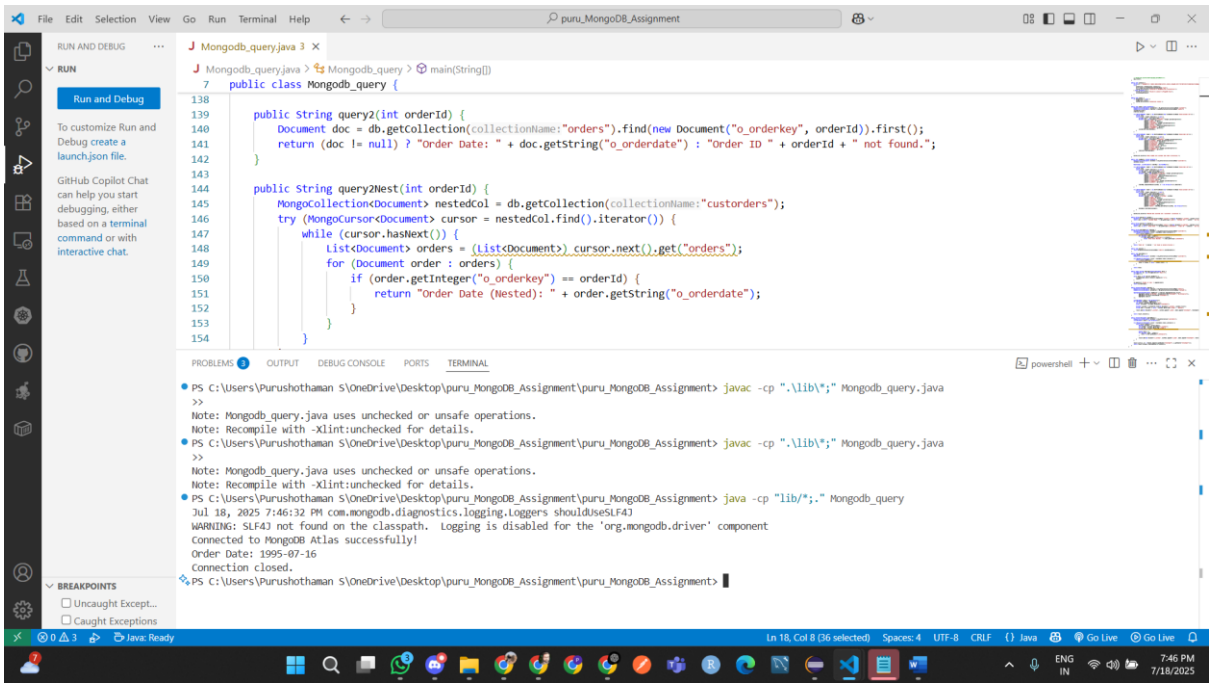
To loadNest()



