

## 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

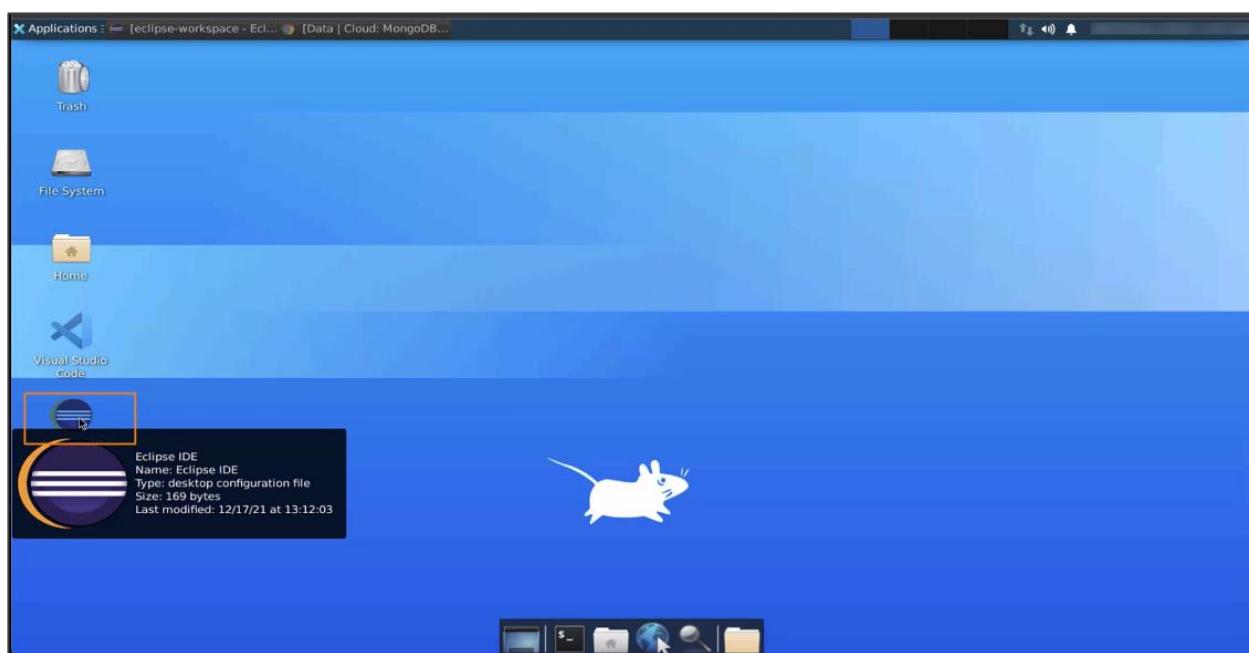
**Prerequisites:** None

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



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

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for **MongoDBCRUDOperations**, including **src/main/java**, **src/test/java**, **JRE System Library [JavaSE-1.8]**, **Maven Dependencies**, and **target**.
- DBOperations.java:** The code is displayed in the central editor window. It defines a class **DBOperations** that initializes a **MongoClient**, **MongoDatabase**, and **MongoCollection**. It includes a constructor and a **try** block that prints connection details.
- Outline View:** On the right, it shows the class structure with methods like **insertCustomer**, **insertCustomers**, **convertDocumentToCustomer**, **getALLCustomers**, **getCustomerByEmail**, and **updateCustomer**.
- Problems, Javadoc, Declaration, Console:** These tabs are visible at the bottom of the editor.

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

## 1.3 Create a new method named **deleteCustomer**

The screenshot shows the code editor with the **DBOperations.java** file open. A new method, **deleteCustomer**, has been added to the class. The code is as follows:

```

120     } catch (Exception e) {
121         System.out.println("Something went Wrong: "+e);
122     }
123 }
124 }
125
126 public void upsertCustomer(String email, String key, String value) {
127     try {
128         Bson filter = Filters.eq("email", email);
129         Bson updateOperation = Updates.push(key, value);
130         UpdateOptions options = new UpdateOptions();
131         options.upsert(true);
132         //collection.updateOne(filter, updateOperation);
133         UpdateResult result = collection.updateOne(filter, updateOperation, options);
134         System.out.println("Result is: "+result);
135         System.out.println("Document Updated for email "+email);
136     } catch (Exception e) {
137         System.out.println("Something went Wrong: "+e);
138     }
139 }
140
141 public void upsertOffersToCustomers() {
142     try {
143         Bson filter = Filters.gte("points", 70);
144         Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
145         UpdateResult result = collection.updateMany(filter, updateOperation);
146         System.out.println("Result is: "+result);
147     } catch (Exception e) {
148         System.out.println("Something went Wrong: "+e);
149     }
150 }
151
152 public void deleteCustomer(String email) {
153 }
154
155
156 }
157

```

The line **152 public void deleteCustomer(String email) {** is highlighted in blue, indicating it is the new method being created.

