Lesson 01 Demo 08

Deleting Documents Using MongoDB CRUD Operations

Objective: To delete documents from the MongoDB collection using delete operation with try catch blocks

Tools required: Eclipse IDE

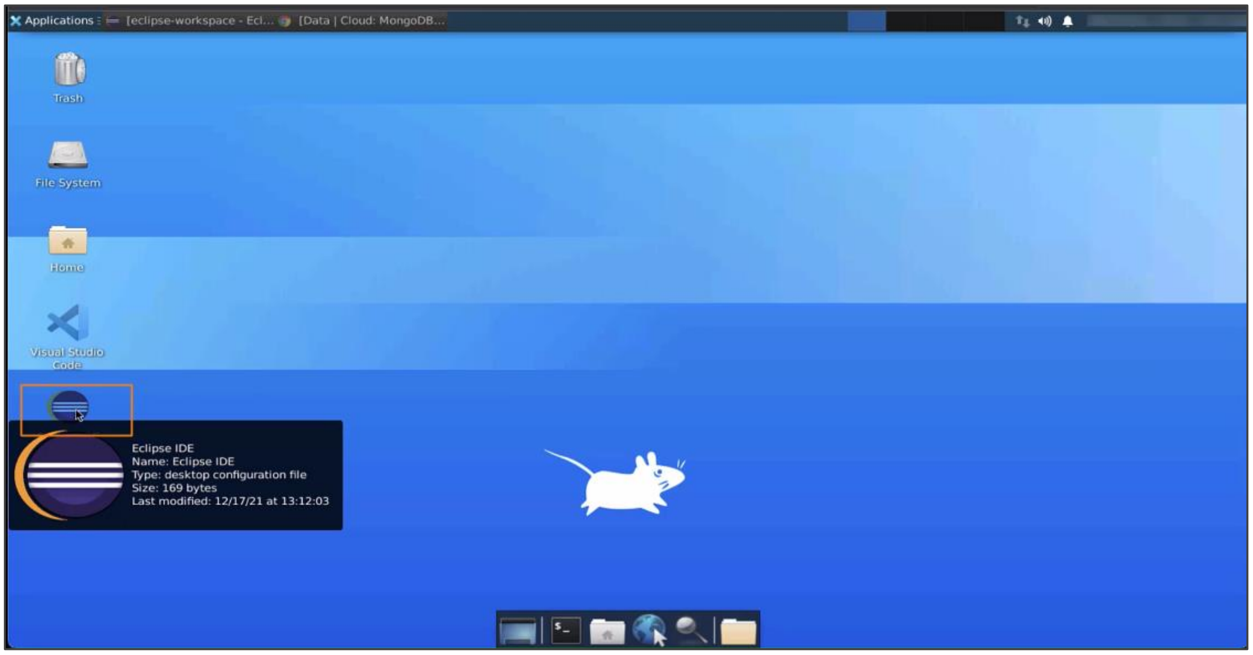
Prerequisites: None

Steps to be followed:

1. Create methods to delete customers
2. Handle exceptions

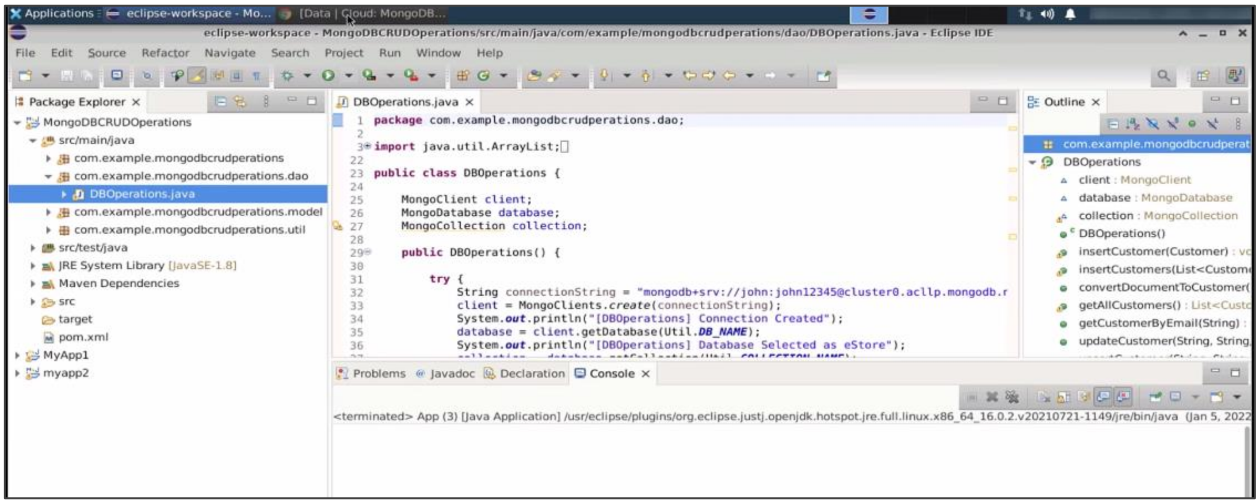
Step 1: Create methods to delete customers

1.1 Open the **Eclipse IDE**



Eclipse IDE
Name: Eclipse IDE
Type: desktop configuration file
Size: 169 bytes
Last modified: 12/17/21 at 13:12:03

1.2 Go the project **MongoDBCRUDOperations** and open the file **DBOperations.java** file

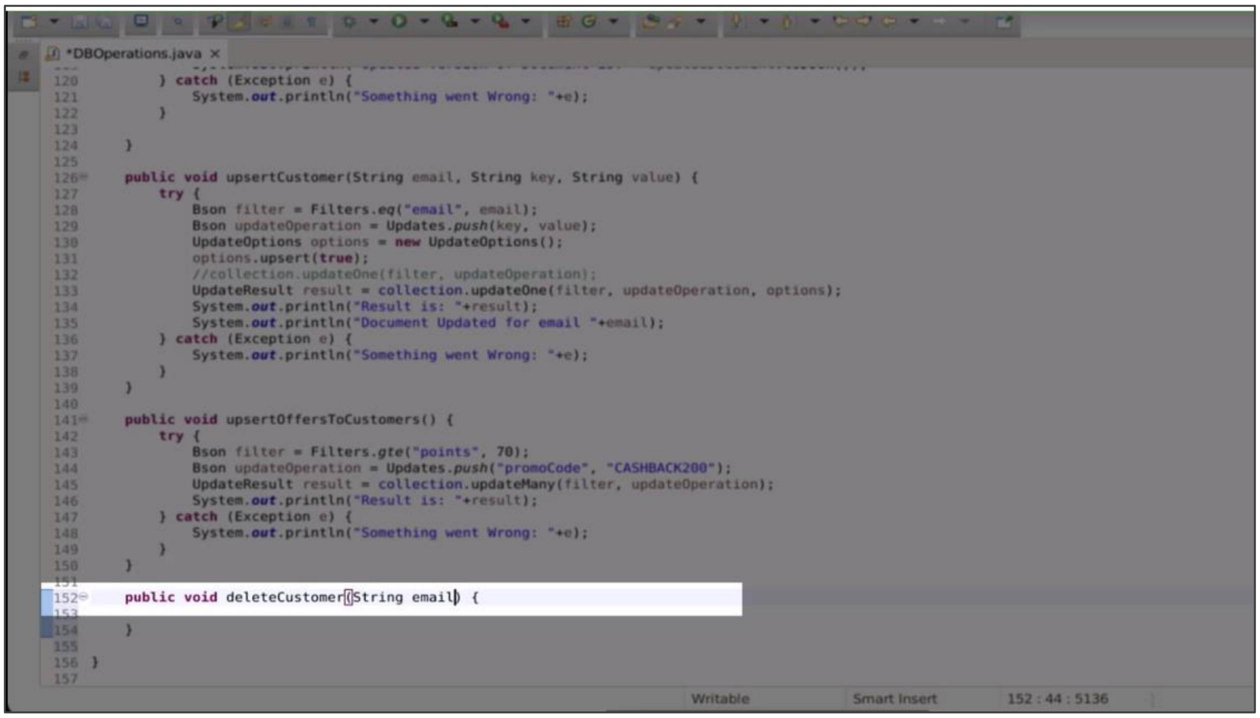


```java
package com.example.mongodbcrudperations.dao;

import java.util.ArrayList;

public class DBOperations {

    MongoClient client;
    MongoDatabase database;
    MongoCollection collection;

    public DBOperations() {

        try {
            String connectionString = "mongodb+srv://john:john12345@cluster0.acllp.mongodb.r
            client = MongoClients.create(connectionString);
            System.out.println("[DBOperations] Connection Created");
            database = client.getDatabase(Util.DB_NAME);
            System.out.println("[DBOperations] Database Selected as eStore");
```

```
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_16.0.2.v20210721-1149/jre/bin/java (Jan 5, 2022
```

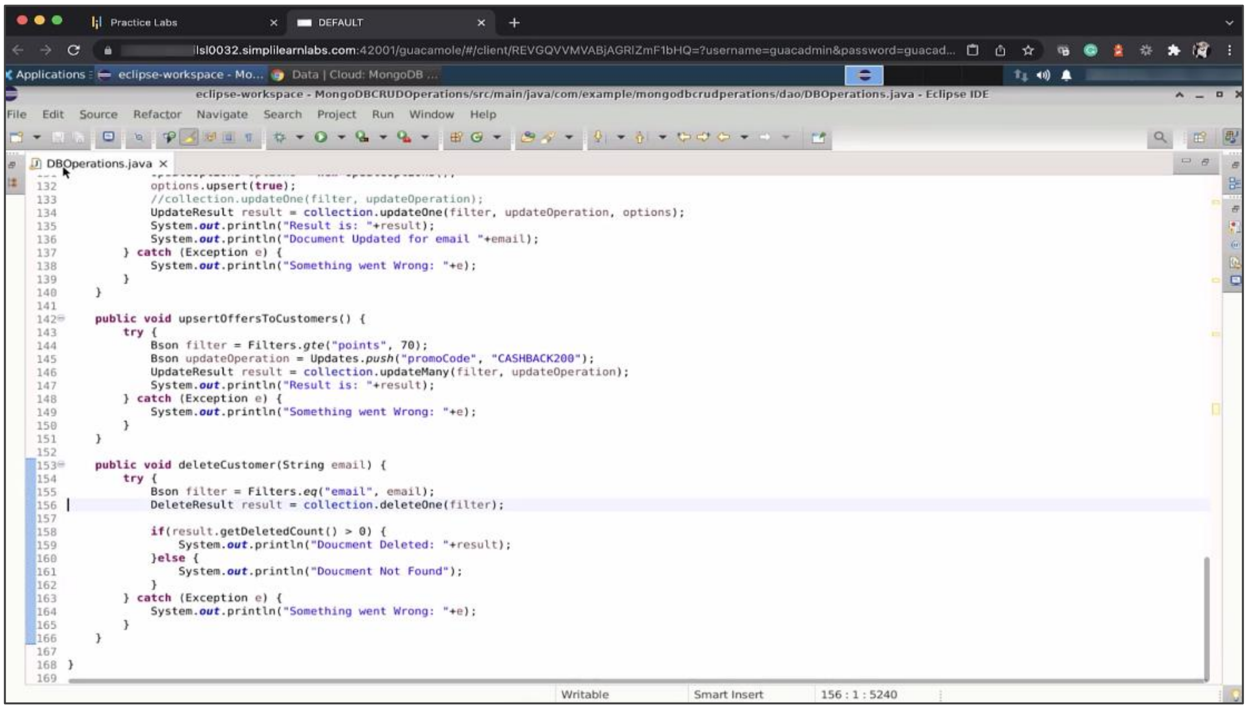
Note: Please refer to the previous demo on how to create the **MongoDBCRUDOperations** project

