

Lesson 01 Demo 07

Updating Documents Using MongoDB CRUD Operations

Objective: To work with the CRUD operations and update the documents for existing and non-existing customers

Tools required: Eclipse IDE

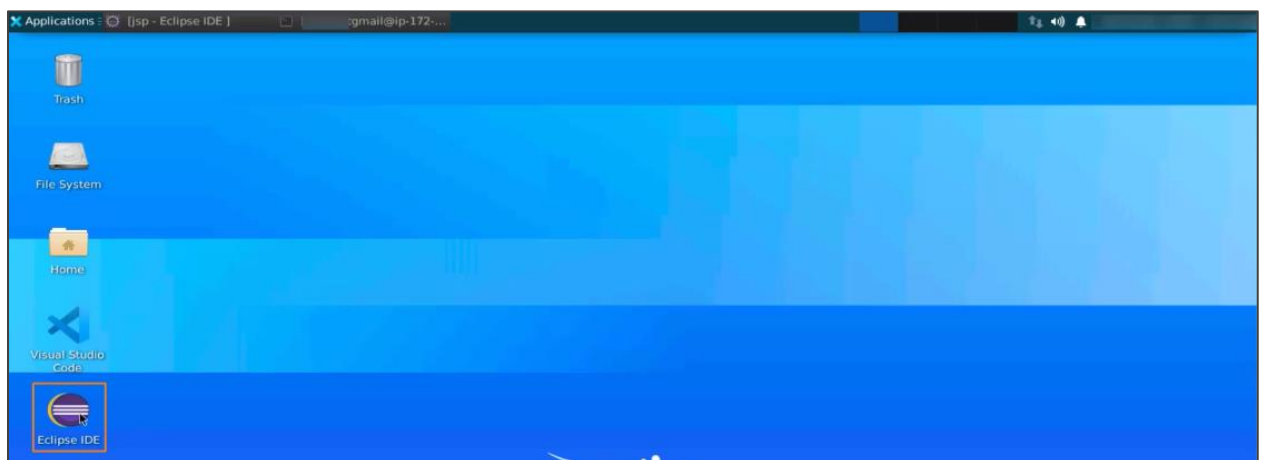
Prerequisites: None

Steps to be followed:

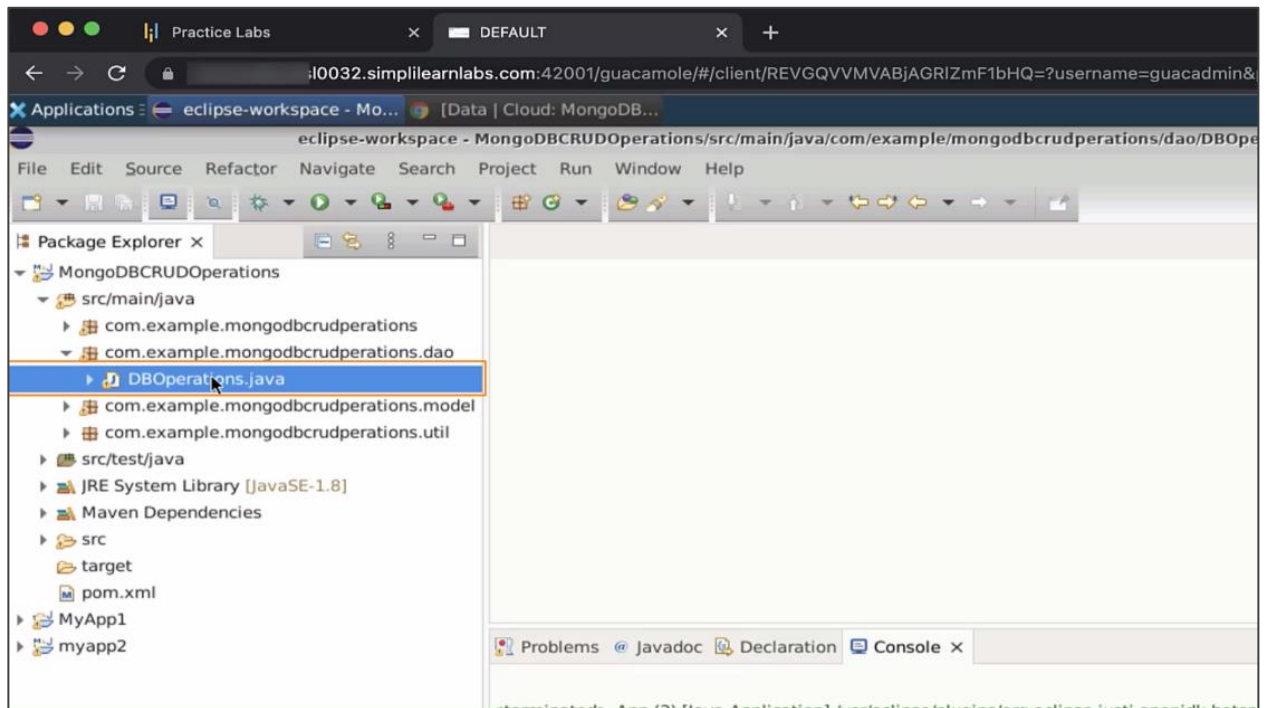
1. Update customer operations
2. Add try catch block
3. Execute another operation for an update
4. Create new function
5. Create more functions
6. Write a new method

Step 1: Update customer operations

1.1 Open the Eclipse IDE



1.2 Navigate to the project **MongoDBCRUDOperations** and open the **DBOperations.java** file



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

1.3 Write a new method to update the customer and add inputs to the method

```

73      customer.setEmail(doc.get("email").toString());
74
75      return customer;
76  }
77
78  public List<Customer> getAllCustomers(){
79      List<Customer> customers = new ArrayList<Customer>();
80      try {
81
82          List<Document> documents = (List<Document>) collection
83              .find()
84              .sort(Sorts.ascending("name"))
85              .into(new ArrayList<Document>());
86          for(Document doc : documents) {
87              customers.add(convertDocumentToCustomer(doc));
88          }
89      } catch (Exception e) {
90          System.out.println("Something went Wrong: "+e);
91      }
92      return customers;
93  }
94
95
96  public Customer getCustomerByEmail(String email) {
97      /*Document filter = new Document("email", email);
98      Document document = (Document) collection.find(filter).first();*/
99      Document document = (Document) collection.find(Filters.eq("email", email)).first();
100      Customer customer = convertDocumentToCustomer(document);
101      return customer;
102  }
103
104
105  public void updateCustomer(String email, String key, String value) {
106      Bson filter = Filters.eq("email", email);
107      Bson updateOperation = Updates
108
109  }
110
111