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

```

132     options.upsert(true);
133     //collection.updateOne(filter, updateOperation);
134     UpdateResult result = collection.updateOne(filter, updateOperation, options);
135     System.out.println("Result is: "+result);
136     System.out.println("Document Updated for email "+email);
137 } catch (Exception e) {
138     System.out.println("Something went Wrong: "+e);
139 }
140 }
141
142 public void upsertOffersToCustomers() {
143     try {
144         Bson filter = Filters.gte("points", 70);
145         Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
146         UpdateResult result = collection.updateMany(filter, updateOperation);
147         System.out.println("Result is: "+result);
148     } catch (Exception e) {
149         System.out.println("Something went Wrong: "+e);
150     }
151 }
152
153 public void deleteCustomer(String email) {
154     try {
155         Bson filter = Filters.eq("email", email);
156         DeleteResult result = collection.deleteOne(filter);
157
158         if(result.getDeletedCount() > 0) {
159             System.out.println("Document Deleted: "+result);
160         } else {
161             System.out.println("Document Not Found");
162         }
163     } catch (Exception e) {
164         System.out.println("Something went Wrong: "+e);
165     }
166 }
167
168 }
169 
```

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

```

26
27     /*List<Customer> customers = new ArrayList<Customer>();
28     customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date());
29     customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date());
30     customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date());
31
32     operations.insertCustomers(customers);*/
33
34     List<Customer> customers = operations.getAllCustomers();
35     /*for(Customer customer : customers) {
36         System.out.println(customer);
37     }*/
38     customers.forEach(customer -> {
39         System.out.println(customer);
40     });
41
42     /*System.out.println("~~~~~");
43     System.out.println("Fetching customer with email: fionna@example.com");
44     Customer customer = operations.getCustomerByEmail("fionna@example.com");
45     System.out.println(customer);*/
46
47     //operations.updateCustomer("fionna@example.com", "points", 100);
48     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     //operations.upsertOffersToCustomers();
52     //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
53
54     String email
55     operations.deleteCustomer("fionna@example");
56 }
```

## 1.6 Save the file and run the code

Eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudoperations/App.java - Eclipse IDE

Search Project Run Window Help

X

ions = new DBOperations();

= new Customer("John Watson", "+91 999999 1111", "john@example.com", 98.4f, new Date(), new Date());  
"Customer Details: ");  
customer);  
stomer(customer);\*/

stomers = new ArrayList<Customer>();  
ustomer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date());  
ustomer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date());  
ustomer("Anna", "+91 999999 4444", "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 experience");  
OffersToCustomers();  
Customer("fionna@example.com", "promoCode", "JUMBO");

You should see the deleted count as 1.

## 1.7 Return to the database and refresh

The screenshot shows the MongoDB Cloud Explorer interface. On the left, the sidebar includes sections for Project 0, DEPLOYMENT (with Databases like Data Lake, sample\_mflix, sample\_restaurants, sample\_supplies, sample\_training, sample\_weatherdata), DATA SERVICES (Triggers, Data API PREVIEW, SECURITY), and a Quickstart section. The main area displays the 'Atlas' tab for the 'Project 0' database. Under 'Collections', the 'estore' collection is selected, showing its details: COLLECTION SIZE: 598B, TOTAL DOCUMENTS: 3, INDEXES TOTAL SIZE: 36KB. Below this, there are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A 'FILTER' button is present. At the bottom, the 'QUERY RESULTS 1-3 OF 3' section displays three documents:

```

_id: ObjectId("61d41c088d05da0c1fa283e9")
name: "John Watson"
phone: "+91 999999 11111"
email: "john@example.com"
temperature: 98.4
intime: 2022-01-04T10:01:41.939+00:00
outtime: 2022-01-04T10:01:41.939+00:00
points: 30

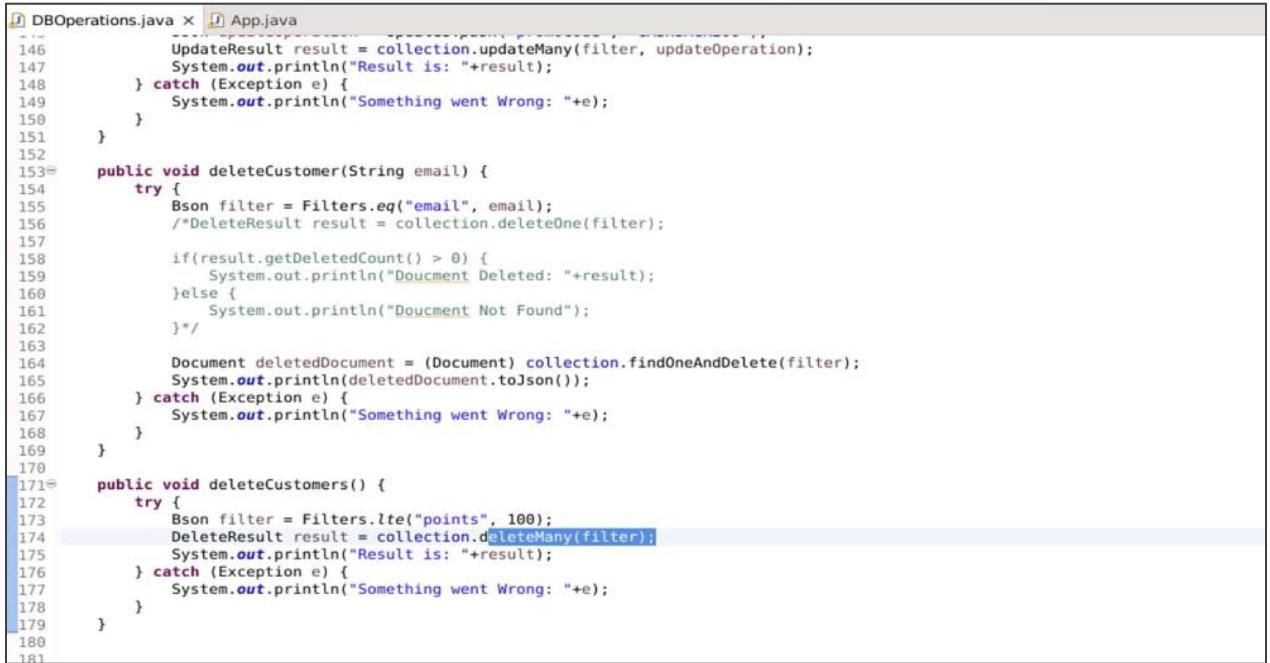
_id: ObjectId("61d41c088d05da0c1fa283ea")
name: "Mike"
phone: "+91 999999 33333"
email: "mike@example.com"
temperature: 98.5
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
feedback: []
points: 70
promoCode: Array
  0: "CASHBACK200"

_id: ObjectId("61d41c088d05da0c1fa283ea")
name: "Lonna"
phone: "+91 999999 12345"
email: "lonna@example.com"
temperature: 98.5999984741211
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
address: []
promoCode: Array
  0: "CASHBACK200"