1.3 Create a new method named **deleteCustomer**



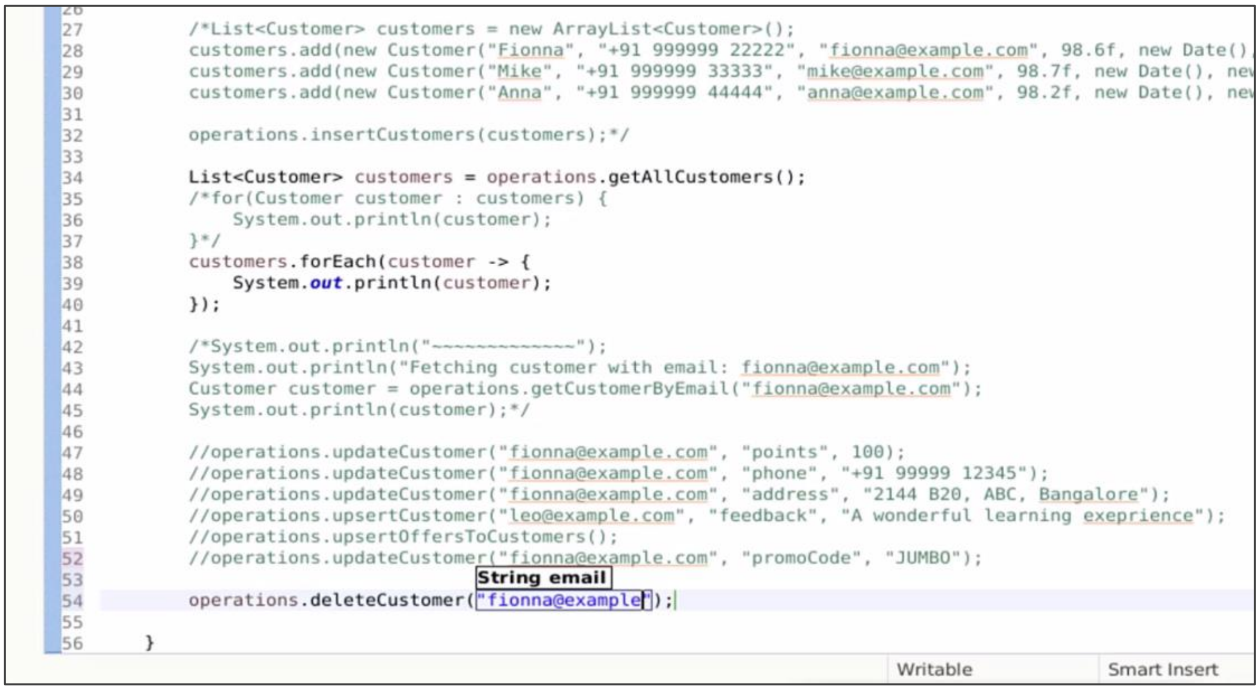
```java
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }

    }

    public void upsertCustomer(String email, String key, String value) {
        try {
            Bson filter = Filters.eq("email", email);
            Bson updateOperation = Updates.push(key, value);
            UpdateOptions options = new UpdateOptions();
            options.upsert(true);
            //collection.updateOne(filter, updateOperation);
            UpdateResult result = collection.updateOne(filter, updateOperation, options);
            System.out.println("Result is: "+result);
            System.out.println("Document Updated for email "+email);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void upsertOffersToCustomers() {
        try {
            Bson filter = Filters.gte("points", 70);
            Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
            UpdateResult result = collection.updateMany(filter, updateOperation);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomer(String email) {

    }

}
```

1.4 Add a try catch block and write the code to delete the document if the count is greater than **0**



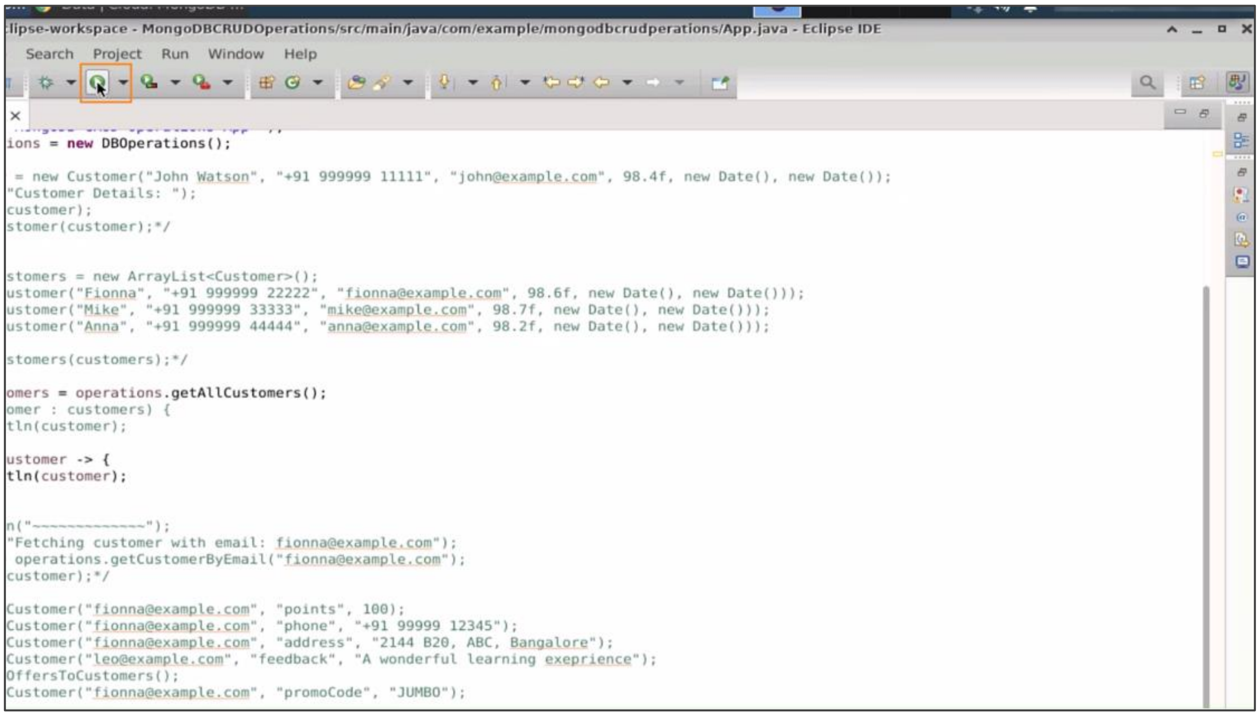
```
            options.upsert(true);
            //collection.updateOne(filter, updateOperation);
            UpdateResult result = collection.updateOne(filter, updateOperation, options);
            System.out.println("Result is: "+result);
            System.out.println("Document Updated for email "+email);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void upsertOffersToCustomers() {
        try {
            Bson filter = Filters.gte("points", 70);
            Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
            UpdateResult result = collection.updateMany(filter, updateOperation);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomer(String email) {
        try {
            Bson filter = Filters.eq("email", email);
            DeleteResult result = collection.deleteOne(filter);

            if(result.getDeletedCount() > 0) {
                System.out.println("Doucment Deleted: "+result);
            }else {
                System.out.println("Doucment Not Found");
            }
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

}
```

1.5 Navigate back to the **App.java** file and write the **delete** operation

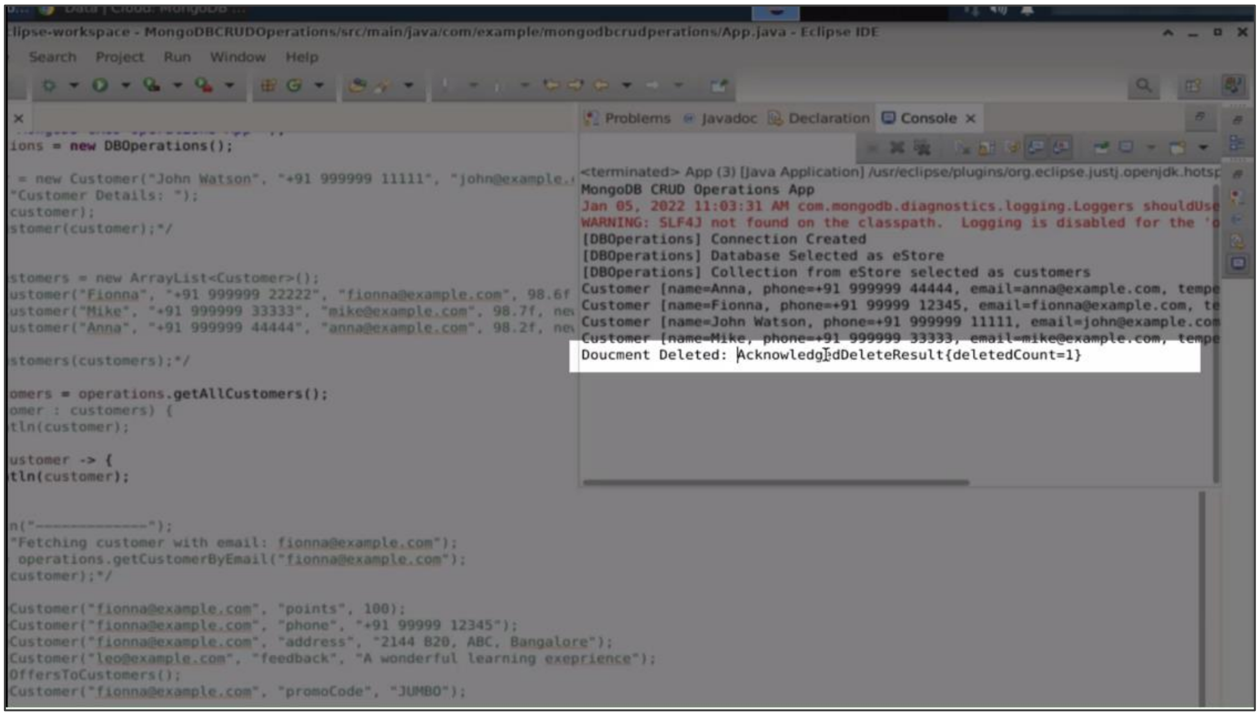


```
        /*List<Customer> customers = new ArrayList<Customer>();
        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date()
        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), ne
        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), ne

        operations.insertCustomers(customers);*/

        List<Customer> customers = operations.getAllCustomers();
        /*for(Customer customer : customers) {
            System.out.println(customer);
        }*/
        customers.forEach(customer -> {
            System.out.println(customer);
        });

        /*System.out.println("~~~~~~~~~~~~~~");
        System.out.println("Fetching customer with email: fionna@example.com");
        Customer customer = operations.getCustomerByEmail("fionna@example.com");
        System.out.println(customer);*/

        //operations.updateCustomer("fionna@example.com", "points", 100);
        //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
        //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
        //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
        //operations.upsertOffersToCustomers();
        //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
                                   String email
        operations.deleteCustomer("fionna@example");

    }
```