```

1.4 Select the **set** method to set the key-value pair to filter the data later

```

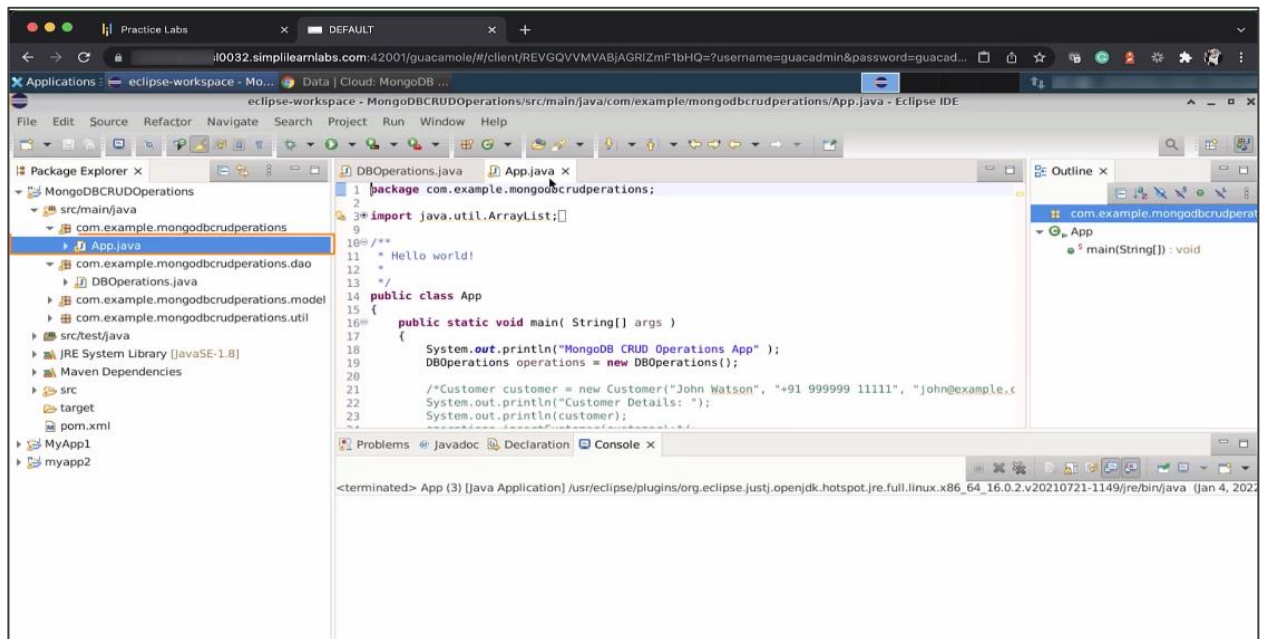
74      return customer;
75
76  }
77
78  public List<Customer> getAllCustomers(){
79      List<Customer> customers = new ArrayList<Customer>();
80      try {
81
82          List<Document> documents = (List<Document>) collection
83              .find()
84              .sort(Sorts.ascending("name"))
85              .into(new ArrayList<Document>());
86          for(Document doc : documents) {
87              customers.add(convertDocumentToCustomer(doc));
88          }
89      } catch (Exception e) {
90          System.out.println("Something went Wrong: "+e);
91      }
92      return customers;
93  }
94
95
96  public Customer getCustomerByEmail(String email) {
97      /*Document filter = new Document("email", email);
98      Document document = (Document) collection.find(filter).first();*/
99      Document document = (Document) collection.find(Filters.eq("email", email)).first();
100      Customer customer = convertDocumentToCustomer(document);
101      return customer;
102  }
103
104
105  public void updateCustomer(String email, String key, String value) {
106      Bson filter = Filters.eq("email", email);
107      Bson updateOperation = Updates.set
108
109  }
110
111

```

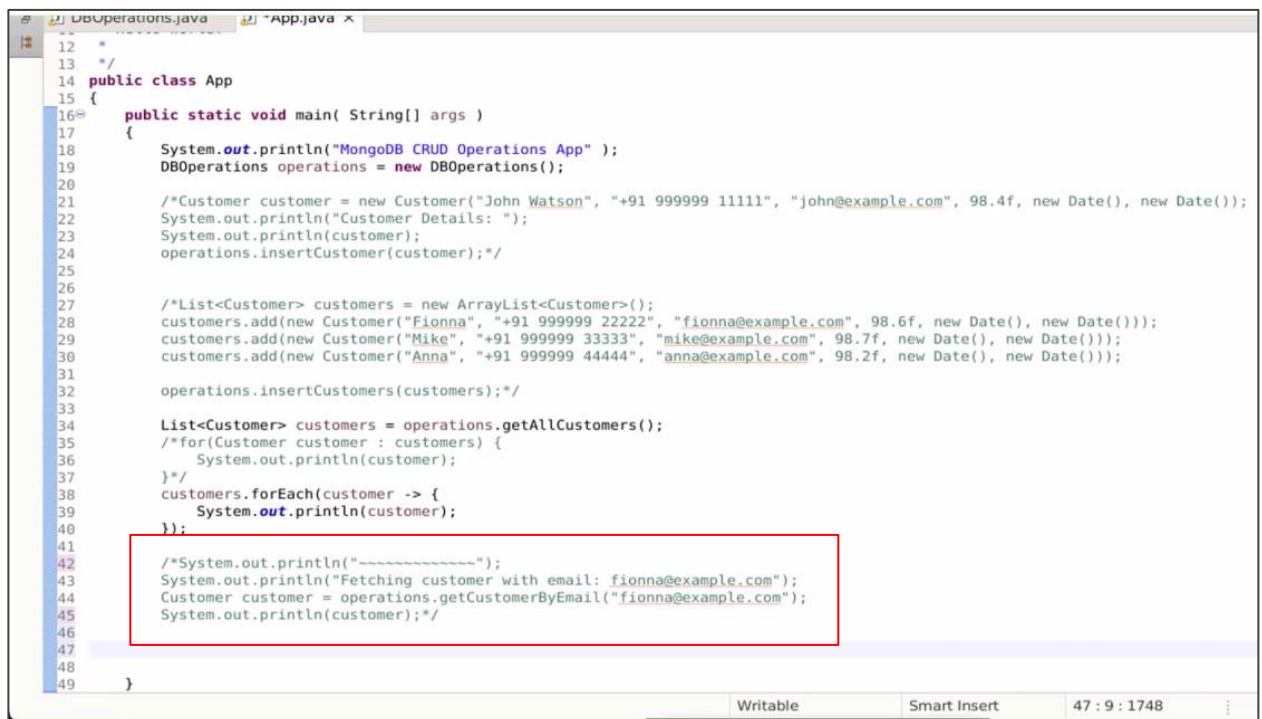
set(String fieldName, Titem value) : Bson - Updates
setOnInsert(Bson value) : Bson - Updates
setOnInsert(String fieldName, Titem value) : Bson - Updates
addEachToSet(String fieldName, List<Titem> values)
addToSet(String fieldName, Titem value) : Bson - Updates
unset(String fieldName) : Bson - Updates

Creates an update that sets the value of the field with the given name to the given value.
Type Parameters:
<Titem> the value type
Parameters:
fieldName the non-null field name
value the value, which may be null
Returns:
 the update
[@mongodb.driver.manual](#)
[reference/operator/update/set/ \\$set](#)

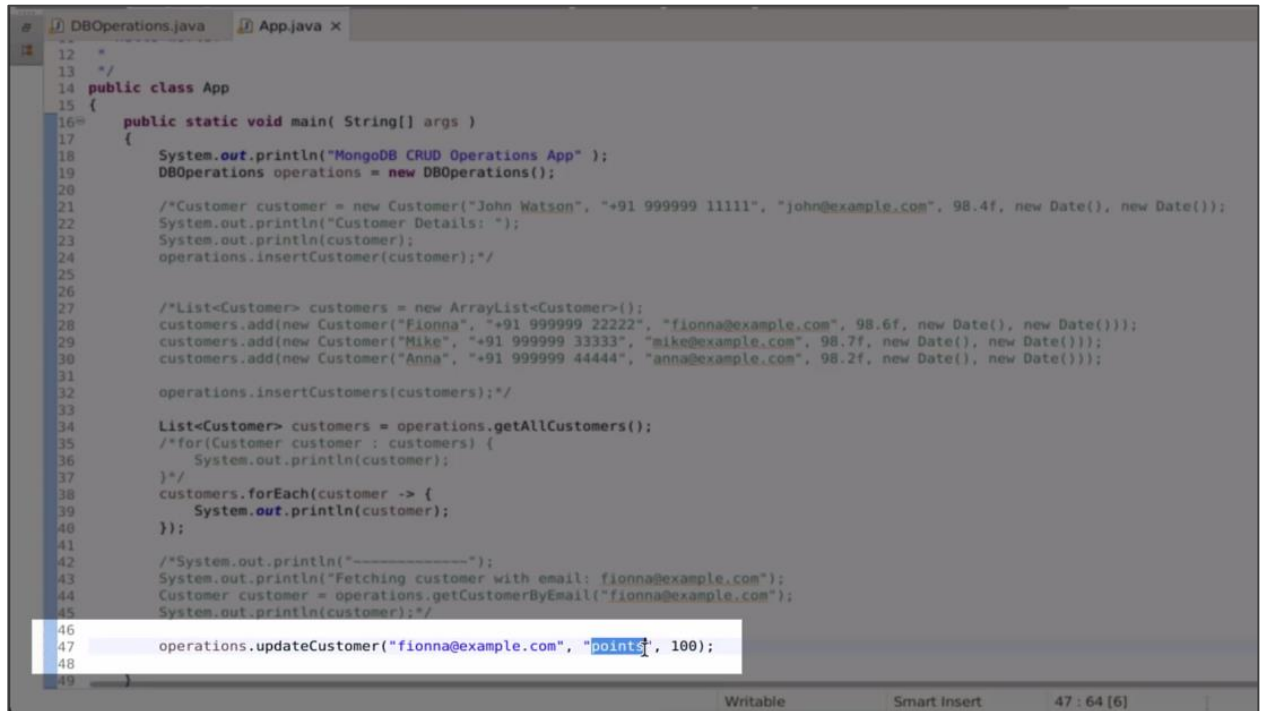
1.5 Navigate back to the App.java file



1.6 Comment out the print statements in the code



1.7 Use the **updateCustomer** function to update the details of the customer. Note that the key is **points** and the value is **100**



```
12  *
13  */
14  public class App
15  {
16      public static void main( String[] args )
17      {
18          System.out.println("MongoDB CRUD Operations App" );
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
49      }
```