Query 1



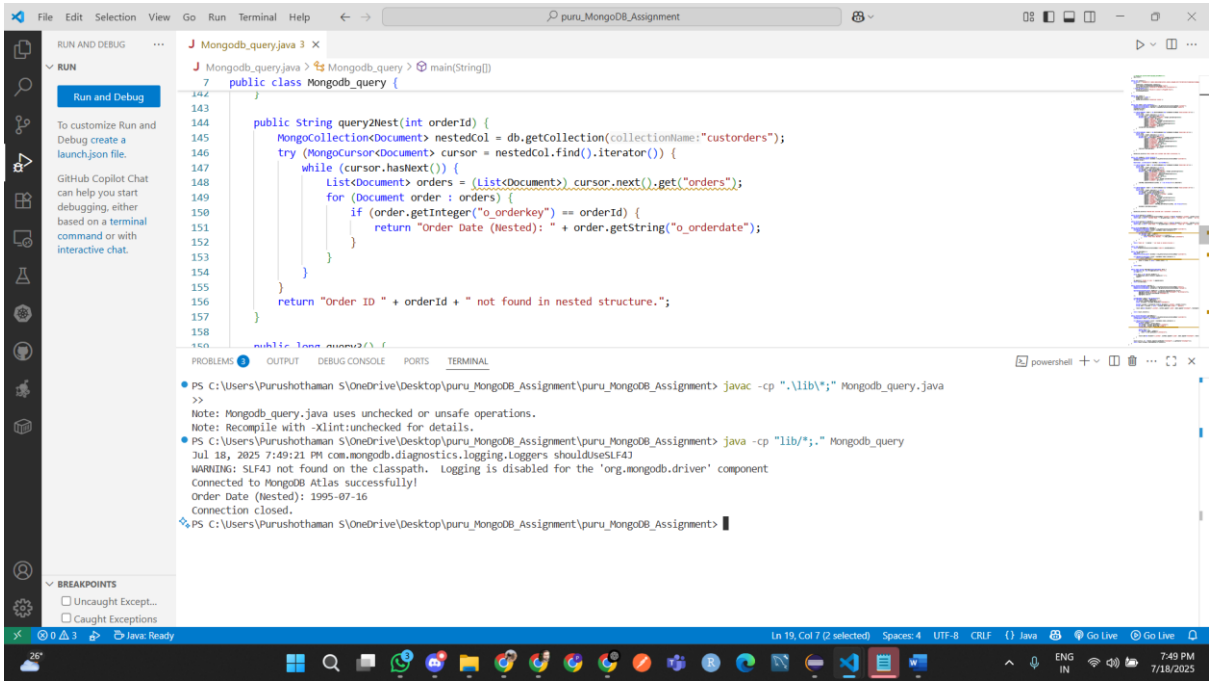
Query2()



```java
public class Mongodb_query {

    public String query2(int orderId) {
        Document doc = db.getCollection("orders").find(new Document("o_orderkey", orderId)).first();
        return (doc != null) ? "Order Date: " + doc.getString("o_orderdate") : "Order ID " + orderId + " not found.";
    }

    public String query2Nest(int orderId) {
        MongoCollection<Document> nestedCol = db.getCollection("custorders");
        try (MongoCursor<Document> cursor = nestedCol.find().iterator()) {
            while (cursor.hasNext()) {
                List<Document> orders = (List<Document>) cursor.next().get("orders");
                for (Document order : orders) {
                    if (order.getInteger("o_orderkey") == orderId) {
                        return "Order Date (Nested): " + order.getString("o_orderdate");
                    }
                }
            }
```

```
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment> javac -cp ".\lib\*;" Mongodb_query.java
>>
Note: Mongodb_query.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment> javac -cp ".\lib\*;" Mongodb_query.java
>>
Note: Mongodb_query.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment> java -cp "lib/*;." Mongodb_query
Jul 18, 2025 7:46:32 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component
Connected to MongoDB Atlas successfully!
Order Date: 1995-07-16
Connection closed.
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment>
```