1.6 Save the file and run the code



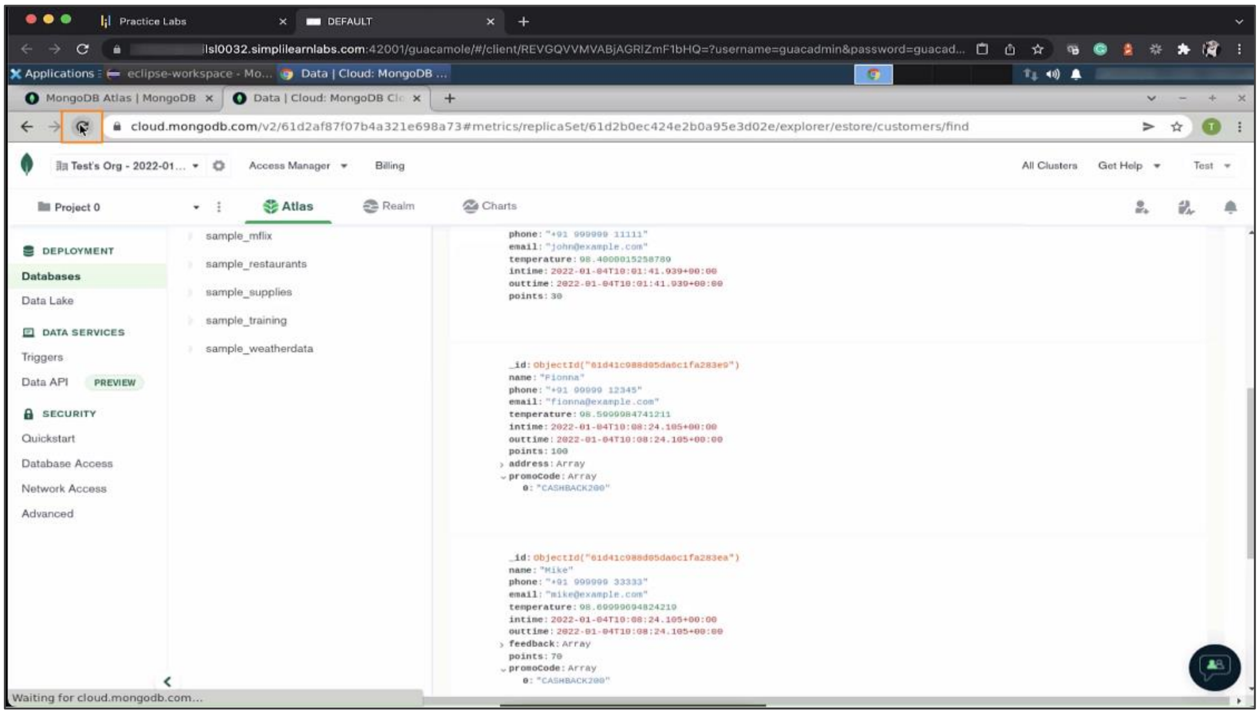
```
ions = new DBOperations();

 = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
"Customer Details: ");
customer);
stomer(customer);*/


stomers = new ArrayList<Customer>();
ustomer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
ustomer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
ustomer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

stomers(customers);*/

omers = operations.getAllCustomers();
omer : customers) {
tln(customer);

ustomer -> {
tln(customer);


n("~~~~~~~~~~~~~");
"Fetching customer with email: fionna@example.com");
 operations.getCustomerByEmail("fionna@example.com");
customer);*/

Customer("fionna@example.com", "points", 100);
Customer("fionna@example.com", "phone", "+91 99999 12345");
Customer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
Customer("leo@example.com", "feedback", "A wonderful learning exeprience");
OffersToCustomers();
Customer("fionna@example.com", "promoCode", "JUMBO");
```



```
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
MongoDB CRUD Operations App
Jan 05, 2022 11:03:31 AM com.mongodb.diagnostics.logging.Loggers shouldUse
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, tempe
Customer [name=Fionna, phone=+91 99999 12345, email=fionna@example.com, te
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, tempe
Doucment Deleted: AcknowledgedDeleteResult{deletedCount=1}
```

You should see the deleted count as 1.

1.7 Return to the database and refresh

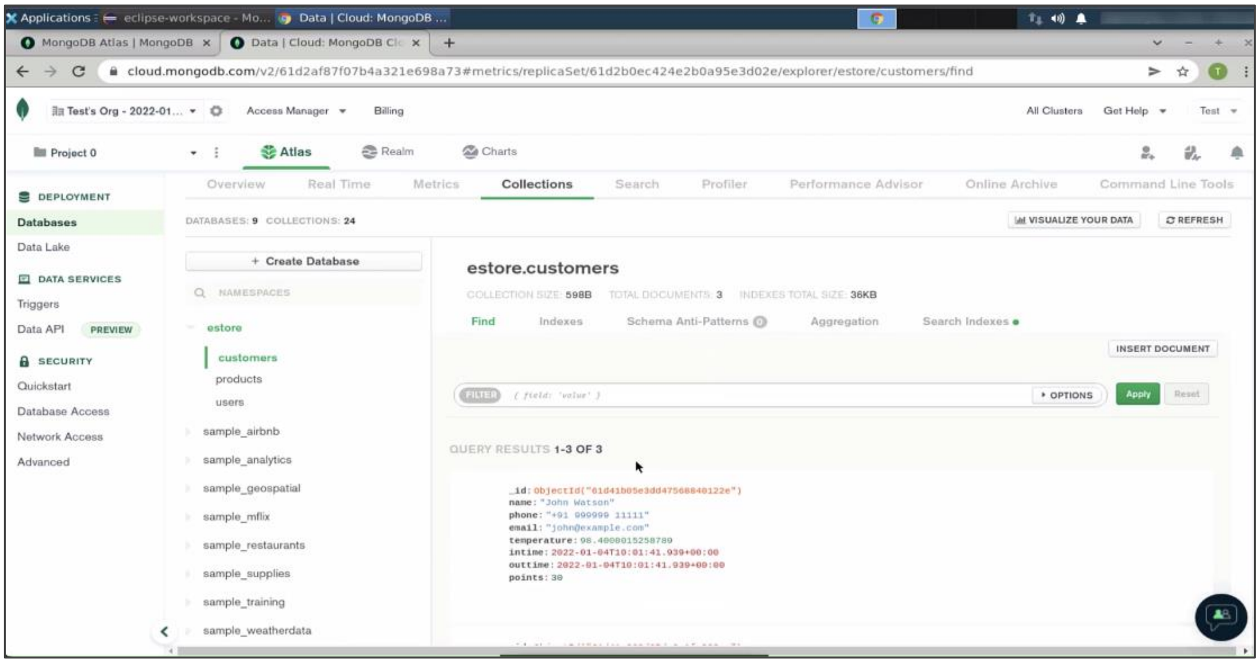


[redacted]lsl0032.simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABjAGRlZmF1bHQ=?username=guacadmin&password=guacad...

cloud.mongodb.com/v2/61d2af87f07b4a321e698a73#metrics/replicaSet/61d2b0ec424e2b0a95e3d02e/explorer/estore/customers/find

```
phone: "+91 999999 11111"
email: "john@example.com"
temperature: 98.4000015258789
intime: 2022-01-04T10:01:41.939+00:00
outtime: 2022-01-04T10:01:41.939+00:00
points: 30

_id: ObjectId("61d41c988d05da6c1fa283e9")
name: "Fionna"
phone: "+91 99999 12345"
email: "fionna@example.com"
temperature: 98.5999984741211
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
points: 100
> address: Array
v promoCode: Array
    0: "CASHBACK200"

_id: ObjectId("61d41c988d05da6c1fa283ea")
name: "Mike"
phone: "+91 999999 33333"
email: "mike@example.com"
temperature: 98.69999694824219
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
> feedback: Array
points: 70
v promoCode: Array
    0: "CASHBACK200"
```

Waiting for cloud.mongodb.com...



cloud.mongodb.com/v2/61d2af87f07b4a321e698a73#metrics/replicaSet/61d2b0ec424e2b0a95e3d02e/explorer/estore/customers/find