Writable Smart Insert 47 : 64 [6]

1.8 Go back to the **DBOperations.java**, add a collection and execute the **updateOne** function to perform a single update

```

74     return customer;
75 }
76
77 public List<Customer> getAllCustomers(){
78     List<Customer> customers = new ArrayList<Customer>();
79     try {
80
81         List<Document> documents = (List<Document>) collection
82             .find()
83             .sort(Sorts.ascending("name"))
84             .into(new ArrayList<Document>());
85         for(Document doc : documents) {
86             customers.add(convertDocumentToCustomer(doc));
87         }
88     } catch (Exception e) {
89         System.out.println("Something went Wrong: "+e);
90     }
91     return customers;
92 }
93
94 public Customer
95     /*Document
96     Document doc
97     Customer cu
98     return cust
99
100
101
102
103
104
105 public void upd
106     Bson filter
107     Bson update
108     collection.updateOne
109
110
111 }

```

updateOne(Bson filter, Bson update) : UpdateResult
 Update a single document in the collection according to the specified arguments.

updateOne(Bson filter, List update) : UpdateResult
 Use this method to only update the corresponding fields in the document according to the update operators used in the update document. To replace the entire document with a new document, use the corresponding [replaceOne\(Bson, Object\)](#) method.

updateOne(Bson filter, List update, UpdateOptions updateOptions) : UpdateResult
 Use this method to only update the corresponding fields in the document according to the update operators used in the update document. To replace the entire document with a new document, use the corresponding [replaceOne\(Bson, Object\)](#) method.

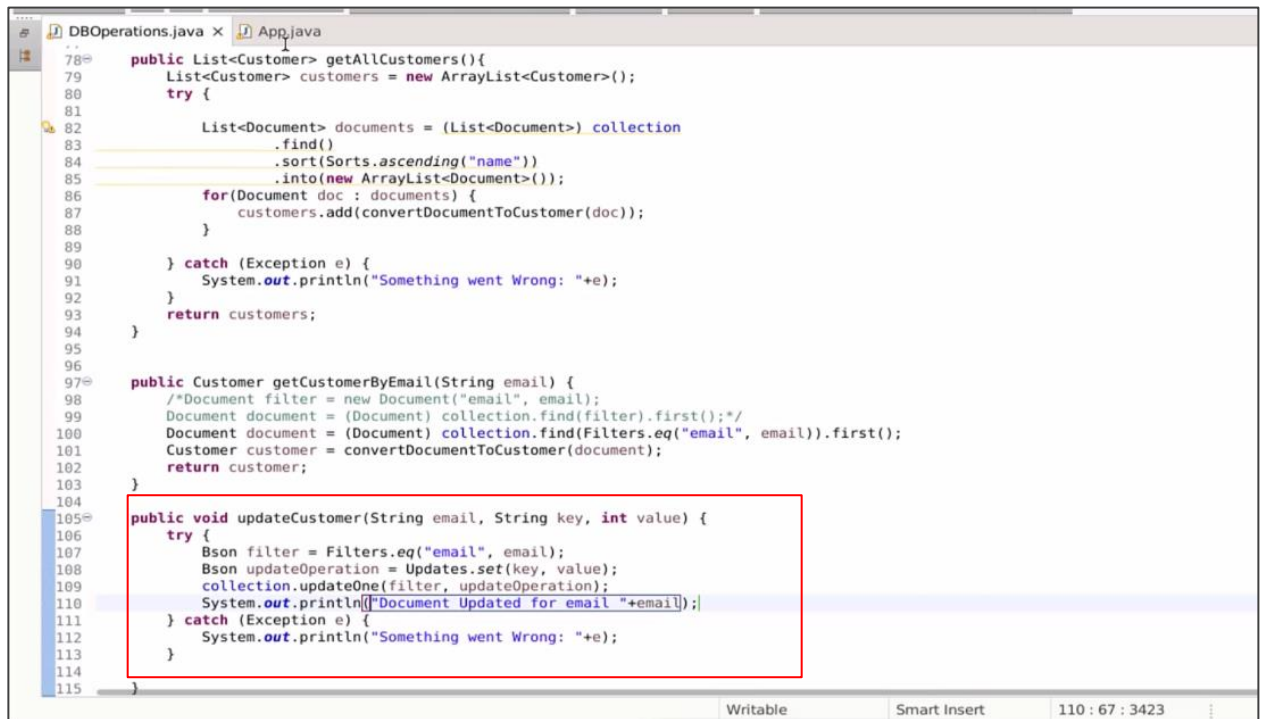
updateOne(ClientSession clientSession, Bson filter, Bson update) : UpdateResult
 Note: Supports retryable writes on MongoDB server versions 3.6 or higher when the retryWrites setting is enabled.

updateOne(ClientSession clientSession, Bson filter, List update) : UpdateResult
 Note: Supports retryable writes on MongoDB server versions 3.6 or higher when the retryWrites setting is enabled.

Parameters:
 filter a document describing the query filter, which

Step 2: Add try catch block

2.1 Add a try catch block in the **DBOperations.java** file for the update operation



```
DBOperations.java x App.java
78 public List<Customer> getAllCustomers(){
79     List<Customer> customers = new ArrayList<Customer>();
80     try {
81
82         List<Document> documents = (List<Document>) collection
83             .find()
84             .sort(Sorts.ascending("name"))
85             .into(new ArrayList<Document>());
86         for(Document doc : documents) {
87             customers.add(convertDocumentToCustomer(doc));
88         }
89     } catch (Exception e) {
90         System.out.println("Something went Wrong: "+e);
91     }
92     return customers;
93 }
94
95
96
97 public Customer getCustomerByEmail(String email) {
98     /*Document filter = new Document("email", email);
99     Document document = (Document) collection.find(filter).first();*/
100    Document document = (Document) collection.find(Filters.eq("email", email)).first();
101    Customer customer = convertDocumentToCustomer(document);
102    return customer;
103 }
104
105 public void updateCustomer(String email, String key, int value) {
106     try {
107         Bson filter = Filters.eq("email", email);
108         Bson updateOperation = Updates.set(key, value);
109         collection.updateOne(filter, updateOperation);
110         System.out.println("Document Updated for email "+email);
111     } catch (Exception e) {
112         System.out.println("Something went Wrong: "+e);
113     }
114 }
115 }
```