```

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

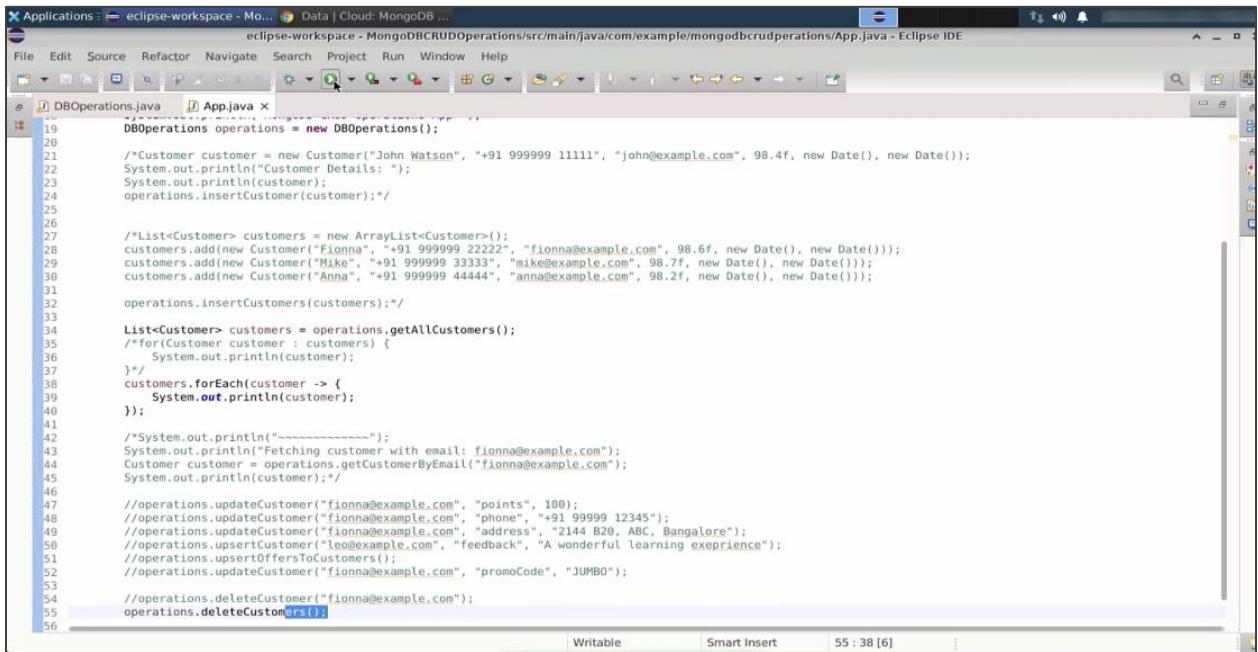


```

146     UpdateResult result = collection.updateMany(filter, updateOperation);
147     System.out.println("Result is: "+result);
148   } catch (Exception e) {
149     System.out.println("Something went Wrong: "+e);
150   }
151 }
152
153 public void deleteCustomer(String email) {
154   try {
155     Bson filter = Filters.eq("email", email);
156     /*DeleteResult result = collection.deleteOne(filter);
157
158     if(result.getDeletedCount() > 0) {
159       System.out.println("Document Deleted: "+result);
160     }else {
161       System.out.println("Document Not Found");
162     }*/
163
164     Document deletedDocument = (Document) collection.findOneAndDelete(filter);
165     System.out.println(deletedDocument.toJson());
166   } catch (Exception e) {
167     System.out.println("Something went Wrong: "+e);
168   }
169 }
170
171 public void deleteCustomers() {
172   try {
173     Bson filter = Filters.lte("points", 100);
174     DeleteResult result = collection.deleteMany(filter);
175     System.out.println("Result is: "+result);
176   } catch (Exception e) {
177     System.out.println("Something went Wrong: "+e);
178   }
179 }
180
181