DATABASES: 9 COLLECTIONS: 24

estore.customers

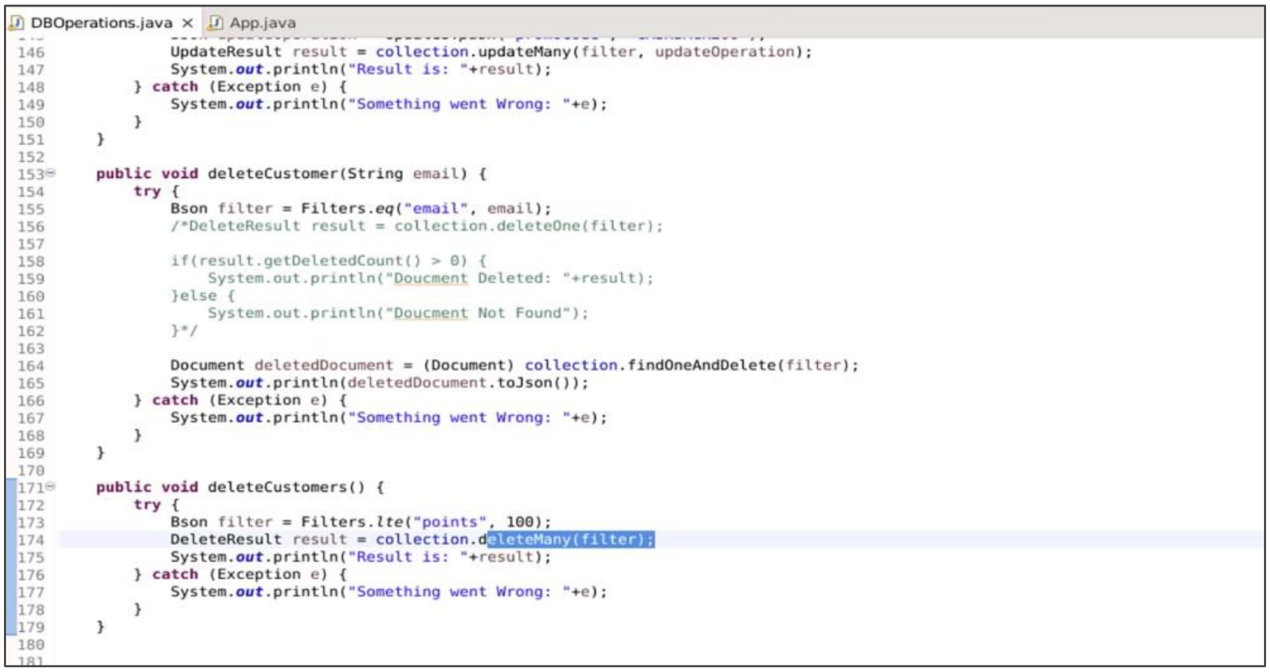
COLLECTION SIZE: 598B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("61d41b05e3dd47568840122e")
name: "John Watson"
phone: "+91 999999 11111"
email: "john@example.com"
temperature: 98.4000015258789
intime: 2022-01-04T10:01:41.939+00:00
outtime: 2022-01-04T10:01:41.939+00:00
points: 30
```

You should see only three documents now, compared to four previously.

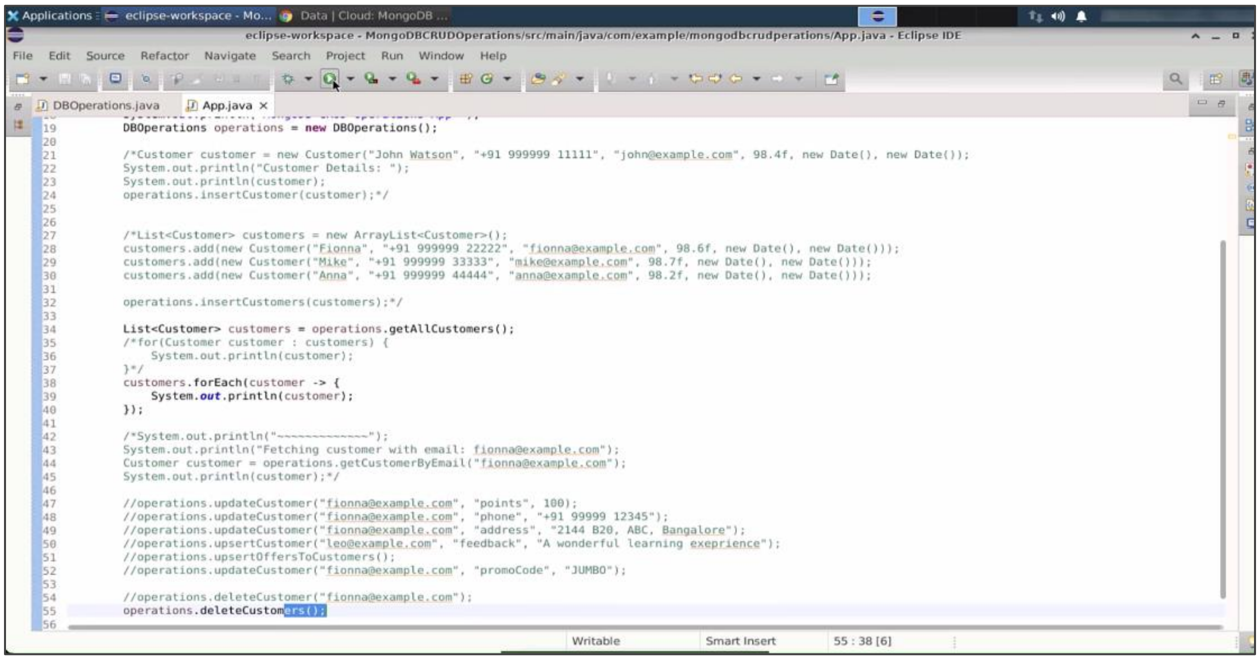
1.8 Go back to **App.java** and write a new operation **deleteCustomers** to delete the customers with points less than 100



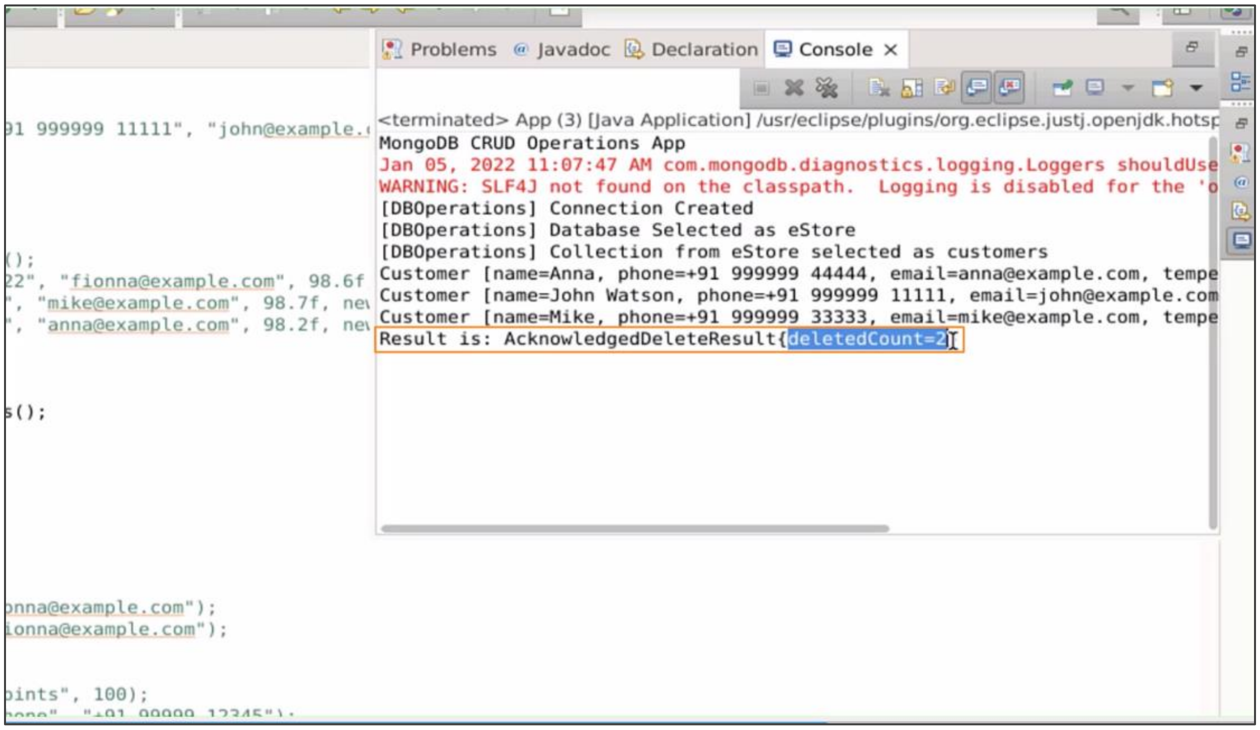
DBOperations.java × | App.java

```
            UpdateResult result = collection.updateMany(filter, updateOperation);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomer(String email) {
        try {
            Bson filter = Filters.eq("email", email);
            /*DeleteResult result = collection.deleteOne(filter);

            if(result.getDeletedCount() > 0) {
                System.out.println("Doucment Deleted: "+result);
            }else {
                System.out.println("Doucment Not Found");
            }*/

            Document deletedDocument = (Document) collection.findOneAndDelete(filter);
            System.out.println(deletedDocument.toJson());
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomers() {
        try {
            Bson filter = Filters.lte("points", 100);
            DeleteResult result = collection.deleteMany(filter);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }
```

1.9 Save the file and run the code


Applications
eclipse-workspace - Mo...
Data | Cloud: MongoDB ...
eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudperations/App.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
DBOperations.java
App.java ×
Writable
Smart Insert
55 : 38 [6]

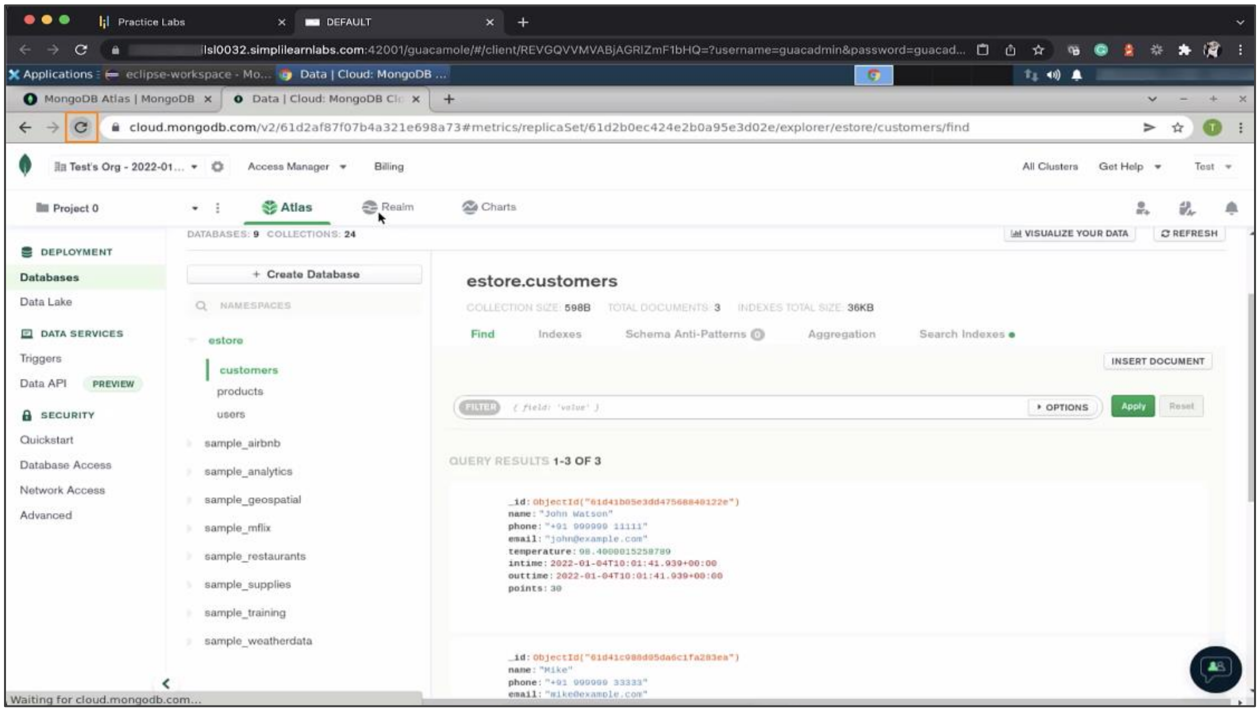
```
        DBOperations operations = new DBOperations();

        /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
        System.out.println("Customer Details: ");
        System.out.println(customer);
        operations.insertCustomer(customer);*/


        /*List<Customer> customers = new ArrayList<Customer>();
        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

        operations.insertCustomers(customers);*/

        List<Customer> customers = operations.getAllCustomers();
        /*for(Customer customer : customers) {
            System.out.println(customer);
        }*/
        customers.forEach(customer -> {
            System.out.println(customer);
        });

        /*System.out.println("~~~~~~~~~~~~~");
        System.out.println("Fetching customer with email: fionna@example.com");
        Customer customer = operations.getCustomerByEmail("fionna@example.com");
        System.out.println(customer);*/

        //operations.updateCustomer("fionna@example.com", "points", 100);
        //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
        //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
        //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
        //operations.upsertOffersToCustomers();
        //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");

        //operations.deleteCustomer("fionna@example.com");
        operations.deleteCustomers();
```