Writable Smart Insert 110 : 67 : 3423

2.2 Go back to **App.java** and run the code

```

12  *
13  */
14  public class App
15  {
16      public static void main( String[] args )
17      {
18          System.out.println("MongoDB CRUD Operations App");
19          DBOperations operations = new DBOperations();
20
21          /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22          System.out.println("Customer Details: ");
23          operations.insertCustomer(customer);*/
24
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      }
49  }

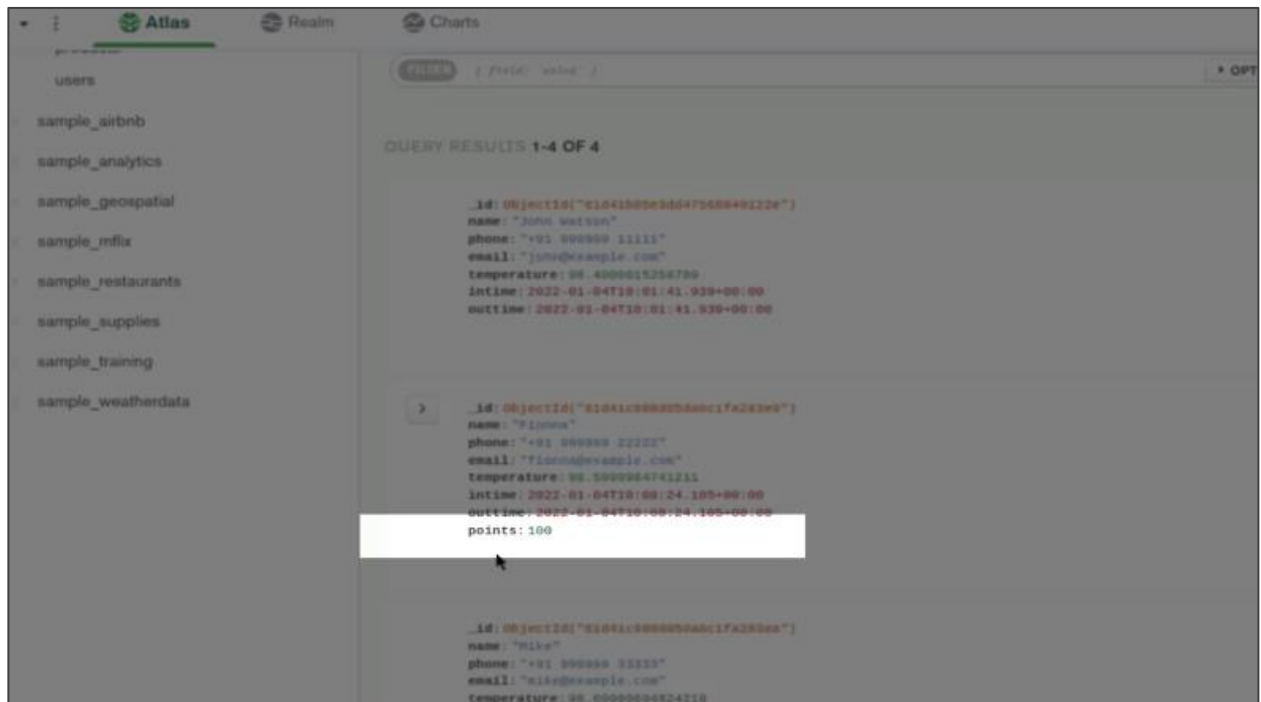
```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 04, 2022 3:18:50 PM com.mongodb.diagnostics.logging.Loggers shouldUseSLF4J
WARNING: SLF4J not found on the classpath. Logging is disabled for the org.mongodb.driver
[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, temperature=98.2f, points=0]
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, temperature=98.6f, points=0]
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com, temperature=98.4f, points=0]
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, temperature=98.7f, points=0]
Document Updated for email fionna@example.com

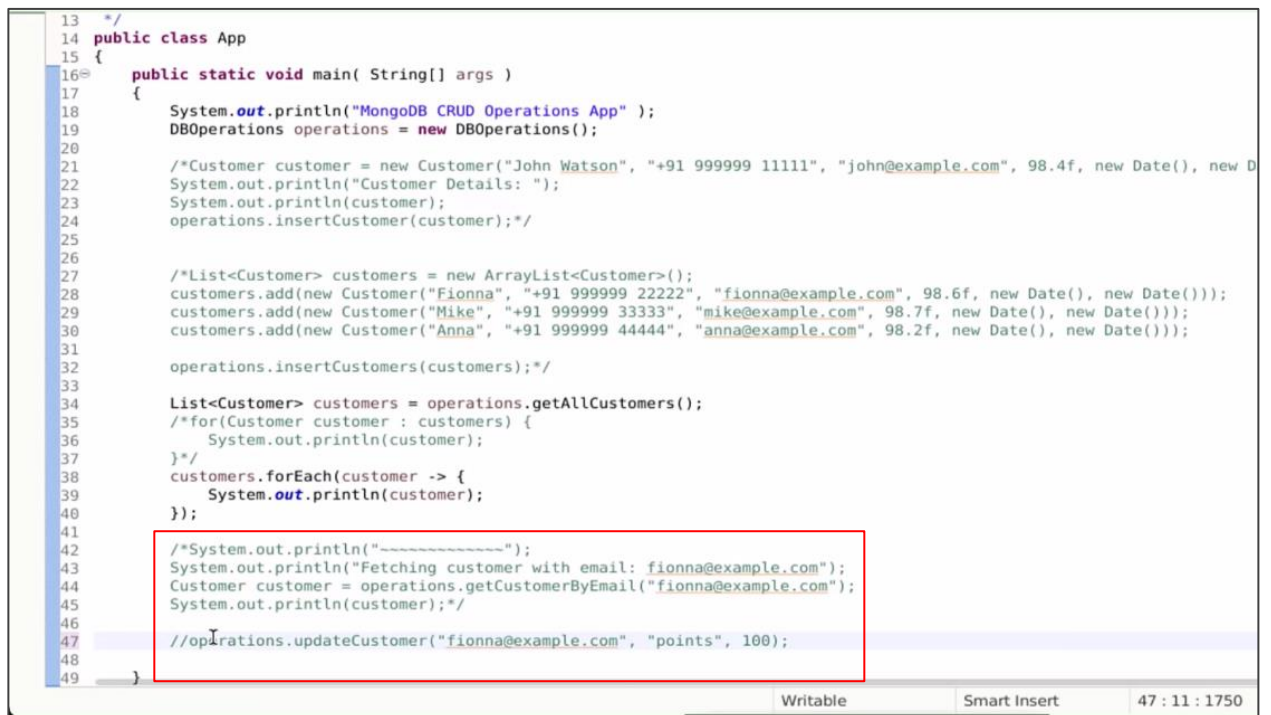
```

You can see the output as **Document Updated for email fionna@example.com.**



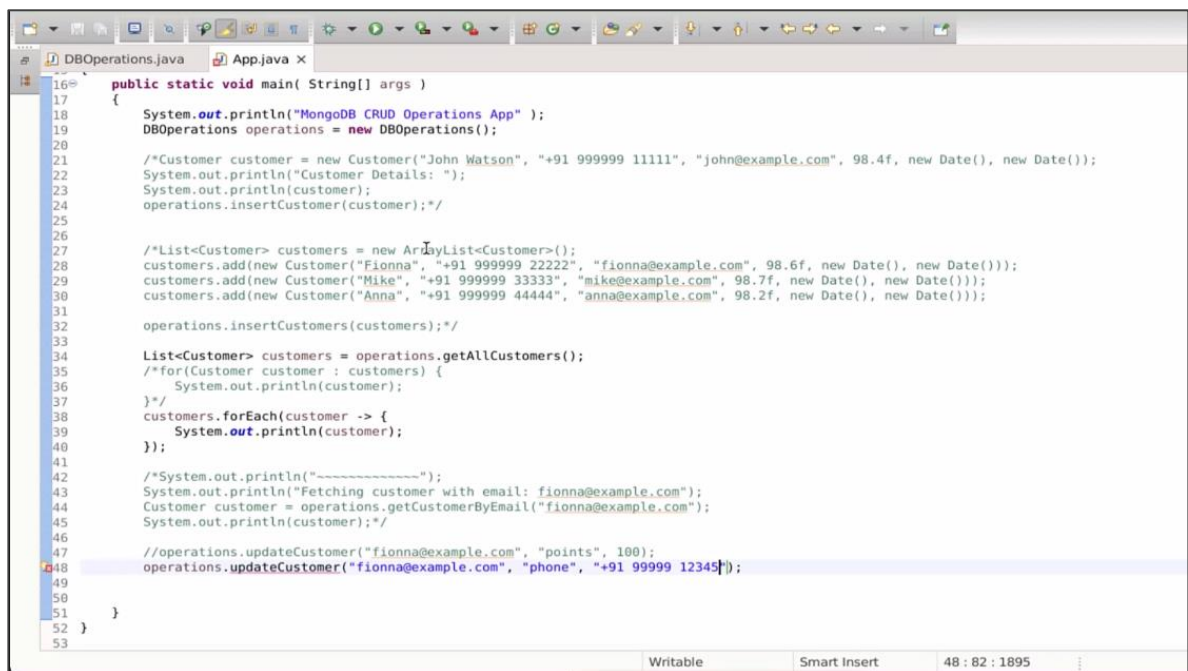
A new key-value pair will be added to the database.

2.3 Comment out the **updateCustomer** operation



Step 3: Execute another operation for an update

3.1 Write another operation for **updateCustomer** to update the phone number of the customer