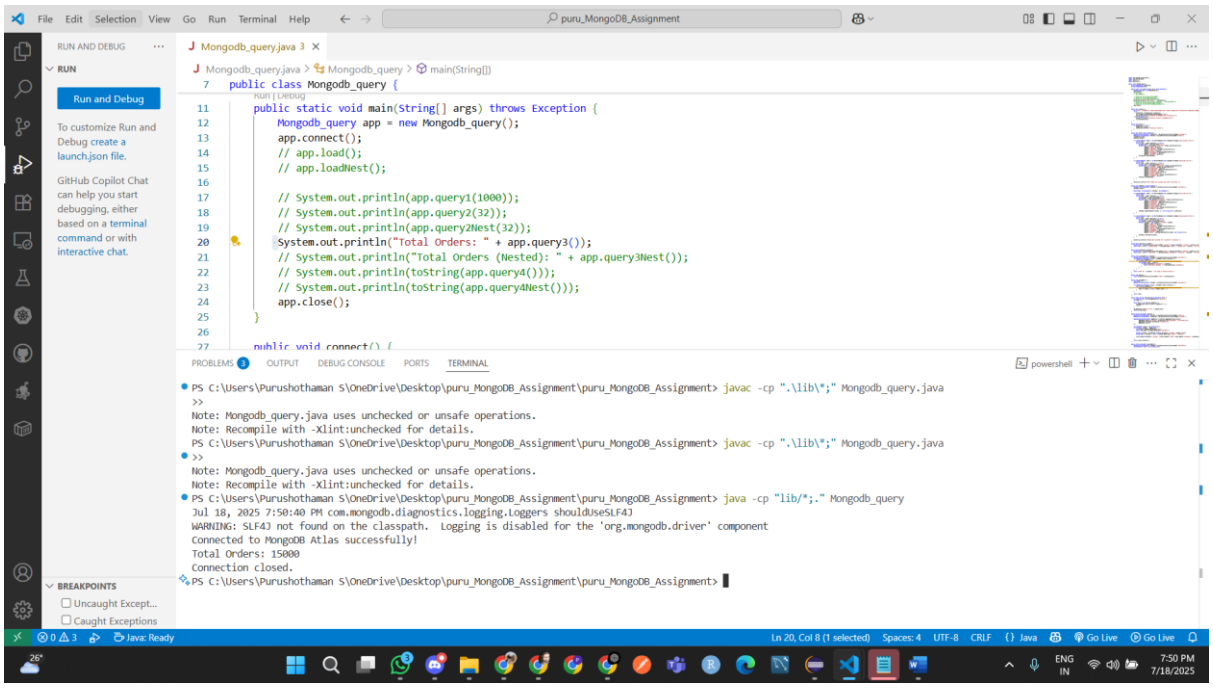
Query 2 Nested()



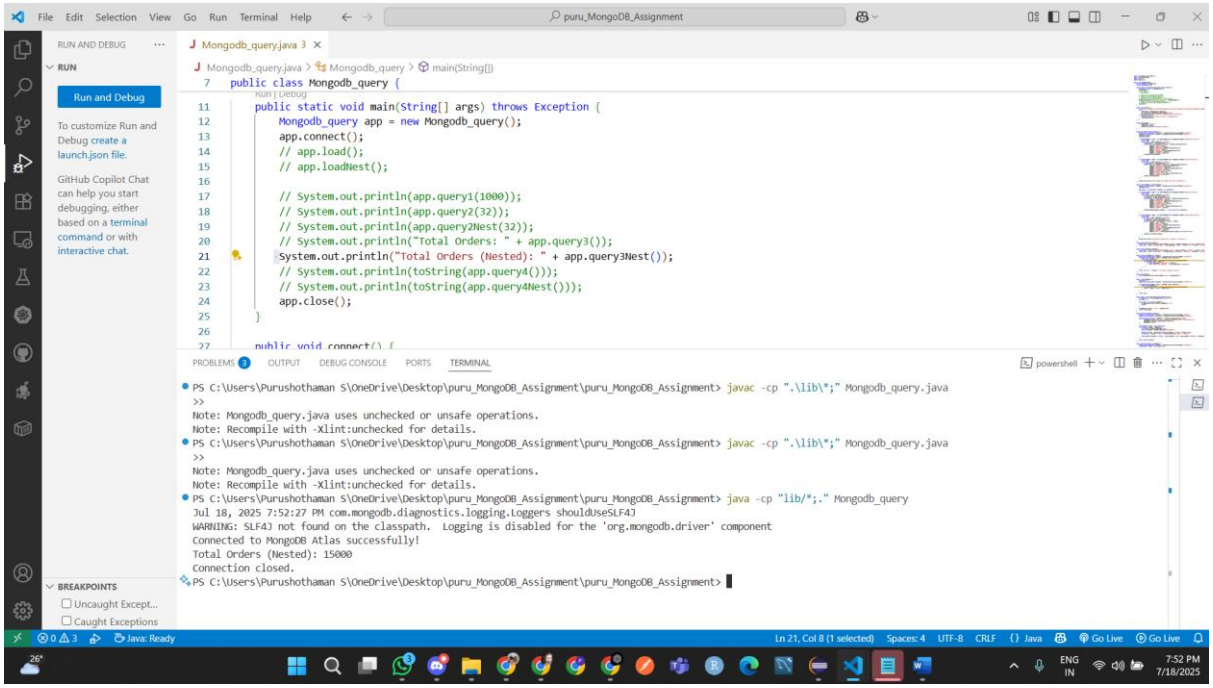
```java
public class Mongodb_query {

    public String query2Nest(int orderId) {
        MongoCollection<Document> nestedCol = db.getCollection("custorders");
        try (MongoCursor<Document> cursor = nestedCol.find().iterator()) {
            while (cursor.hasNext()) {
                List<Document> orders = (List<Document>) cursor.next().get("orders");
                for (Document order : orders) {
                    if (order.getInteger("o_orderkey") == orderId) {
                        return "Order Date (Nested): " + order.getString("o_orderdate");
                    }
                }
            }
        }
        return "Order ID " + orderId + " not found in nested structure.";
    }
```

```
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment> javac -cp ".\lib\*;" Mongodb_query.java
>>
Note: Mongodb_query.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment> java -cp "lib/*;." Mongodb_query
Jul 18, 2025 7:49:21 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'org.mongodb.driver' component
Connected to MongoDB Atlas successfully!
Order Date (Nested): 1995-07-16
Connection closed.
PS C:\Users\Purushothaman S\OneDrive\Desktop\puru_MongoDB_Assignment\puru_MongoDB_Assignment>
```

