|  |
| --- |
|  |

**LAB 2: CRUD Operations in MongoDB**

|  |  |
| --- | --- |
| **Track** | Data & AI |
| **Role** | Software Development - Backend |
| **Level** | Level 2 |
| **Course** | NoSQL Databases |
| **Scope** | * Implement CRUD operations on telecom data in MongoDB. * Perform bulk data manipulation and advanced queries. * Apply atomic operations and transactions for data integrity. |
| **Prerequisite** | * Basic understanding of MongoDB and NoSQL databases. * MongoDB Compass and Mongo Shell installed on your system. * Familiarity with JSON data structure. |

**Objective**

Effectively perform CRUD operations on telecom data using MongoDB Compass and Mongo Shell, utilize advanced data manipulation techniques like bulk operations, and ensure data integrity through atomic operations and transactions.

**Tasks**

**Dataset: telecom\_dataset\_v2.json**

1. **Create Operations:**

* Insert the telecom dataset as a JSON file into MongoDB Compass.
* Perform bulk insertion of the dataset into MongoDB using Mongo Shell.

1. **Read Operations:**

* Query customer data in MongoDB Compass based on subscription region.
* Use Mongo Shell to perform advanced queries, such as filtering customers with premium plans and high monthly bills.

1. **Insert Data Using Queries:**

* Update individual customer records in MongoDB Compass (e.g., changing subscription plans).
* Perform bulk updates on customer records using Mongo Shell.

1. **Delete Operations:**

* Delete a specific customer record in MongoDB Compass.
* Perform bulk deletion of customers based on specific criteria using Mongo Shell.

1. **Advanced CRUD Techniques:**

* Perform atomic updates, such as incrementing customer tenure, using Mongo Shell.
* Implement transactions to update customer data and insert billing records simultaneously.