```
16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App" );
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 99999 12345");
49
50 }
51
52 }
53
```

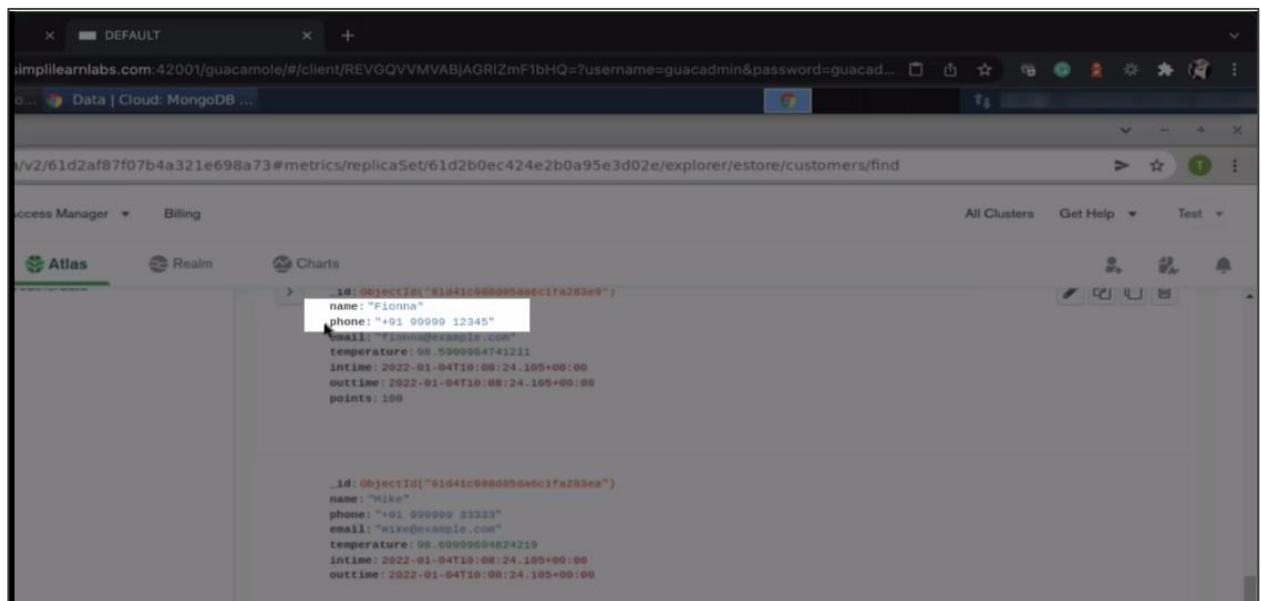

You can see the output as **Document Updated** for email **fionna@example.com**.

The screenshot shows the Eclipse IDE with a Java project named 'MongoDB CRUD Operations'. The main class is 'App.java'. The code includes a 'main' method that initializes a 'DBOperations' object and performs several database operations. The console output shows the following sequence of events:

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.just.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 04, 2022 3:28:30 PM com.mongodb.diagnostics.logging.Loggers shouldD...
WARNING: SLF4J not found on the classpath. Logging is disabled for the 's...
[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, te...
Customer [name=Fionna, phone=+91 999999 22222, email=fionna@example.com, t...
Customer [name=John Watson, phone=+91 999999 11111, email=john@example...
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, te...
Document Updated for email fionna@example.com
  
```

3.3 Go back and refresh the collection



The phone number for **Fionna** should now be updated.

3.4 Now, use another method **Updates.push** to see how you can update the data

```

79 List<Customer> customers = new ArrayList<Customer>();
80 try {
81     List<Document> documents = (List<Document>) collection
82         .find()
83         .sort(Sorts.ascending("name"))
84         .into(new ArrayList<Document>());
85     for(Document doc : documents) {
86         customers.add(convertDocumentToCustomer(doc));
87     }
88 } catch (Exception e) {
89     System.out.println("Something went Wrong: "+e);
90 }
91 return customers;
92 }
93
94
95
96
97 public Customer getCustomerByEmail(String email) {
98     /*Document filter = new Document("email", email);
99     Document document = (Document) collection.find(filter).first();*/
100    Document document = (Document) collection.find(Filters.eq("email", email)).first();
101    Customer customer = convertDocumentToCustomer(document);
102    return customer;
103 }
104
105 public void updateCustomer(String email, String key, String value) {
106     try {
107         Bson filter = Filters.eq("email", email);
108         //Bson updateOperation = Updates.set(key, value);
109         Bson updateOperation = Updates.push(key, value);
110         collection.updateOne(filter, updateOperation);
111         System.out.println("Document Updated for email "+email);
112     } catch (Exception e) {
113         System.out.println("Something went Wrong: "+e);
114     }
115 }

```

3.5 Go back to **App.java** and comment out the earlier part of the code

```

17
18 System.out.println("MongoDB CRUD Operations App");
19 DBOperations operations = new DBOperations();
20
21 /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 99999 12345");
49
50
51 }
52 }
53

```


3.6 Use the `updateCustomer` method and enter a new address, for example, **2144 B20, ABC, Bangalore**

```

17
18 System.out.println("MongoDB CRUD Operations App ");
19 DBOperations operations = new DBOperations();
20
21 /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 99999 12345");
49 operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50
51
52 }
53 }

```

3.7 Save the file and run the code

```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App ");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 99999 12345");
49     operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50
51
52 }
53 }

```



```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com");
22     System.out.println("Customer Details: ");
23     System.out.println(customer);
24     operations.insertCustomer(customer);*/
25
26     /*List<Customer> customers = new ArrayList<Customer>();
27     customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.4f, null));
28     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, null));
29     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, null));
30     operations.insertCustomers(customers);*/
31
32     List<Customer> customers = operations.getAllCustomers();
33     /*for(Customer customer : customers) {
34         System.out.println(customer);
35     }*/
36     customers.forEach(customer -> {
37         System.out.println(customer);
38     });
39
40     /*System.out.println("-----");
41     System.out.println("Fetching customer with email: fionna@example.com");
42     Customer customer = operations.getCustomerByEmail("fionna@example.com");
43     System.out.println(customer);*/
44
45     //operations.updateCustomer("fionna@example.com", "points", 100);
46     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
47     operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
48
49 }
50
51
52
53

```

Console Output:

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.just.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 04, 2022 3:22:03 PM com.mongodb.diagnostics.logging.Logger shouldUseS...
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
Document Updated for email fionna@example.com

```

You can see the output as **Document Updated for email fionna@example.com**.

3.8 Go back to the **customers** collection and refresh the document list

Database: eStore

Collection: customers

Indexes: Schema Anti-Patterns Aggregation Search Index

FILTER { field: 'value' }

QUERY RESULTS 1-4 OF 4

```

{
  "_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
}

```

```

{
  "_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
}

```

customers

products

users

sample_airbnb

sample_analytics

sample_geospatial

sample_mflix

sample_restaurants

sample_supplies

sample_training

sample_weatherdata

FILTER { field: 'value' }

QUERY RESULTS 1-4 OF 4

_id: ObjectId("61d41b05e3dd47568840122e")

name: "John Watson"

phone: "+91 999999 1111"

email: "john@example.com"

temperature: 98.4800015258789

intime: 2022-01-04T10:01:41.939+00:00

outtime: 2022-01-04T10:01:41.939+00:00

>

_id: ObjectId("61d41c988d05da6c1fa283e9")

name: "Fiona"

phone: "+91 99999 12345"

email: "fiona@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

0: "2144 B20, ABC, Bangalore"

The push operation should have created an array of addresses.

Step 4: Create a new function

- 4.1 In the **DBOperations.java** file, create a new function called **upsertCustomer()** to update new records in the document