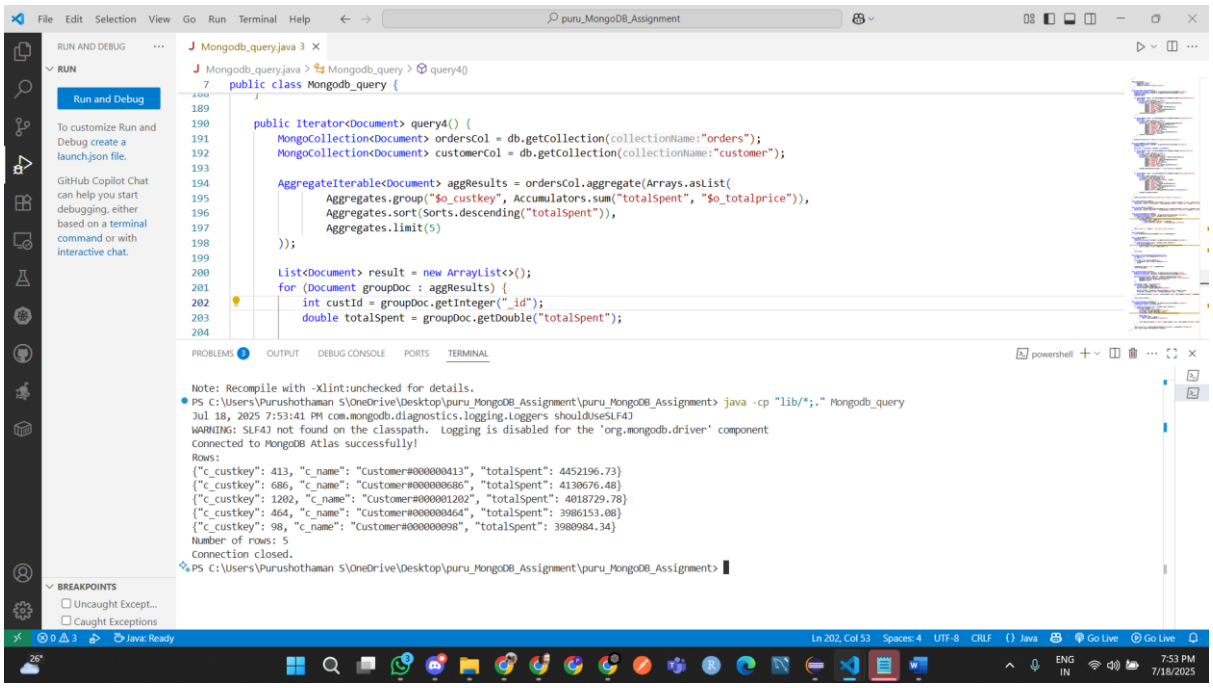
Query 3



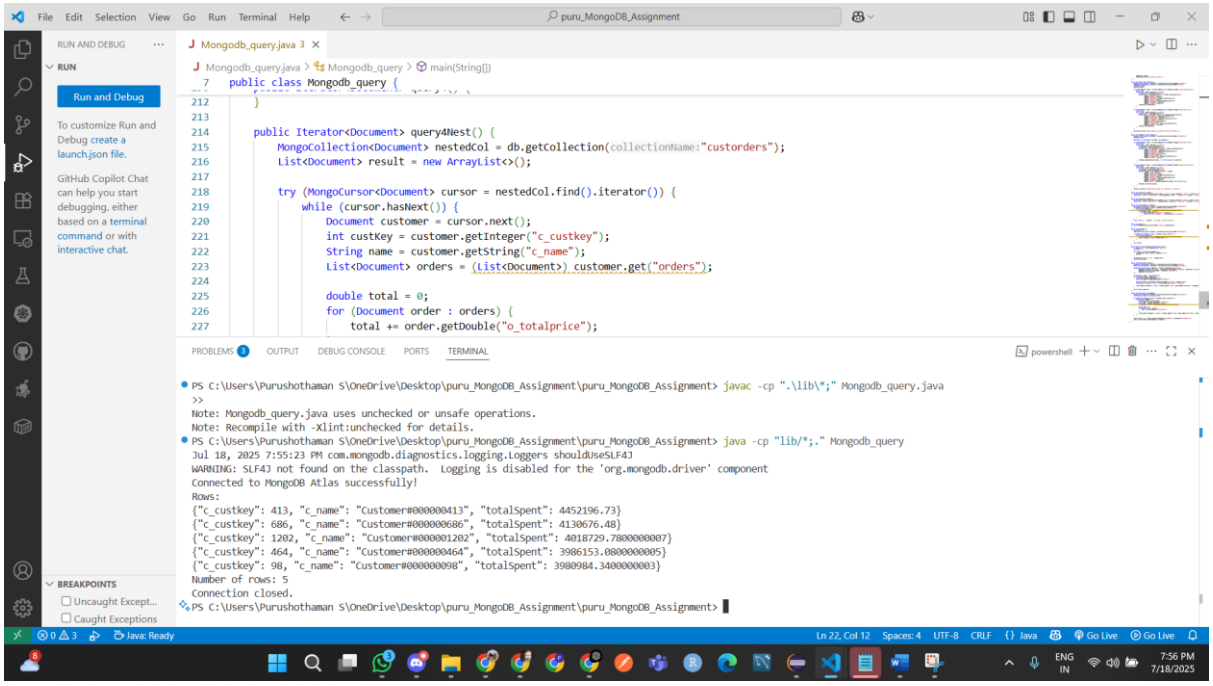
Query 3 Nested



Query 4



Query4 Nested



Conclusion

In this assignment, we explored the fundamental operations of MongoDB using both flat and nested data models to store and query TPC-H customer and order data. By implementing methods to load data into collections, perform queries, and retrieve insights using the MongoDB Java Driver, we gained hands-on experience with NoSQL document-oriented databases.

The assignment highlighted the flexibility of MongoDB in handling both structured and semi-structured data, its ease of integration with Java applications, and its suitability for scalable and high-performance applications. Additionally, by working with nested collections, we learned how MongoDB can represent relational data hierarchies naturally, reducing the need for complex joins.

Overall, this exercise strengthened our understanding of MongoDB's data modelling, CRUD operations, and aggregation capabilities in a real-world context

Git Link: https://github.com/PurushothamanShanmugam/MongoDB_BigData