```

## 1.9 Save the file and run the code



```

File Edit Source Refactor Navigate Search Project Run Window Help
DBOperations.java App.java X
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 1111", "john@example.com", 98.4f, new Date(), new Date());
22     System.out.println("Customer Details: ");
23     System.out.println(customer);
24     operations.insertCustomer(customer);*/
25
26
27     /*List<Customer> customers = new ArrayList<Customer>();
28     customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date());
29     customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date()));
30     customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32     operations.insertCustomers(customers);*/
33
34     List<Customer> customers = operations.getAllCustomers();
35     /*for(Customer customer : customers) {
36       System.out.println(customer);
37     }*/
38     customers.forEach(customer -> {
39       System.out.println(customer);
40     });
41
42     /*System.out.println("~~~~~");
43     System.out.println("Fetching customer with email: fionna@example.com");
44     Customer customer = operations.getCustomerByEmail("fionna@example.com");
45     System.out.println(customer);*/
46
47     //operations.updateCustomer("fionna@example.com", "points", 100);
48     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50     //operations.updateCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     //operations.updateOffersToCustomers();
52     //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
53
54     //operations.deleteCustomer("fionna@example.com");
55     operations.deleteCustomrs();
56

```

```
91 999999 11111", "john@example..  
);  
22", "fionna@example.com", 98.6f  
", "mike@example.com", 98.7f, ne  
, "anna@example.com", 98.2f, ne  
();  
  
());  
  
onna@example.com");  
Lonna@example.com");  
  
oints", 100);  
2000" "+91 99999 12345").  
  
<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

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with sections for Project 0, Atlas, and Charts. The main area shows the 'DEPLOYMENT' section with 'Databases' (9), 'Collections' (24), and a '+ Create Database' button. Below this is a 'Namespaces' section with a search bar and a list of namespaces including 'estore' (which is expanded) and 'sample\_airbnb', 'sample\_analytics', 'sample\_geospatial', 'sample\_mflix', 'sample\_restaurants', 'sample\_supplies', 'sample\_training', and 'sample\_weatherdata'. The 'estore' section contains 'customers', 'products', and 'users' collections. The 'customers' collection is selected, showing its details: Collection Size: 598B, Total Documents: 3, Indexes Total Size: 36KB. Below this are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A 'FILTER' bar with the query `{ field: 'value' }` is present. At the bottom, there are buttons for 'OPTIONS', 'Apply', and 'Reset'. The 'QUERY RESULTS 1-3 OF 3' section displays two documents:

```
_id: ObjectId("61d41b09e3dd47568840122e")
name: "John Watson"
phone: "+01 909999 1111"
email: "john@example.com"
temperature: 08.4000015259780
intime: 2022-01-04T0:01:41.939+00:00
outtime: 2022-01-04T0:01:41.939+00:00
points: 39
```

```
_id: ObjectId("61d41c980d5da6c1fa283ea")
name: "Mike"
phone: "+01 909999 3333"
email: "mike@example.com"
```

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with 'Project 0' selected. Under 'Databases', 'estore' is chosen, and under 'Collections', 'customers' is selected. The main area shows the 'estore.customers' collection with a count of 1948 documents. A specific document is highlighted in the 'QUERY RESULTS' section:

```

_id: ObjectId("61d41c088d85dadcfaf283eb")
name: "Anna"
phone: "+91 999999 4444"
email: "anna@example.com"
temperature: 98.19999999999999
intime: 2022-01-04T10:05:27.957Z
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