```
88     }
89
90     } catch (Exception e) {
91         System.out.println("Something went Wrong: "+e);
92     }
93     return customers;
94 }
95
96
97 public Customer getCustomerByEmail(String email) {
98     /*Document filter = new Document("email", email);
99     Document document = (Document) collection.find(filter).first();*/
100    Document document = (Document) collection.find(Filters.eq("email", email)).first();
101    Customer customer = convertDocumentToCustomer(document);
102    return customer;
103 }
104
105 public void updateCustomer(String email, String key, String value) {
106     try {
107         Bson filter = Filters.eq("email", email);
108         //Bson updateOperation = Updates.set(key, value);
109         Bson updateOperation = Updates.push(key, value);
110         collection.updateOne(filter, updateOperation);
111         System.out.println("Document Updated for email "+email);
112     } catch (Exception e) {
113         System.out.println("Something went Wrong: "+e);
114     }
115 }
116
117
118 public void upsertCustomer(String email, String key, String value) {
119 }
120 }
121
122 }
123
```

Writable Smart Insert 118 : 70 : 36

4.2 Create a try catch block, perform a push operation on the key-value pair, and use the **upsert** function to pass the value as **true**

```
99 public Customer getCustomerByEmail(String email) {
100     /*Document filter = new Document("email", email);
101     Document document = (Document) collection.find(filter).first();*/
102     Document document = (Document) collection.find(Filters.eq("email", email)).first();
103     Customer customer = convertDocumentToCustomer(document);
104     return customer;
105 }
106
107 public void updateCustomer(String email, String key, String value) {
108     try {
109         Bson filter = Filters.eq("email", email);
110         //Bson updateOperation = Updates.set(key, value);
111         Bson updateOperation = Updates.push(key, value);
112         collection.updateOne(filter, updateOperation);
113         System.out.println("Document Updated for email "+email);
114     } catch (Exception e) {
115         System.out.println("Something went Wrong: "+e);
116     }
117 }
118
119
120 public void upsertCustomer(String email, String key, String value) {
121     try {
122         Bson filter = Filters.eq("email", email);
123         Bson updateOperation = Updates.push(key, value);
124         UpdateOptions options = new UpdateOptions();
125         options.upsert(true);
126         //collection.updateOne(filter, updateOperation);
127         UpdateResult result = collection.updateOne(filter, updateOperation, options);
128         System.out.println("Result is: "+result);
129         System.out.println("Document Updated for email "+email);
130     } catch (Exception e) {
131         System.out.println("Something went Wrong: "+e);
132     }
133 }
134
135 }
```

Writable Smart Insert 128 : 52 : 4089

4.3 Go back to **App.java**, use the operation **upsertCustomer(String email, String key, String value)** to update an existing document

```

19  operations = new Operations();
20
21  /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, 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 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 99999 12345");
49  //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50  operations.upsertCustomer("mike@example.com", "feedback", "A wonderful learning");
51
52  }
53

```

4.4 Add inputs to the key-value pair as **feedback** and **A wonderful learning**

```

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 99999 12345");
49  //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
50  operations.upsertCustomer("mike@example.com", "feedback", "A wonderful learning");
51
52  }
53

```

4.5 Save and run the code

```

16= public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App" );
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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     /*List<Customer> customers = new ArrayList<Customer>();
27     customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
28     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
29     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
30
31     operations.insertCustomers(customers);*/
32
33     List<Customer> customers = operations.getAllCustomers();
34     /*for(Customer customer : customers) {
35         System.out.println(customer);
36     }*/
37     customers.forEach(customer -> {
38         System.out.println(customer);
39     });
40
41     /*System.out.println("-----");
42     System.out.println("Fetching customer with email: fionna@example.com");
43     Customer customer = operations.getCustomerByEmail("fionna@example.com");
44     System.out.println(customer);*/
45
46     //operations.updateCustomer("fionna@example.com", "points", 100);
47     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
48     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
49     operations.upsertCustomer("mike@example.com", "feedback", "A wonderful learning");
50
51
52
53

```

```

16= public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App" );
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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     /*List<Customer> customers = new ArrayList<Customer>();
27     customers.add(new Customer("Fionna", "+91 999999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
28     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
29     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, new Date(), new Date()));
30
31     operations.insertCustomers(customers);*/
32
33     List<Customer> customers = operations.getAllCustomers();
34     /*for(Customer customer : customers) {
35         System.out.println(customer);
36     }*/
37     customers.forEach(customer -> {
38         System.out.println(customer);
39     });
40
41     /*System.out.println("-----");
42     System.out.println("Fetching customer with email: fionna@example.com");
43     Customer customer = operations.getCustomerByEmail("fionna@example.com");
44     System.out.println(customer);*/
45
46     //operations.updateCustomer("fionna@example.com", "points", 100);
47     //operations.updateCustomer("fionna@example.com", "phone", "+91 99999 12345");
48     //operations.updateCustomer("fionna@example.com", "address", "2144 B20, ABC, Bangalore");
49     operations.upsertCustomer("mike@example.com", "feedback", "A wonderful learning");
50
51
52
53

```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java:
MongoDB CRUD Operations App
Jan 05, 2022 10:37:55 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=Fionna, phone=+91 999999 44444, email=anna@example.com, tempe
Customer [name=Fionna, phone=+91 999999 12345, email=fionna@example.com, te
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.com, te
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, tempe
Result is: AcknowledgedUpdateResult{matchedCount=1, modifiedCount=1, upser
Document Updated for email mike@example.com

```

You can see the output as **Document Updated for email mike@example.com**.


```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App" );
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com");
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 22222", "fionna@example.com", 98.6f, null));
29     customers.add(new Customer("Mike", "+91 999999 33333", "mike@example.com", 98.7f, null));
30     customers.add(new Customer("Anna", "+91 999999 44444", "anna@example.com", 98.2f, null));
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", "2148 820 ABC, Bangalore");
50     operations.upsertCustomer("mike@example.com", "feedback", "A wonderful learning");
51
52
53 }