```
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
MongoDB CRUD Operations App
Jan 05, 2022 11:07:47 AM com.mongodb.diagnostics.logging.Loggers shouldUse
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, tempe
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, tempe
Result is: AcknowledgedDeleteResult{deletedCount=2}
```

You should see two other documents deleted and the **deletedCount=2**

1.10 Go to the database and refresh



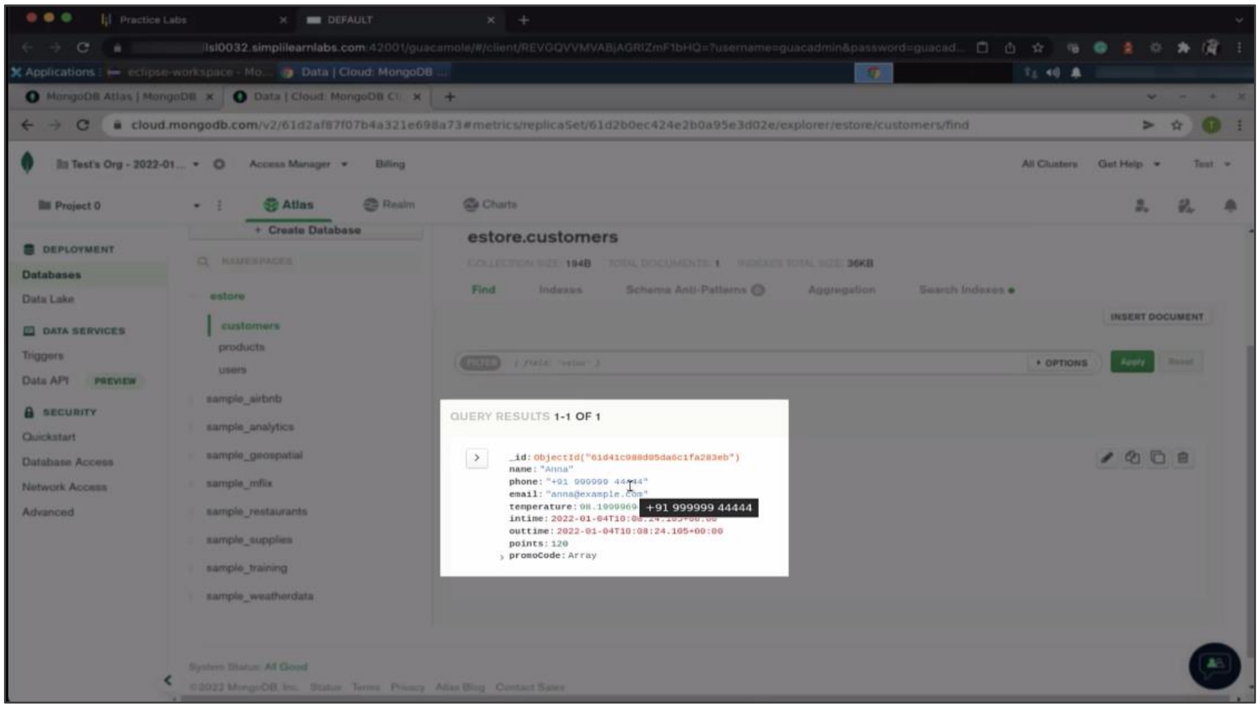
cloud.mongodb.com/v2/61d2af87f07b4a321e698a73#metrics/replicaSet/61d2b0ec424e2b0a95e3d02e/explorer/estore/customers/find

estore.customers

COLLECTION SIZE: 598B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS 1-3 OF 3

```
_id: ObjectId("61d41b05e3dd47568840122e")
name: "John Watson"
phone: "+91 999999 11111"
email: "john@example.com"
temperature: 98.4000015258789
intime: 2022-01-04T10:01:41.939+00:00
outtime: 2022-01-04T10:01:41.939+00:00
points: 30

_id: ObjectId("61d41c988d05da6c1fa283ea")
name: "Mike"
phone: "+91 999999 33333"
email: "mike@example.com"
```

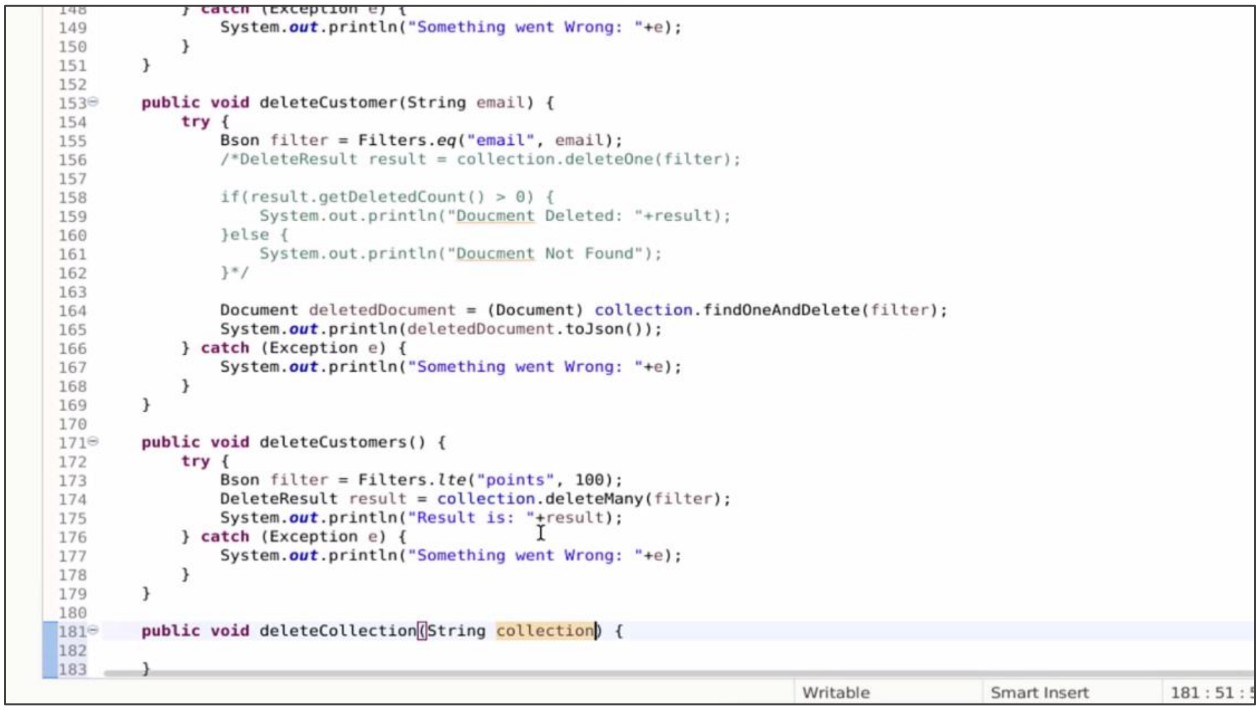


QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("61d41c988d05da6c1fa283eb")
name: "Anna"
phone: "+91 999999 44444"
email: "anna@example.com"
temperature: 98.1999969
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
points: 120
promoCode: Array
```

You should see only one record remaining and the others deleted.

1.11 Navigate back to **DBOperations.java** and create a new method to delete the collection

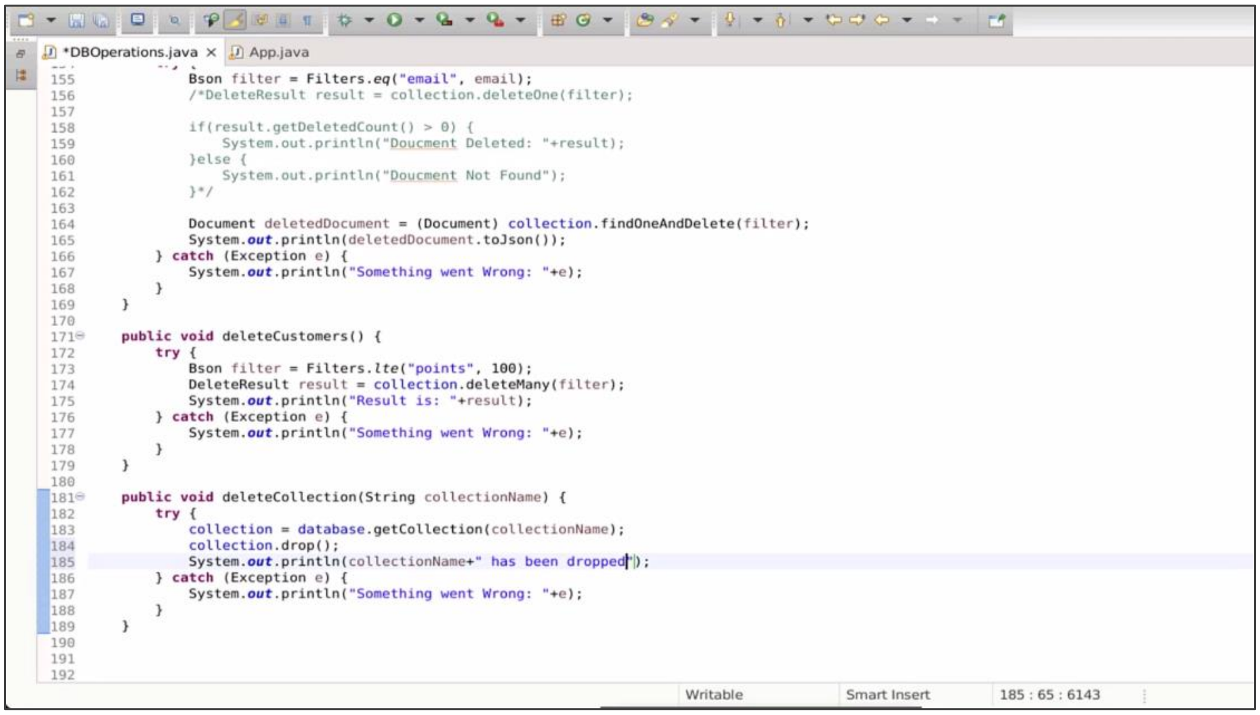


```
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomer(String email) {
        try {
            Bson filter = Filters.eq("email", email);
            /*DeleteResult result = collection.deleteOne(filter);

            if(result.getDeletedCount() > 0) {
                System.out.println("Doucment Deleted: "+result);
            }else {
                System.out.println("Doucment Not Found");
            }*/

            Document deletedDocument = (Document) collection.findOneAndDelete(filter);
            System.out.println(deletedDocument.toJson());
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomers() {
        try {
            Bson filter = Filters.lte("points", 100);
            DeleteResult result = collection.deleteMany(filter);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCollection(String collection) {

    }
```