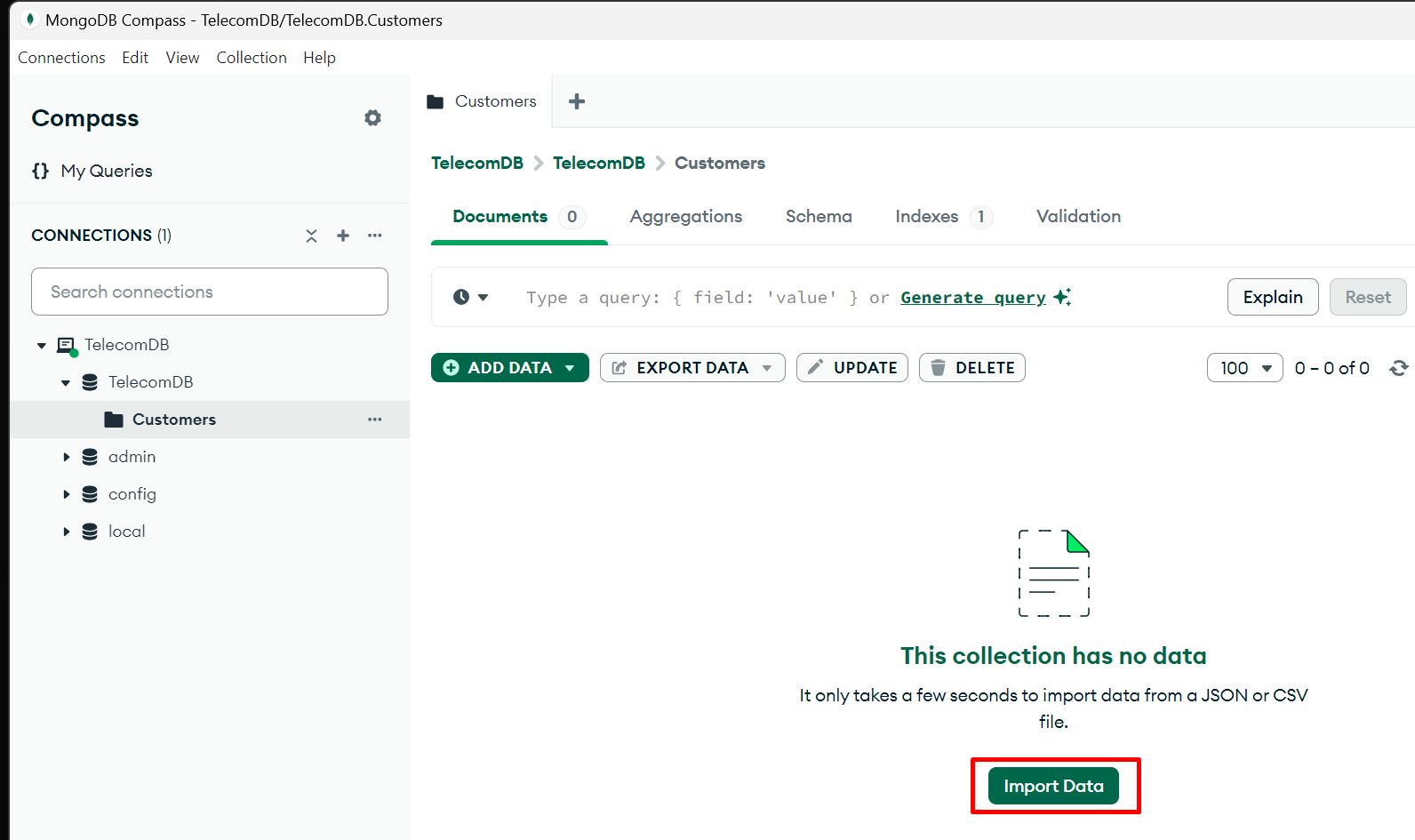
**Outcomes**

* Load and insert telecom data into MongoDB using Compass and Mongo Shell.
* Perform CRUD operations, including bulk inserts and atomic updates.
* Implement transactions to ensure data consistency.
* Gain real-world experience managing telecom data in MongoDB.

**Solution**

##### Create Operations:

* 1. **Insert Dataset as JSON (MongoDB Compass):**
  + Open MongoDB Compass and connect to your local MongoDB server.
  + Navigate to the **TelecomDB** database. If it doesn’t exist, create a new database called **TelecomDB**.
  + Click on the Customers collection and select **Add Data**. (Creaet a new collection if it doesn’t exist).
  + In the dialog box, choose **Import** **JSON or CSV file**.
  + Or you can click on **Import Data** button.



* + Next Select the json file from your local system.
  + And click on **Import** Button.

A screenshot of a computer

Description automatically generated

* + Once data is imported successfully data will be visible in Customers collection.

A screenshot of a computer

Description automatically generated

##### Read Operations:

* 1. **Query Data (MongoDB Compass):**
  + Navigate to the Customers collection in MongoDB Compass.
  + In the query bar, search for customers based on their subscription region. For example, to find all customers in Europe, use the following filter:

{ "Subscription.Region": "Europe" }

* + Click **Find** to view the filtered results.

A screenshot of a computer

Description automatically generated

* 1. **Query Data (MongoDB Compass):**
  + Use Mongo Shell to query customers with a Premium plan and a monthly bill greater than $80:

db.Customers.find({

    "Subscription.Plan": "Premium",

    "Subscription.Monthly\_Bill": { $gt: 80 }

  });

* + View the results directly in Mongo Shell.

A screenshot of a computer program

Description automatically generated

* + Find customers from "North America" who have a tenure of more than 50 months.

db.telecom.find({

    "Subscription.Region": "North America",

    "Subscription.Tenure\_Months": { $gt: 50 }

});

* + Output:



* + Get customers with monthly bills greater than $80 sorted in descending order by their bill

db.telecom.find({ "Subscription.Monthly\_Bill": { $gt: 80 } })

          .sort({ "Subscription.Monthly\_Bill": -1 });

* + Output:



* + Find the first 5 customers who use more than 60GB of data

db.telecom.find({ "Subscription.Data\_Usage\_GB": { $gt: 60 } })

          .limit(5);

* + Output:



##### Update Individual Records (MongoDB Compass):

* 1. **Update Individual Records (MongoDB Compass):**
  + In MongoDB Compass, go to the Customers collection.
  + Search for a specific customer, Customer\_ID "CUST002".

A screenshot of a computer

Description automatically generated

* + Click on the document to open it, then click **Edit**.
  + Modify the Subscription.Plan field from "Standard" to "Premium” and click **Update** to commit the changes.

A screenshot of a computer

Description automatically generated

* + Once updated try to find the Subscription.Plan with "Standard"for Customer\_ID "CUST002" to see any records.

A screenshot of a computer

Description automatically generated

* + You can also update directly from MongoDB Shell.
  + Update the "Plan" of all customers in the "Asia" region with a "Basic" plan to "Standard"

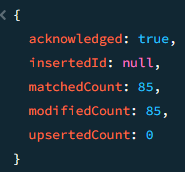
db.telecom.updateMany(

    { "Subscription.Region": "Asia", "Subscription.Plan": "Basic" },

    { $set: { "Subscription.Plan": "Standard" } }

);

* + Output:



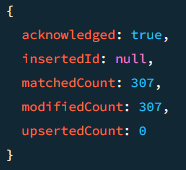
* + Increase the "Monthly\_Bill" by 10% for all customers who have used more than 70GB of data

db.telecom.updateMany(

    { "Subscription.Data\_Usage\_GB": { $gt: 70 } },

    { $mul: { "Subscription.Monthly\_Bill": 1.1 } });

* + Output:



* 1. **Bulk Update (Mongo Shell):**
  + Use Mongo Shell to bulk update the subscription plan for all customers in the Asia region to "Premium":
  + Verify the update by running a query to check the updated records.

db.Customers.updateMany(

  { "Subscription.Region": "Asia" },

  { $set: { "Subscription.Plan": "Premium" } }

);

);

A screenshot of a computer

Description automatically generated

* + Verify the update by running a query to check the updated records.