```

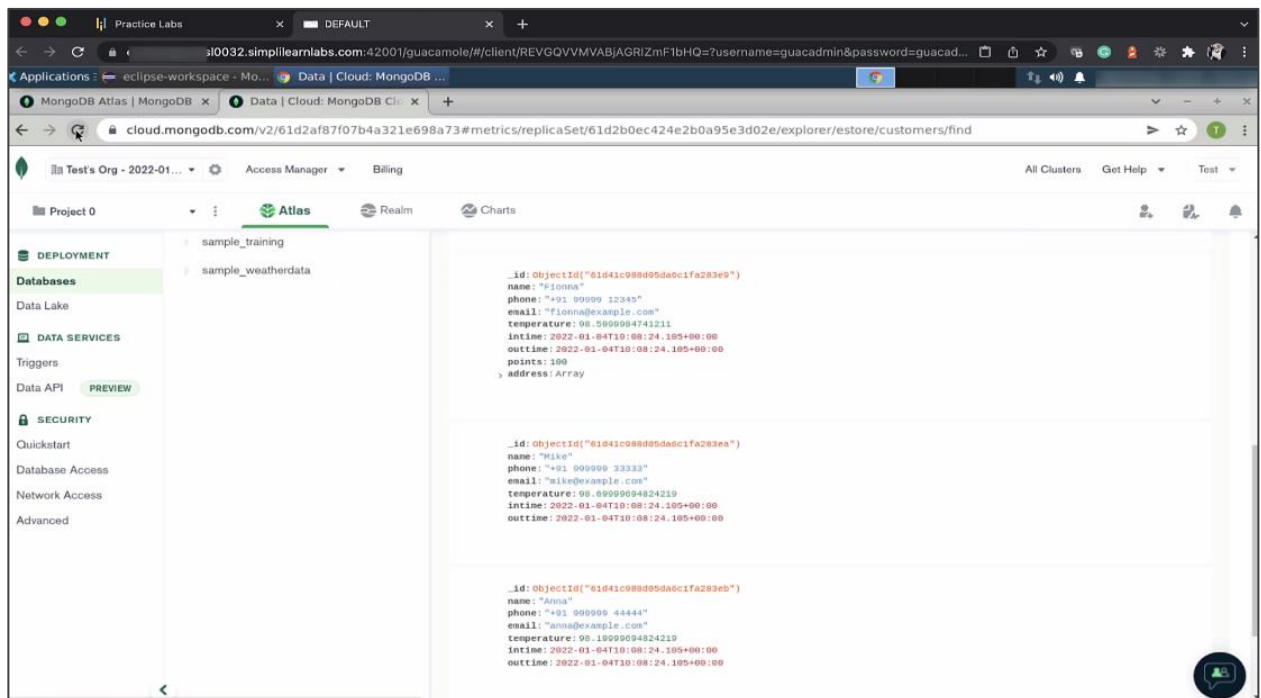
```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/
MongoDB CRUD Operations App
Jan 05, 2022 10:37:55 AM com.mongodb.diagnostics.logging.Loggers should be
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, te
Customer [name=Fionna, phone=+91 999999 12345, email=fionna@example.com, te
Customer [name=John Watson, phone=+91 999999 11111, email=john@example.co
Customer [name=Mike, phone=+91 999999 33333, email=mike@example.com, te
Result is: [{"_id":"61d41c98885da6c1fa28314","name":"mike","phone":"+91 999999 33333","email":"mike@example.com","temperature":98.6999994824219,"intime":2022-01-04T10:08:24.105+00:00,"outtime":2022-01-04T10:08:24.105+00:00,"address":null,"feedback":"A wonderful learning"}]
Document Updated for email mike@example.com

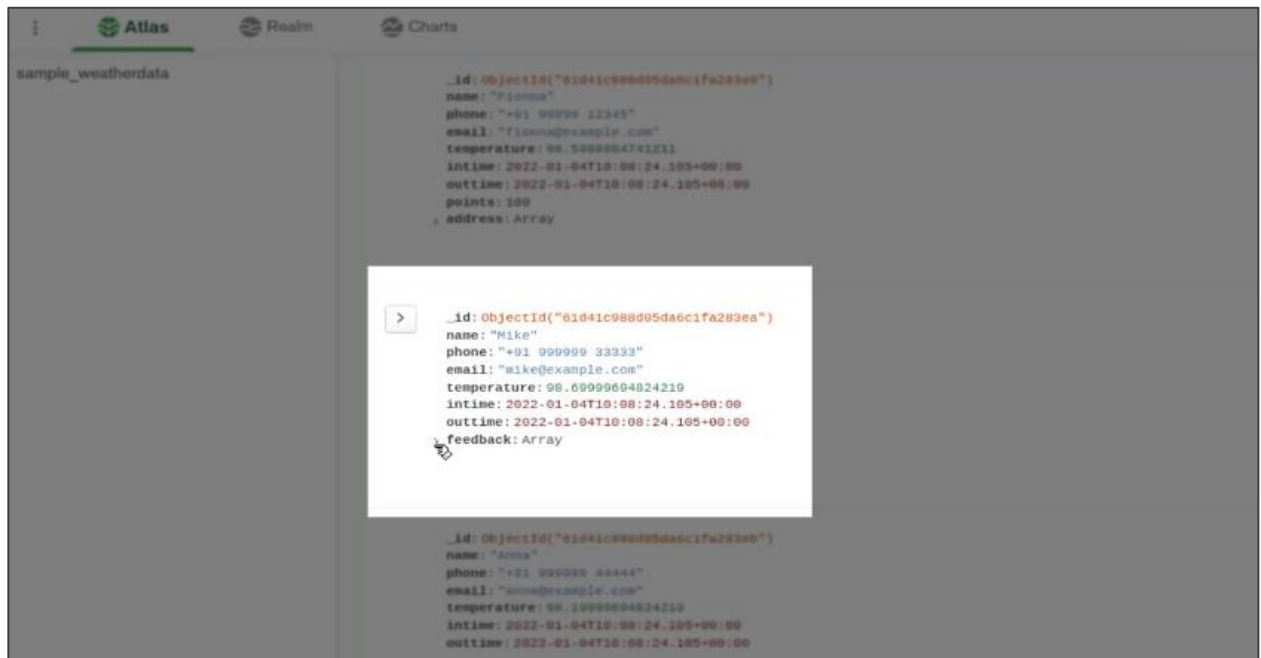
```

You can see the results state as **AcknowledgeUpdateResult**.

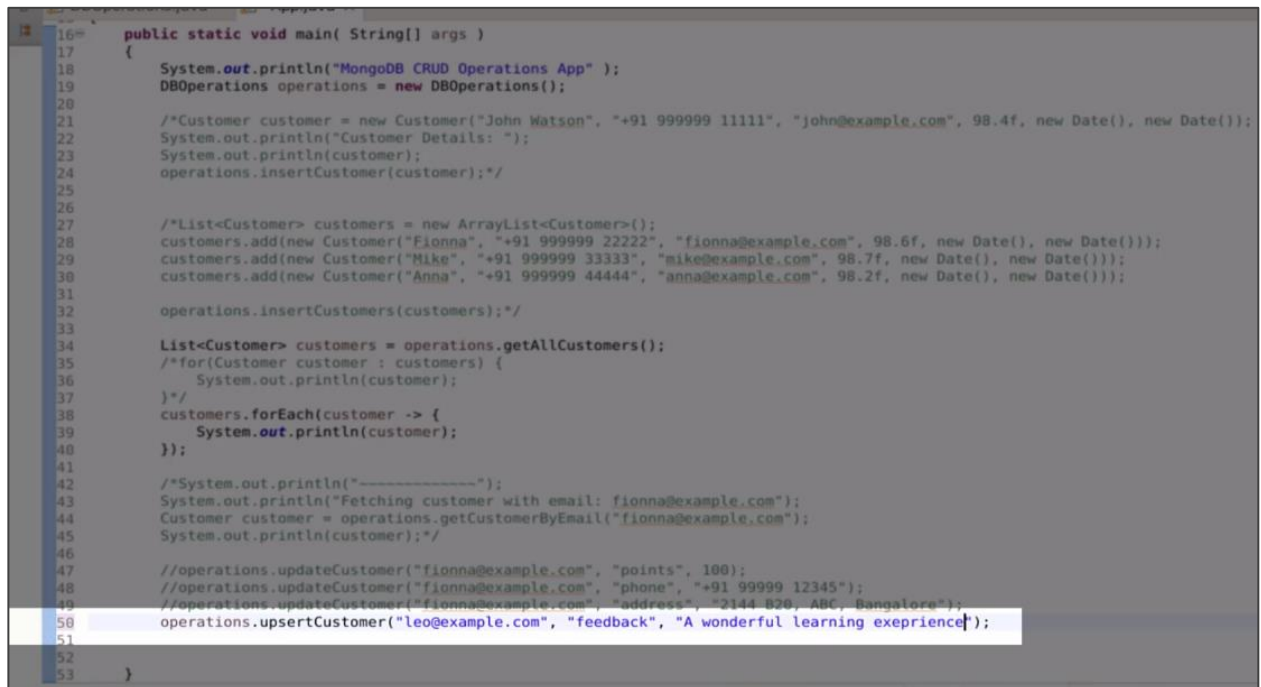
4.6 Go back to the database and refresh the screen



You can see the field is updated.



4.7 Go back to **App.java** and change the inputs to **leo@example.com**, which is non-existing data



4.8 Save and run the code

```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 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 }