Writable   Smart Insert   181 : 51 : 5

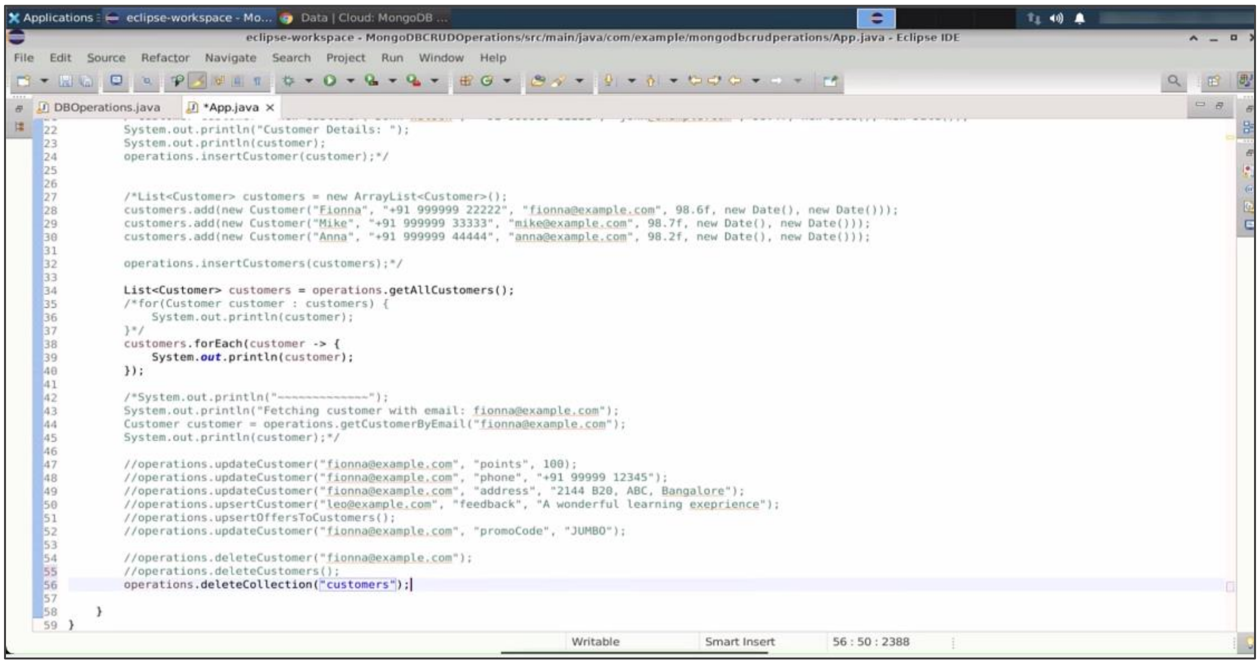
Step 2: Handle exceptions

2.1 In the **DBOperations.java** file, write a try catch block to handle the exception:



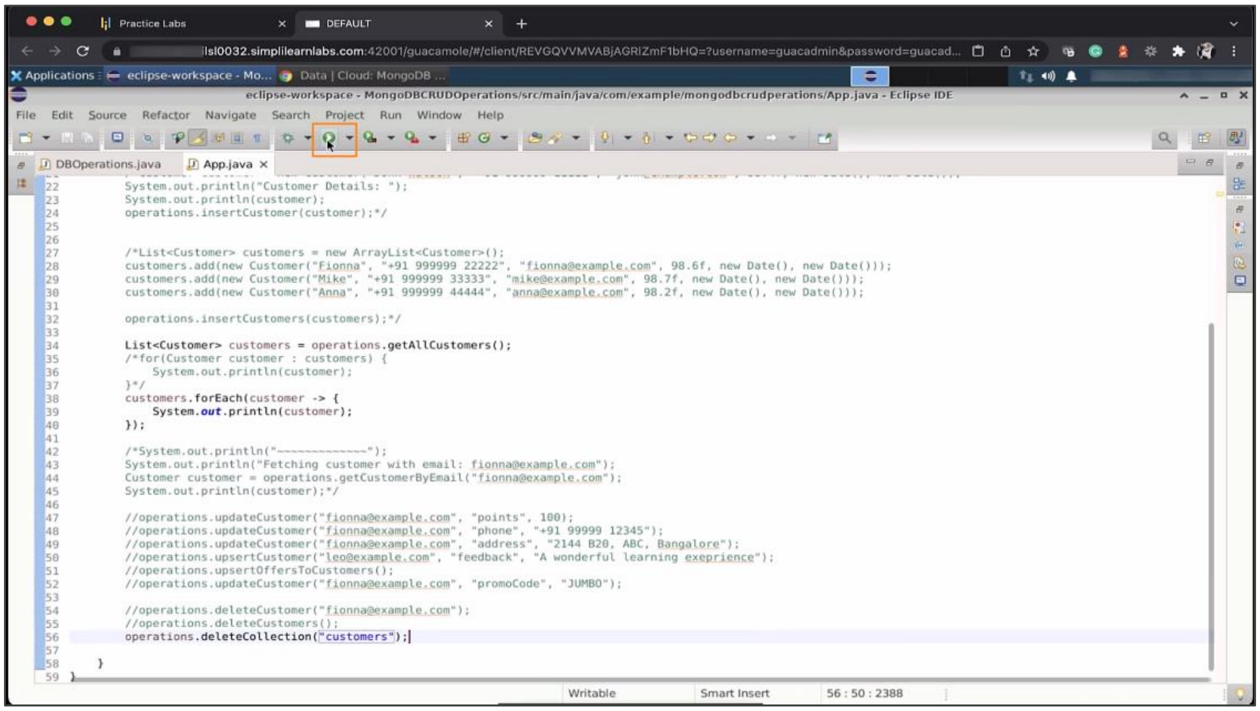
```java
            Bson filter = Filters.eq("email", email);
            /*DeleteResult result = collection.deleteOne(filter);

            if(result.getDeletedCount() > 0) {
                System.out.println("Doucment Deleted: "+result);
            }else {
                System.out.println("Doucment Not Found");
            }*/

            Document deletedDocument = (Document) collection.findOneAndDelete(filter);
            System.out.println(deletedDocument.toJson());
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCustomers() {
        try {
            Bson filter = Filters.lte("points", 100);
            DeleteResult result = collection.deleteMany(filter);
            System.out.println("Result is: "+result);
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }

    public void deleteCollection(String collectionName) {
        try {
            collection = database.getCollection(collectionName);
            collection.drop();
            System.out.println(collectionName+" has been dropped");
        } catch (Exception e) {
            System.out.println("Something went Wrong: "+e);
        }
    }
```

2.2 Navigate back to **App.java** and call the **deleteCollection** method

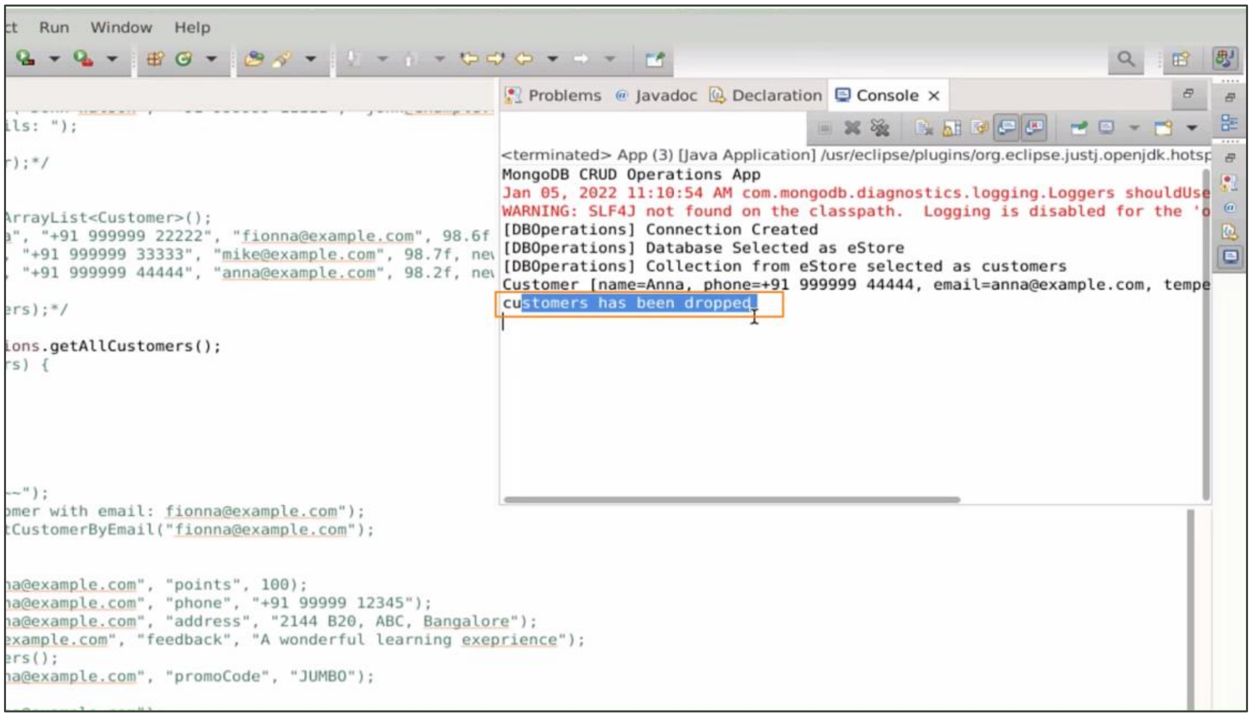


```java
        System.out.println("Customer Details: ");
        System.out.println(customer);
        operations.insertCustomer(customer);*/


        /*List<Customer> customers = new ArrayList<Customer>();
        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

        operations.insertCustomers(customers);*/

        List<Customer> customers = operations.getAllCustomers();
        /*for(Customer customer : customers) {
            System.out.println(customer);
        }*/
        customers.forEach(customer -> {
            System.out.println(customer);
        });

        /*System.out.println("~~~~~~~~~~~~~");
        System.out.println("Fetching customer with email: fionna@example.com");
        Customer customer = operations.getCustomerByEmail("fionna@example.com");
        System.out.println(customer);*/

        //operations.updateCustomer("fionna@example.com", "points", 100);
        //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
        //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
        //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
        //operations.upsertOffersToCustomers();
        //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");

        //operations.deleteCustomer("fionna@example.com");
        //operations.deleteCustomers();
        operations.deleteCollection("customers");

    }
}
```

2.3 Save the file and run the code


Practice Labs
DEFAULT
eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudperations/App.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
DBOperations.java
App.java
Writable
Smart Insert
56 : 50 : 2388

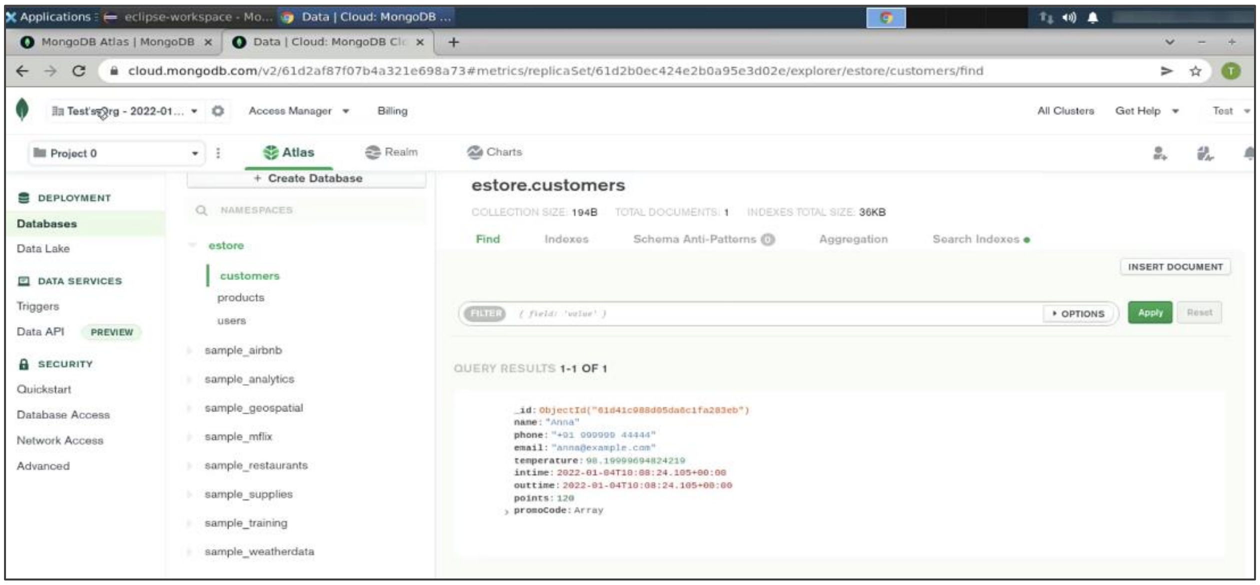
```
        System.out.println("Customer Details: ");
        System.out.println(customer);
        operations.insertCustomer(customer);*/


        /*List<Customer> customers = new ArrayList<Customer>();
        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

        operations.insertCustomers(customers);*/

        List<Customer> customers = operations.getAllCustomers();
        /*for(Customer customer : customers) {
            System.out.println(customer);
        }*/
        customers.forEach(customer -> {
            System.out.println(customer);
        });

        /*System.out.println("~~~~~~~~~~~~~");
        System.out.println("Fetching customer with email: fionna@example.com");
        Customer customer = operations.getCustomerByEmail("fionna@example.com");
        System.out.println(customer);*/

        //operations.updateCustomer("fionna@example.com", "points", 100);
        //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
        //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
        //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
        //operations.upsertOffersToCustomers();
        //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");

        //operations.deleteCustomer("fionna@example.com");
        //operations.deleteCustomers();
        operations.deleteCollection("customers");

    }
}
```



```
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
MongoDB CRUD Operations App
Jan 05, 2022 11:10:54 AM com.mongodb.diagnostics.logging.Loggers shouldUse
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
Customer [name=Anna, phone=+91 999999 44444, email=anna@example.com, tempe
customers has been dropped.
```