A screenshot of a computer

Description automatically generated

##### Delete Operations:

* 1. **Bulk Delete (Mongo Shell):**
  + Use Mongo Shell to bulk delete customers with a tenure of fewer than 12 months:

db.Customers.deleteMany({ "Subscription.Tenure\_Months": { $lt: 12 } });

A screen shot of a computer

Description automatically generated

* + Confirm the deletion by querying the collection.

A screenshot of a computer

Description automatically generated

* + Delete all customers with a tenure of less than 12 months

db.telecom.deleteMany({ "Subscription.Tenure\_Months": { $lt: 12 } });

* + Output:



* + Delete a specific customer by their Customer\_ID (e.g., CUST050)

db.telecom.deleteOne({ "Customer\_ID": "CUST050" });

* + Output:



##### Advanced CRUD Techniques

* 1. **Atomic Operations:**
  + Increment the Tenure\_Months field for all customers by 1 month:

db.Customers.updateMany(

  {},

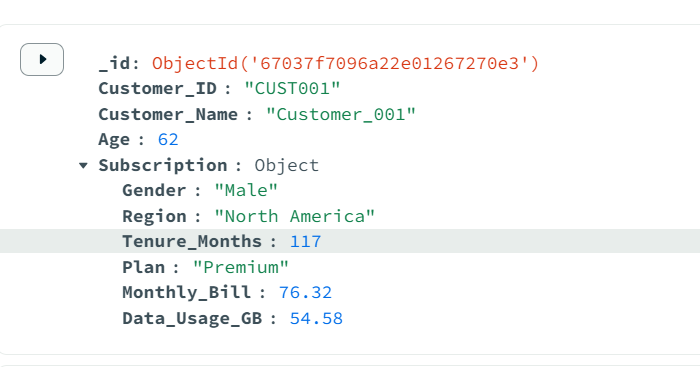
  { $inc: { "Subscription.Tenure\_Months": 1 } }

);

A screenshot of a computer program

Description automatically generated

* + Verify that all customers' Tenure\_Months have been updated successfully.

A screenshot of a computer

Description automatically generated

* 1. **Transactions Operations:**
  + Start a session in Mongo Shell and perform a transaction that updates a customer's billing information and inserts a record into a Billing collection. Both operations will either succeed or fail together.
  + Starts a new session and begins a transaction.
  + Updates the customer's billing information for Customer 001:
    1. Sets the new monthly bill to $80.00.
    2. Updates the plan to "Premium Plus".

// Start a session

const session = db.getMongo().startSession();

// Start a transaction

session.startTransaction();

try {

  // Get the collections

  const customersCollection = session.getDatabase("TelecomDB").Customers;

  const billingCollection = session.getDatabase("TelecomDB").Billing;

  // Update customer's billing information

  const updateResult = customersCollection.updateOne(

    { Customer\_ID: "CUST001" },

    {

      $set: {

        "Subscription.Monthly\_Bill": 80.00,

        "Subscription.Plan": "Premium Plus"

      }

    }

  );

  console.log(`Updated ${updateResult.modifiedCount} customer document(s)`);

* + Inserts a new record into the Billing collection with the updated information.
  + If all operations are successful, it commits the transaction.
  + If any error occurs, it aborts the transaction.
  + Finally, it ends the session.

  // Insert a record into the Billing collection

  const billingInsertResult = billingCollection.insertOne({

    Customer\_ID: "CUST001",

    Billing\_Date: new Date(),

    Amount: 80.00,

    Plan: "Premium Plus"

  });

  console.log(`Inserted billing record with ID: ${billingInsertResult.insertedId}`);

  // Commit the transaction

  session.commitTransaction();

  console.log("Transaction committed successfully");

} catch (error) {

  // If an error occurred, abort the transaction

  session.abortTransaction();

  console.error("Transaction aborted. Error:", error);

} finally {

  // End the session

  session.endSession();

}

* + Verify the transaction was successful by querying both the Customers and Billing collections.

A screenshot of a computer

Description automatically generated