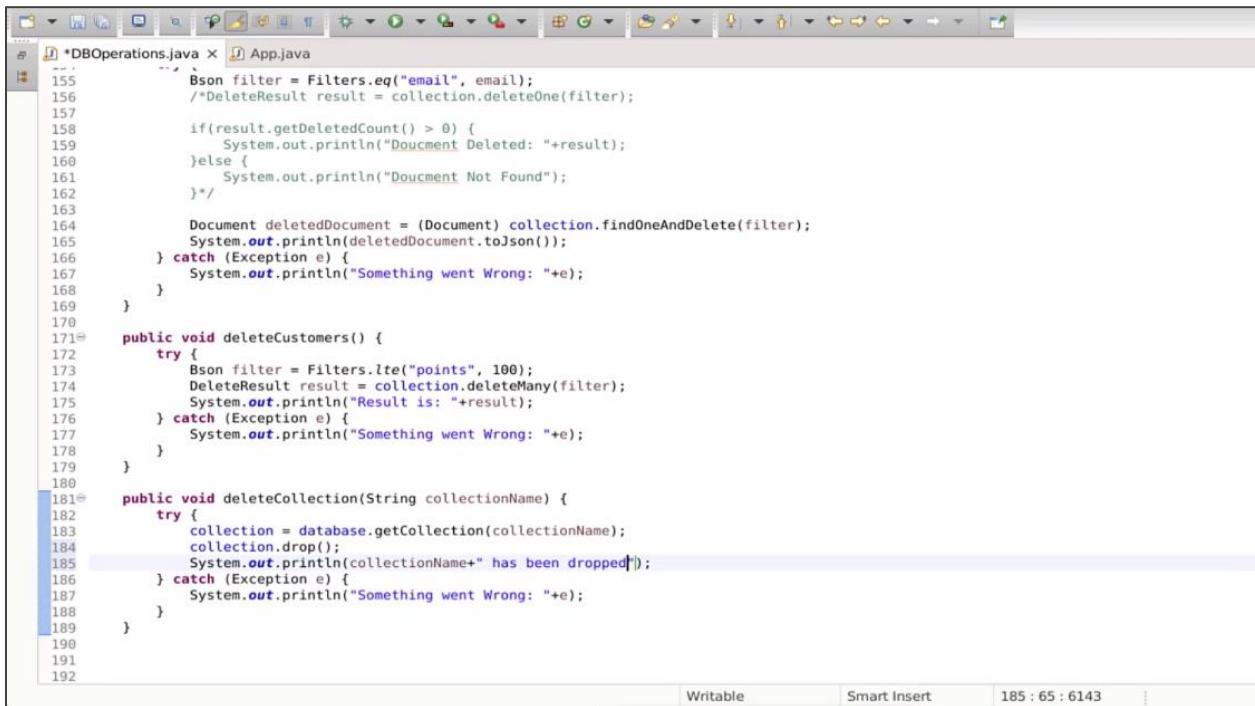
```

148         } catch (Exception e) {
149             System.out.println("Something went Wrong: "+e);
150         }
151     }
152
153     public void deleteCustomer(String email) {
154         try {
155             Bson filter = Filters.eq("email", email);
156             /*DeleteResult result = collection.deleteOne(filter);
157
158             if(result.getDeletedCount() > 0) {
159                 System.out.println("Document Deleted: "+result);
160             }else {
161                 System.out.println("Document Not Found");
162             }*/
163
164             Document deletedDocument = (Document) collection.findOneAndDelete(filter);
165             System.out.println(deletedDocument.toJson());
166         } catch (Exception e) {
167             System.out.println("Something went Wrong: "+e);
168         }
169     }
170
171     public void deleteCustomers() {
172         try {
173             Bson filter = Filters.lte("points", 100);
174             DeleteResult result = collection.deleteMany(filter);
175             System.out.println("Result is: "+result);
176         } catch (Exception e) {
177             System.out.println("Something went Wrong: "+e);
178         }
179     }
180
181     public void deleteCollection(String collection) {
182
183     }

```

## Step 2: Handle exceptions

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

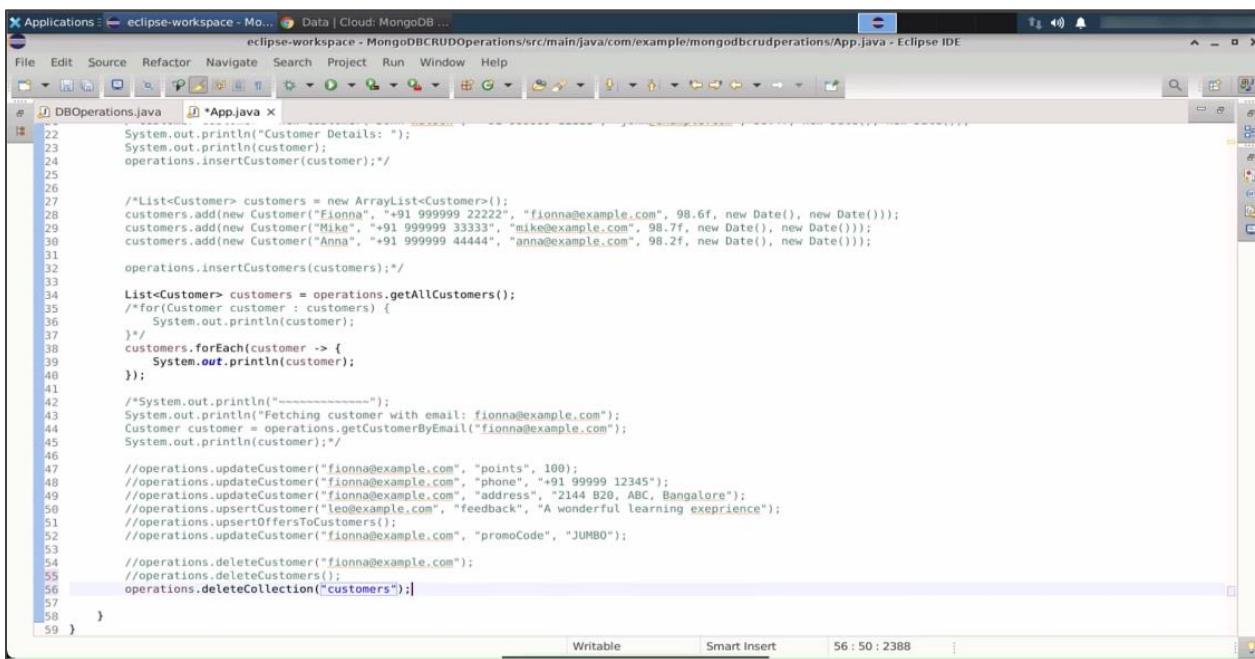


```

155     Bson filter = Filters.eq("email", email);
156     /*DeleteResult result = collection.deleteOne(filter);
157 
158     if(result.getDeletedCount() > 0) {
159         System.out.println("Document Deleted: "+result);
160     }else {
161         System.out.println("Document Not Found");
162     }*/
163 
164     Document deletedDocument = (Document) collection.findOneAndDelete(filter);
165     System.out.println(deletedDocument.toJson());
166 } catch (Exception e) {
167     System.out.println("Something went Wrong: "+e);
168 }
169 
170 public void deleteCustomers() {
171     try {
172         Bson filter = Filters.lte("points", 100);
173         DeleteResult result = collection.deleteMany(filter);
174         System.out.println("Result is: "+result);
175     } catch (Exception e) {
176         System.out.println("Something went Wrong: "+e);
177     }
178 }
179 
180 public void deleteCollection(String collectionName) {
181     try {
182         collection = database.getCollection(collectionName);
183         collection.drop();
184         System.out.println(collectionName+" has been dropped");
185     } catch (Exception e) {
186         System.out.println("Something went Wrong: "+e);
187     }
188 }
189 
190 
191 }
192 
```