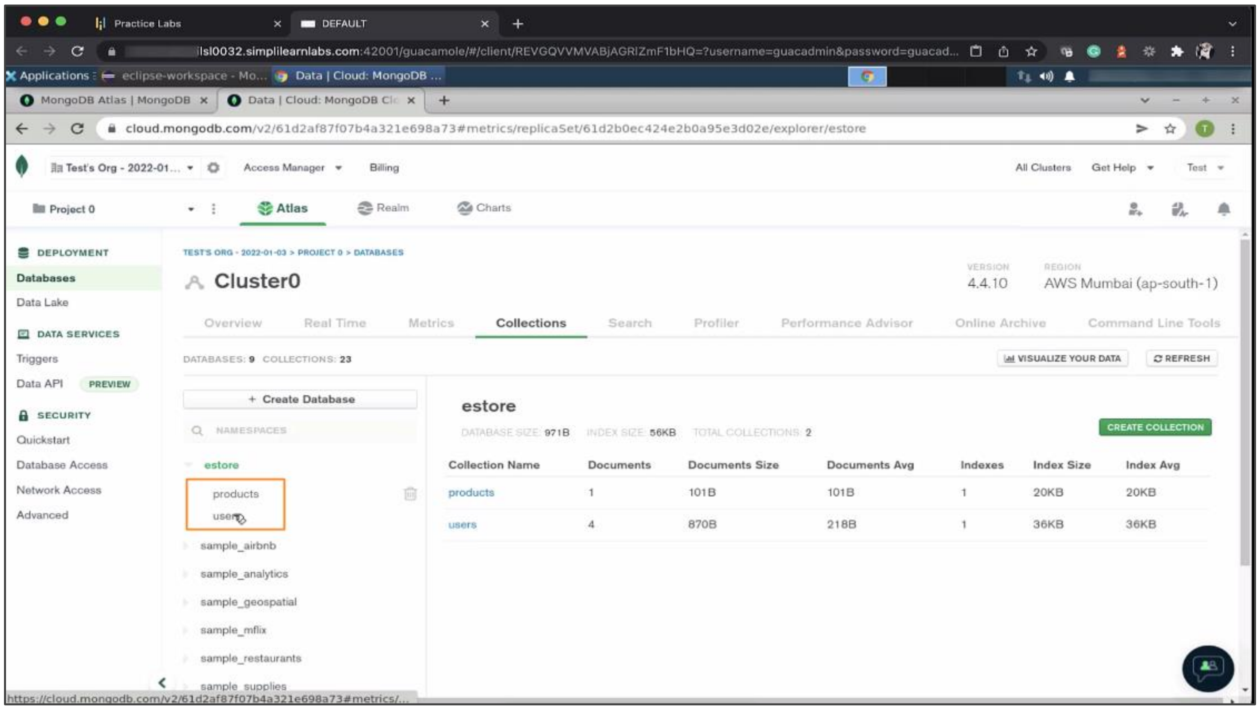
You should see the message **customers have been dropped**.

2.4 Go back to the database and refresh



QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("61d41c988d05da6c1fa283eb")
name: "Anna"
phone: "+91 999999 44444"
email: "anna@example.com"
temperature: 98.19999694824219
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
points: 120
promoCode: Array
```



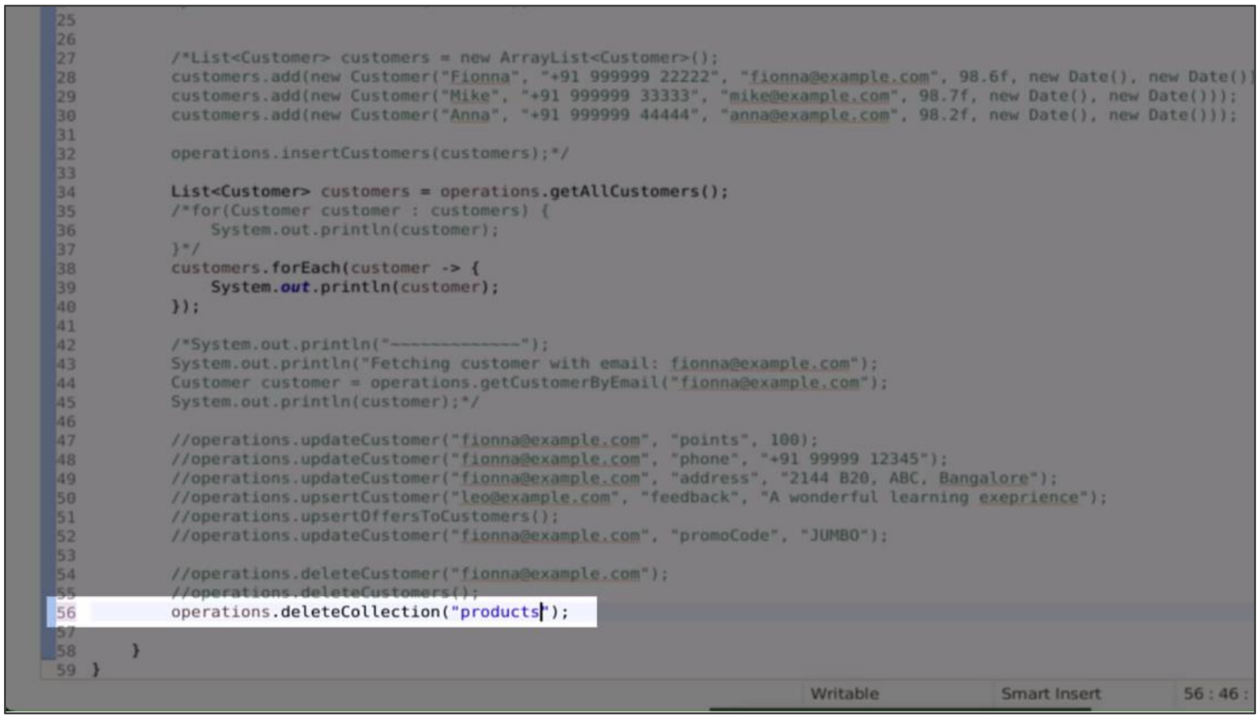
estore

DATABASE SIZE: 971B INDEX SIZE: 56KB TOTAL COLLECTIONS: 2

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
products	1	101B	101B	1	20KB	20KB
users	4	870B	218B	1	36KB	36KB

You should see the products and collections listed.

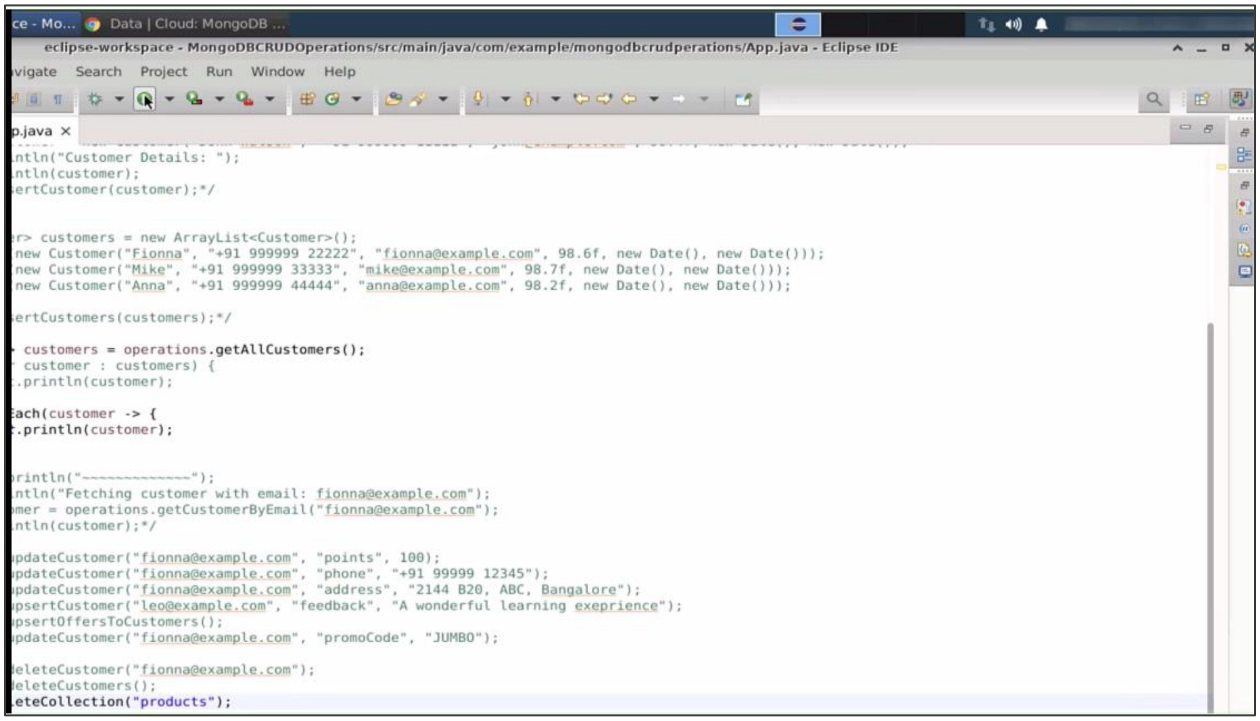
2.5 Navigate back to **App.java** and change the input to **products**



```
        /*List<Customer> customers = new ArrayList<Customer>();
        customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()
        customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
        customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

        operations.insertCustomers(customers);*/

        List<Customer> customers = operations.getAllCustomers();
        /*for(Customer customer : customers) {
            System.out.println(customer);
        }*/
        customers.forEach(customer -> {
            System.out.println(customer);
        });