Writable Smart Insert 185 : 65 : 6143

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



```

22     System.out.println("Customer Details: ");
23     System.out.println(customer);
24     operations.insertCustomer(customer);
25 
26 
27     /*List<Customer> customers = new ArrayList<Customer>();
28     customers.add(new Customer("Fionna", "+91 999999 2222", "fionna@example.com", 98.6f, new Date(), new Date());
29     customers.add(new Customer("Mike", "+91 999999 3333", "mike@example.com", 98.7f, new Date(), new Date()));
30     customers.add(new Customer("Anna", "+91 999999 4444", "anna@example.com", 98.2f, new Date(), new Date()));
31 
32     operations.insertCustomers(customers);*/
33 
34     List<Customer> customers = operations.getAllCustomers();
35     /*for(Customer customer : customers) {
36         System.out.println(customer);
37     }*/
38     customers.forEach(customer -> {
39         System.out.println(customer);
40     });
41 
42     /*System.out.println("-----");
43     System.out.println("Fetching customer with email: fionna@example.com");
44     Customer customer = operations.getCustomerByEmail("fionna@example.com");
45     System.out.println(customer);*/
46 
47     //operations.updateCustomer("fionna@example.com", "points", 100);
48     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     //operations.upsertOffersToCustomers();
52     //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
53 
54     //operations.deleteCustomer("fionna@example.com");
55     //operations.deleteCustomers();
56     operations.deleteCollection("customers");
57 
58 }
59 
```

Writable Smart Insert 56 : 50 : 2388

## 2.3 Save the file and run the code

```

    22     System.out.println("Customer Details: ");
    23     System.out.println(customer);
    24     operations.insertCustomer(customer);/*
    25
    26
    27     */
    28     List<Customer> customers = new ArrayList<Customer>();
    29     customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
    30     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
    31     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
    32
    33     operations.insertCustomers(customers);/*
    34
    35     */
    36     List<Customer> customers = operations.getAllCustomers();
    37     /*forCustomer customer : customers {
    38         System.out.println(customer);
    39     }*/
    40     customers.forEach(customer -> {
    41         System.out.println(customer);
    42     });
    43     /*System.out.println("-----");
    44     System.out.println("Fetching customer with email: fionna@example.com");
    45     Customer customer = operations.getCustomerByEmail("fionna@example.com");
    46     System.out.println(customer);*/
    47     //operations.updateCustomer("fionna@example.com", "points", 100);
    48     //operations.updateCustomer("fionna@example.com", "phone", "+91 999999 12345");
    49     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
    50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
    51     //operations.upsertOfferstoCustomers();
    52     //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
    53
    54     //operations.deleteCustomer("fionna@example.com");
    55     //operations.deleteCustomers();
    56     operations.deleteCollection("customers");
    57 }
    58
  
```

Writable Smart Insert 56 : 50 : 2388

```

  
```

Problems Javadoc Declaration Console

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspace/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 'org.mongodb.\*'  
[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, tempPhone=null]  
**customers has been dropped**