        /*System.out.println("~~~~~~~~~~~~~");
        System.out.println("Fetching customer with email: fionna@example.com");
        Customer customer = operations.getCustomerByEmail("fionna@example.com");
        System.out.println(customer);*/

        //operations.updateCustomer("fionna@example.com", "points", 100);
        //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
        //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
        //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
        //operations.upsertOffersToCustomers();
        //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");

        //operations.deleteCustomer("fionna@example.com");
        //operations.deleteCustomers();
        operations.deleteCollection("products");

    }
}
```

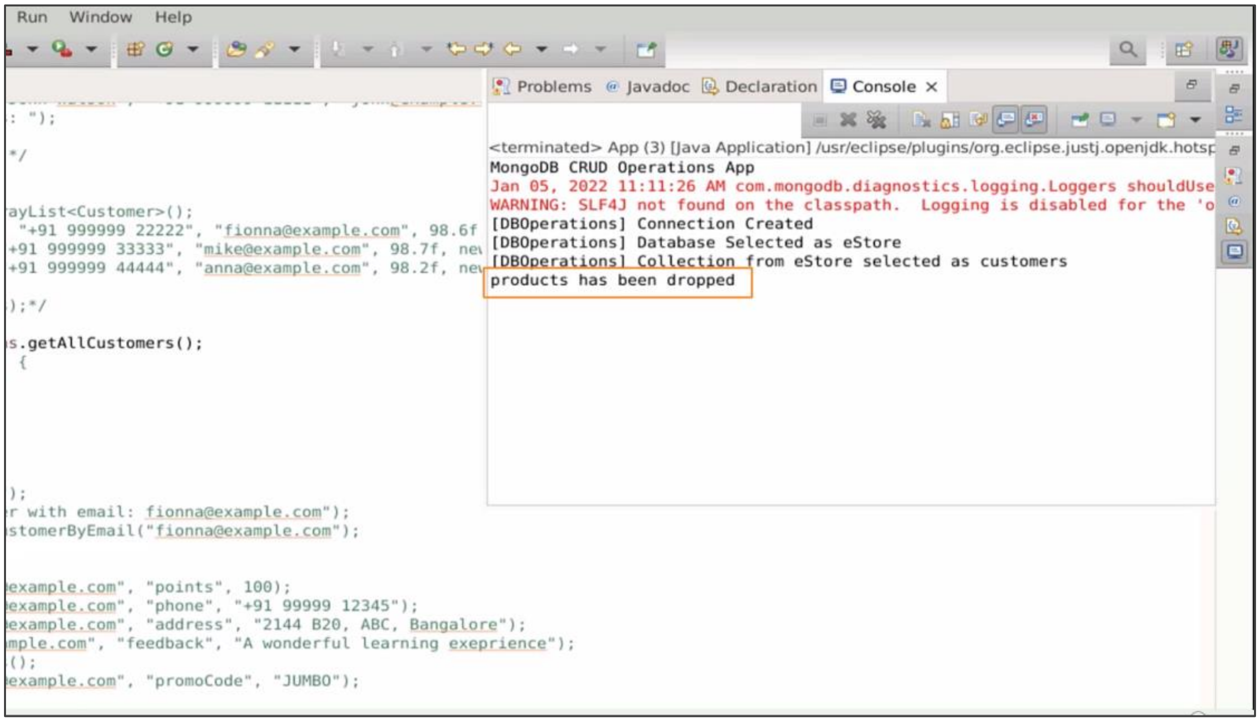
Writable   Smart Insert   56 : 46

2.6 Rerun the code



ce - Mo...
Data | Cloud: MongoDB ...
eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudperations/App.java - Eclipse IDE
vigate Search Project Run Window Help
p.java

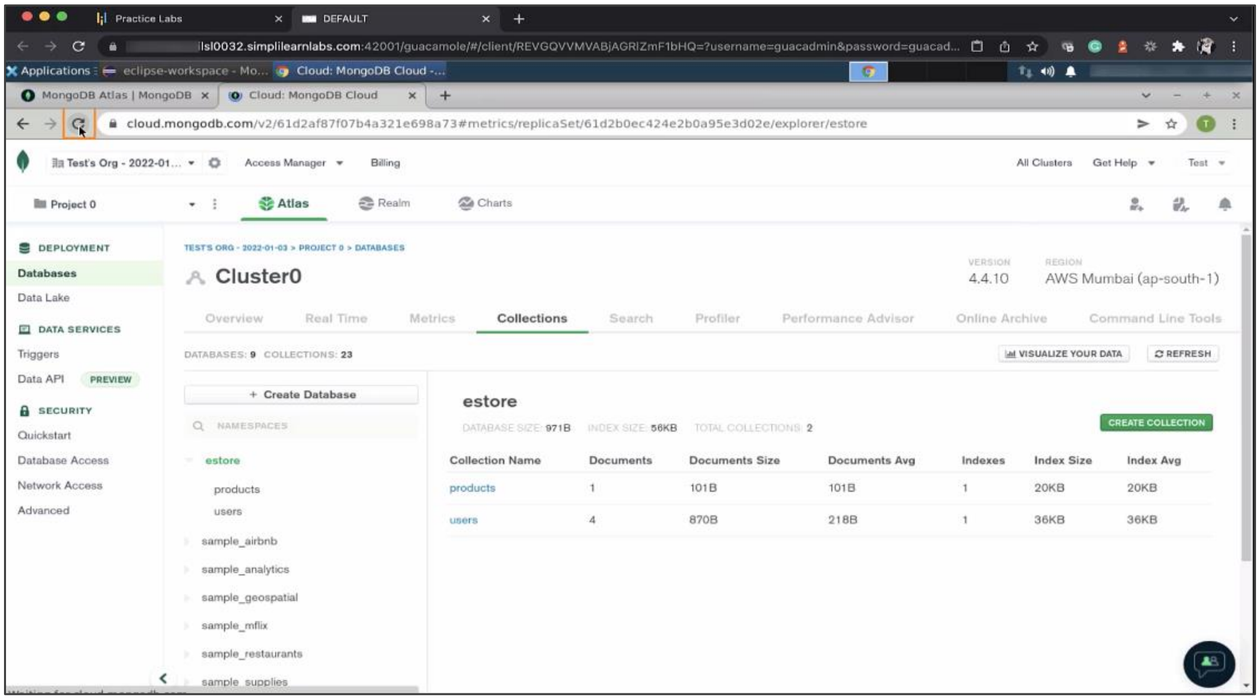
```
ntln("Customer Details: ");
ntln(customer);
ertCustomer(customer);*/


r> customers = new ArrayList<Customer>();
new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));

ertCustomers(customers);*/

 customers = operations.getAllCustomers();
 customer : customers) {
.println(customer);

ach(customer -> {
.println(customer);


rintln("~~~~~~~~~~~~~");
ntln("Fetching customer with email: fionna@example.com");
mer = operations.getCustomerByEmail("fionna@example.com");
ntln(customer);*/

pdateCustomer("fionna@example.com", "points", 100);
pdateCustomer("fionna@example.com", "phone", "+91 99999 12345");
pdateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
psertCustomer("leo@example.com", "feedback", "A wonderful learning exeprience");
psertOffersToCustomers();
pdateCustomer("fionna@example.com", "promoCode", "JUMBO");

eleteCustomer("fionna@example.com");
eleteCustomers();
eteCollection("products");
```



```
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
MongoDB CRUD Operations App
Jan 05, 2022 11:11:26 AM com.mongodb.diagnostics.logging.Loggers shouldUse
WARNING: SLF4J not found on the classpath.  Logging is disabled for the 'o
[DBOperations] Connection Created
[DBOperations] Database Selected as eStore
[DBOperations] Collection from eStore selected as customers
products has been dropped
```

You should see the message **products has been dropped**.

2.7 Go back to the database and refresh

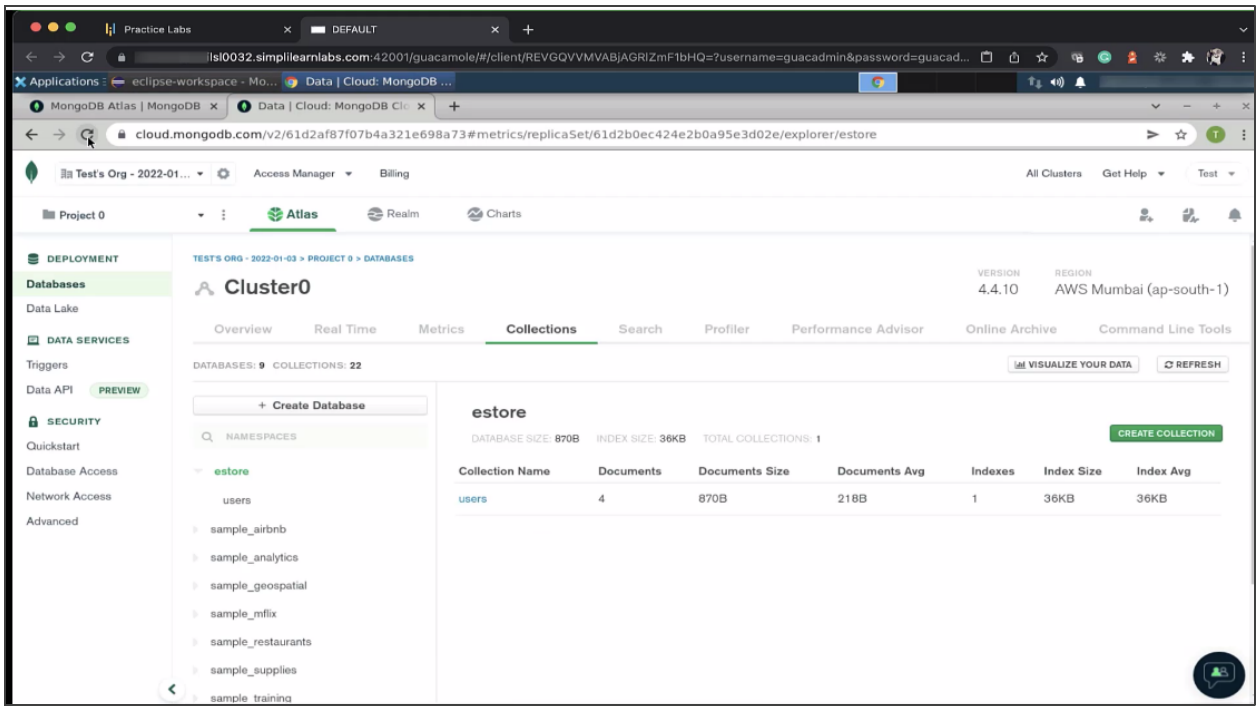


estore

DATABASE SIZE: 971B   INDEX SIZE: 56KB   TOTAL COLLECTIONS: 2

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
products	1	101B	101B	1	20KB	20KB
users	4	870B	218B	1	36KB	36KB

You should see that the **products** collection has been deleted.



Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
users	4	870B	218B	1	36KB	36KB

The **deleteOne** operation can be used to delete a single document. The **deleteMany** operation can be used to delete multiple documents based on a condition.

By following these steps, you have successfully deleted documents from the MongoDB collection using the delete operation with try catch blocks.