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

## 2.4 Go back to the database and refresh

The screenshot shows the MongoDB Cloud interface. On the left, the sidebar includes sections for DEPLOYMENT, DATABASES (Data Lake, estore), DATA SERVICES (Triggers, Data API PREVIEW), and SECURITY (Quickstart, Database Access, Network Access, Advanced). The main area displays the 'estore' database with the 'customers' collection selected. The collection size is 194B, total documents are 1, and indexes total size is 36KB. A query result table shows one document:

```

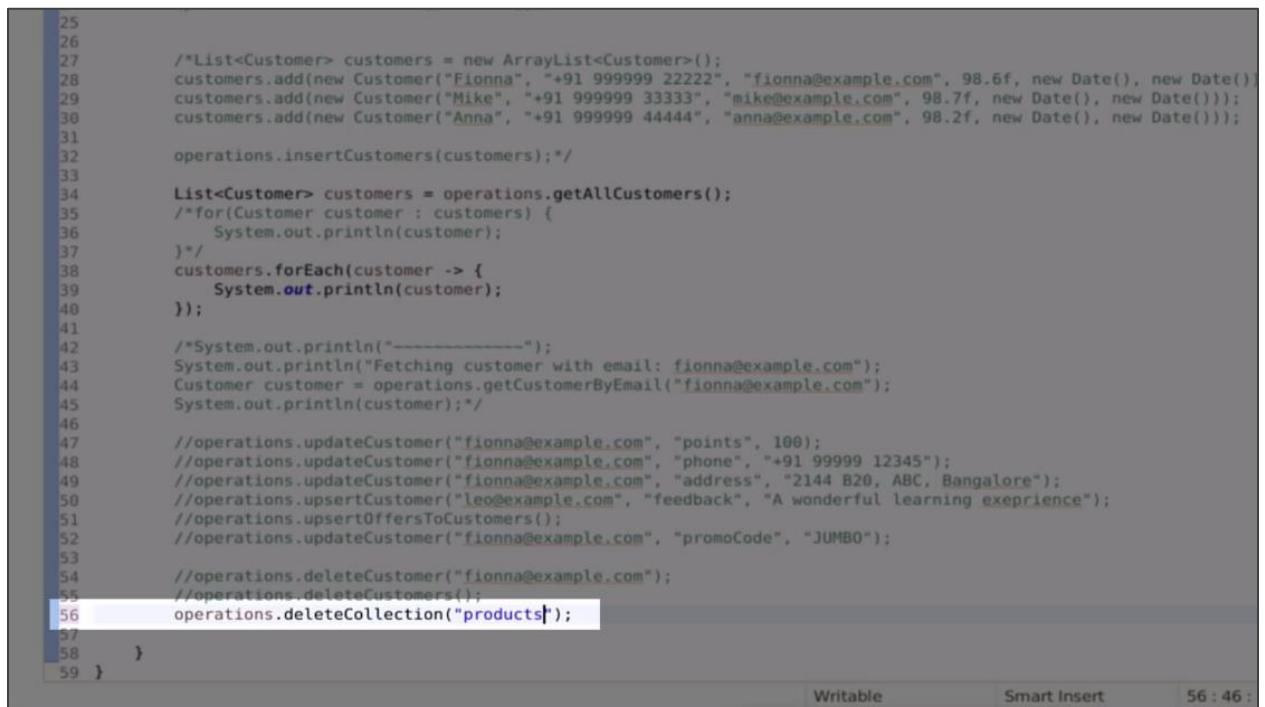
_id: ObjectId("61d41c988d05da6c1fa283eb")
name: "Anna"
phone: "+91 099999 44444"
email: "anna@example.com"
temperature: 98.19999694824219
intime: 2022-01-04T10:08:24.105+00:00
outtime: 2022-01-04T10:08:24.105+00:00
points: 120
> promoCode: Array
  
```

The screenshot shows the MongoDB Cloud interface. The sidebar is identical to the previous screenshot. The main area shows the 'Cluster0' collections page. The 'Collections' tab is selected, showing 9 databases and 23 collections. The 'estore' database is expanded, and the 'products' collection is highlighted with an orange border. The table below lists the collection details:

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

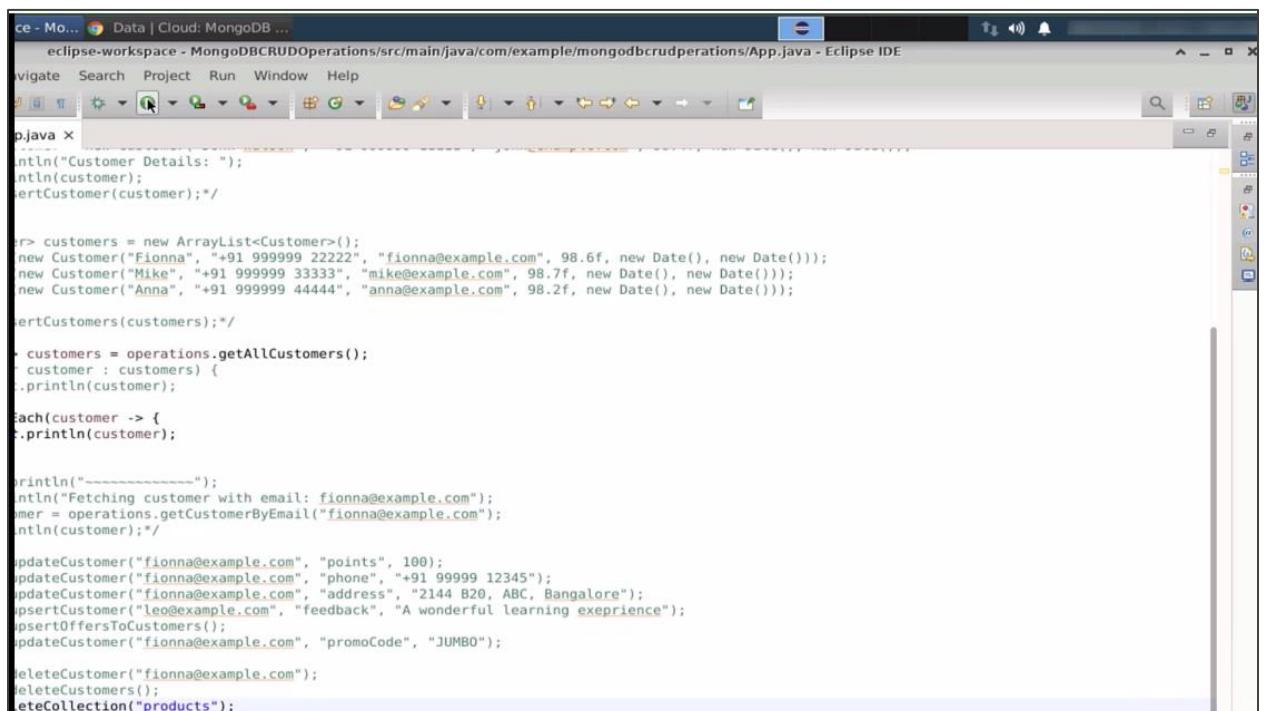


```

25
26
27     /*List<Customer> customers = new ArrayList<Customer>();
28     customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date());
29     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
30     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
31
32     operations.insertCustomers(customers);*/
33
34     List<Customer> customers = operations.getAllCustomers();
35     /*for(Customer customer : customers) {
36         System.out.println(customer);
37     }*/
38     customers.forEach(customer -> {
39         System.out.println(customer);
40     });
41
42     /*System.out.println("~~~~~");
43     System.out.println("Fetching customer with email: fionna@example.com");
44     Customer customer = operations.getCustomerByEmail("fionna@example.com");
45     System.out.println(customer);*/
46
47     //operations.updateCustomer("fionna@example.com", "points", 100);
48     //operations.updateCustomer("fionna@example.com", "phone", "+91 999999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     //operations.upsertOffersToCustomers();
52     //operations.updateCustomer("fionna@example.com", "promoCode", "JUMBO");
53
54     //operations.deleteCustomer("fionna@example.com");
55     //operations.deleteCustomers();
56     operations.deleteCollection("products");
57
58 }
59 }
```

Writable Smart Insert 56 : 46 :

## 2.6 Rerun the code



```

eclipse - Mo... Data | Cloud: MongoDB ...
eclipse-workspace - MongoDBCRUDOperations/src/main/java/com/example/mongodbcrudoperations/App.java - Eclipse IDE
File Edit Navigate Search Project Run Window Help
p.java X
intln("Customer Details: ");
intln(customer);
sertCustomer(customer);*/
*er> 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());
sertCustomers(customers);*/
* customers = operations.getAllCustomers();
* customer : customers) {
*.println(customer);
Each(customer -> {
*.println(customer);

println("~~~~~");
intln("Fetching customer with email: fionna@example.com");
mer = operations.getCustomerByEmail("fionna@example.com");
intln(customer);*/
updateCustomer("fionna@example.com", "points", 100);
updateCustomer("fionna@example.com", "phone", "+91 999999 12345");
updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
upsertOffersToCustomers();
updateCustomer("fionna@example.com", "promoCode", "JUMBO");
deleteCustomer("fionna@example.com");
deleteCustomers();
eteCollection("products");
```

Eclipse IDE screenshot showing Java code and the Console output.

```

Run Window Help
Problems Javadoc Declaration Console X
<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hots
MongoDB CRUD Operations App
Jan 05, 2022 11:11:26 AM com.mongodb.diagnostics.logging.Lloggers 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
);

s.getAllCustomers();
{

};

r with email: fionna@example.com");
stomerByEmail("fionna@example.com");

example.com", "points", 100);
example.com", "phone", "+91 99999 12345");
example.com", "address", "2144 B20, ABC, Bangalore");
mple.com", "feedback", "A wonderful learning exeprience");
();
example.com", "promoCode", "JUMBO");

```

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

## 2.7 Go back to the database and refresh

MongoDB Atlas screenshot showing the 'estore' database in the 'Collections' tab.

**Project 0 - Test's Org - 2022-01-03 > PROJECT 0 > DATABASES**

**Cluster0**

**Collections**

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.

The screenshot shows the MongoDB Cloud interface. On the left, there's a sidebar with 'Project 0' selected. Under 'DEPLOYMENT', 'Databases' is highlighted. The main area shows 'Cluster0' with 'TESTS ORG - 2022-01-03 > PROJECT 0 > DATABASES'. It lists 'DATABASES: 9' and 'COLLECTIONS: 22'. The 'Collections' tab is active. A table shows collection details:

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
users	4	870B	21BB	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.