```

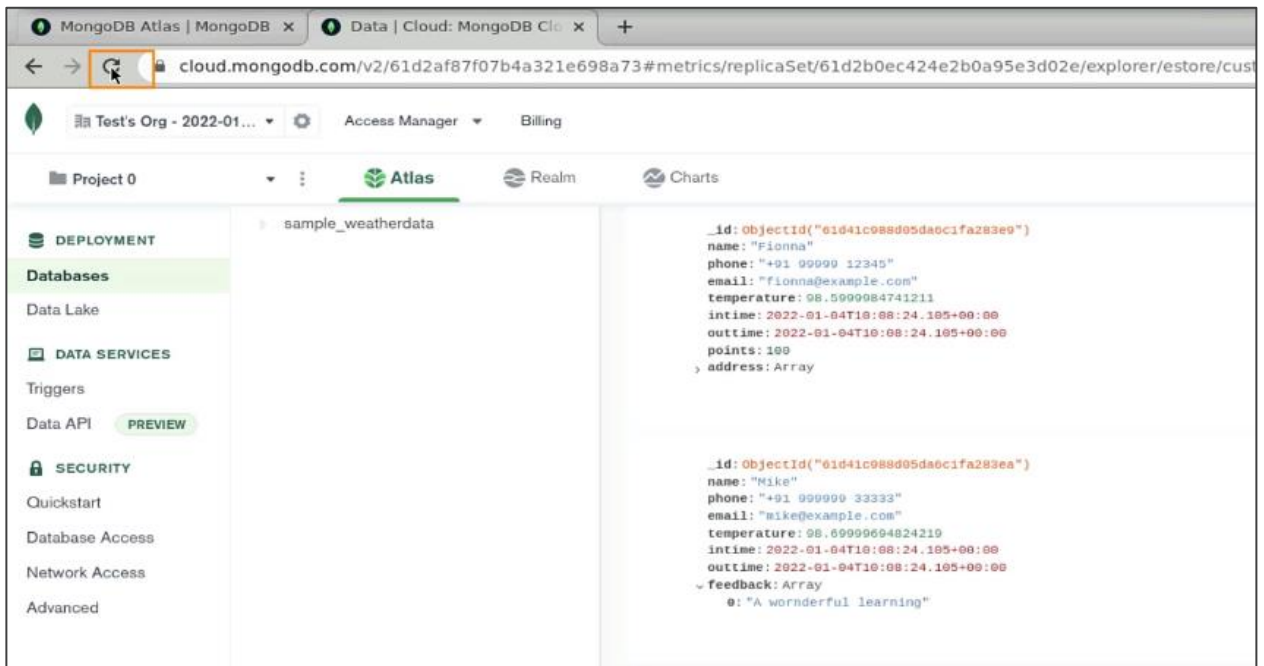
```

<terminated> App (3) [Java Application] Asst/clipse/plugins/org.eclipse.justi.openjdk.hotsp
MongoDB CRUD Operations App
Jan 05, 2022 10:38:53 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
Result is: AcknowledgedUpdateResult(matchedCount=0, modifiedCount=0, upser
Document Updated for email leo@example.com

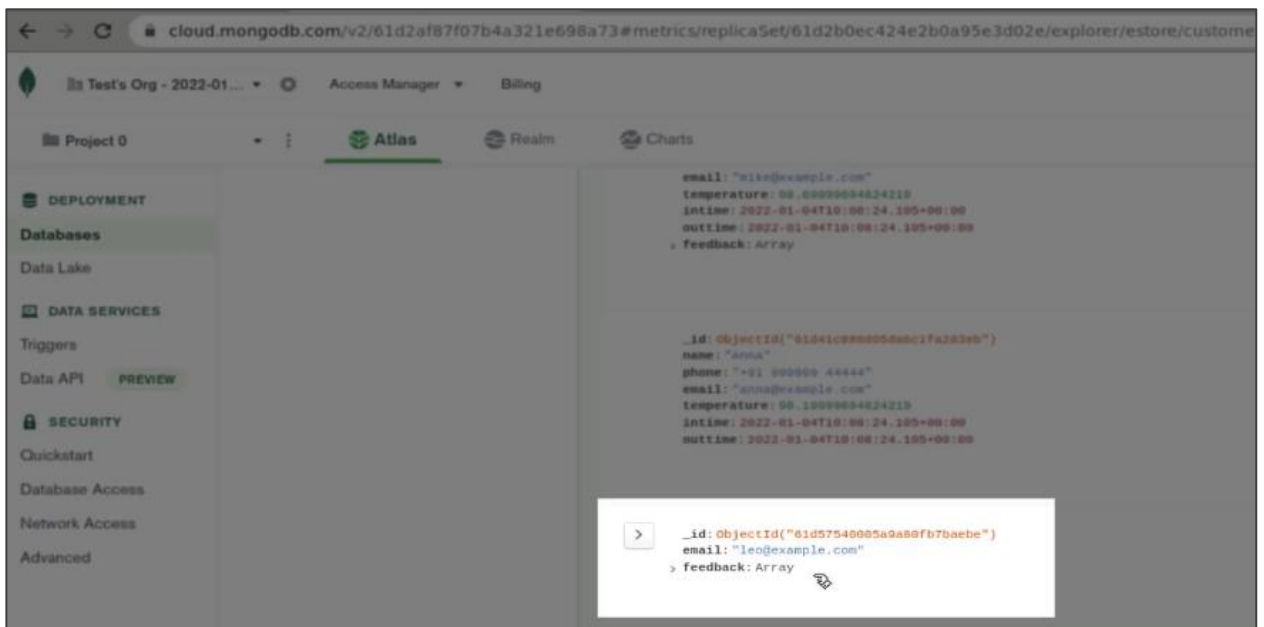
```

You can see that in the acknowledgement result, no match was found, that is, **modifiedCount=0**.

4.9 Go back to the database and refresh the screen



You can see the **upsertOperation** in action.



Step 5: Create more functions

- 5.1 Create a new function **upsertManyCustomers(String email, String key, String value)** to update multiple documents

```
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 upsertManyCustomers() {
    try {
        Bson filter = Filters.gte("points", 50);
        Bson updateOperation = Updates.push("offer", "CASHBACK200");
        UpdateResult result = collection.updateMany(filter, updateOperation);
        System.out.println("Result is: "+result);
    } catch (Exception e) {
        System.out.println("Something went Wrong: "+e);
    }
}
```

- 5.2 Add the field points in the database

The screenshot shows the MongoDB Atlas web interface. On the left, the 'DEPLOYMENT' tab is active, and 'Databases' is selected. The main panel displays a document with the following fields:

- 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

The 'points' field is highlighted in green. Below the document, a 'Updating Document.' message is shown, followed by the updated document structure:

```
1 _id: ObjectId("61d41c988d05da6c1fa283eb")
2 name: "Mike"
3 phone: "+91 99999 33333"
4 email: "mike@example.com"
5 temperature: 98.69999694824219
6 intime: 2022-01-04T10:08:24.105+00:00
7 outtime: 2022-01-04T10:08:24.105+00:00
8 feedback: Array
9 points: 70
```


5.3 Update the try catch block with points as **70** and promocode as **CASHBACK200**

```

DBOperations.java x App.java
110 //Bson updateOperation = Updates.set(key, value);
111 Bson updateOperation = Updates.push(key, value);
112 collection.updateOne(filter, updateOperation);
113 System.out.println("Document Updated for email "+email);
114 } catch (Exception e) {
115     System.out.println("Something went Wrong: "+e);
116 }
117 }
118 }
119 }
120 public void upsertCustomer(String email, String key, String value) {
121     try {
122         Bson filter = Filters.eq("email", email);
123         Bson updateOperation = Updates.push(key, value);
124         UpdateOptions options = new UpdateOptions();
125         options.upsert(true);
126         //collection.updateOne(filter, updateOperation);
127         UpdateResult result = collection.updateOne(filter, updateOperation, options);
128         System.out.println("Result is: "+result);
129         System.out.println("Document Updated for email "+email);
130     } catch (Exception e) {
131         System.out.println("Something went Wrong: "+e);
132     }
133 }
134 }
135 public void upsertManyCustomers() {
136     try {
137         Bson filter = Filters.gte("points", 70);
138         Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
139         UpdateResult result = collection.updateMany(filter, updateOperation);
140         System.out.println("Result is: "+result);
141     } catch (Exception e) {
142         System.out.println("Something went Wrong: "+e);
143     }
144 }
145 }
146 }
147 }

```

5.4 Go back to **App.java** and use the **operation.upsertManyCustomer()** method

```

DBOperations.java x App.java
16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App" );
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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("Customer Details: ");
43     System.out.println(customer);
44     System.out.println(customer);
45
46     //operation
47     //operation
48     //operation
49     //operation
50     //operation
51     operations.upsertManyCustomers();
52 }
53 }

```


5.5 Go back to the **DBOperations.java** file, change the name of the method to **upsertOffersToCustomers()**, and this method will display the promo codes

```

110 //Bson updateOperation = Updates.set(key, value);
111 Bson updateOperation = Updates.push(key, value);
112 collection.updateOne(filter, updateOperation);
113 System.out.println("Document Updated for email "+email);
114 } catch (Exception e) {
115     System.out.println("Something went Wrong: "+e);
116 }
117 }
118 }
119 }
120 public void upsertCustomer(String email, String key, String value) {
121     try {
122         Bson filter = Filters.eq("email", email);
123         Bson updateOperation = Updates.push(key, value);
124         UpdateOptions options = new UpdateOptions();
125         options.upsert(true);
126         //collection.updateOne(filter, updateOperation);
127         UpdateResult result = collection.updateOne(filter, updateOperation, options);
128         System.out.println("Result is: "+result);
129         System.out.println("Document Updated for email "+email);
130     } catch (Exception e) {
131         System.out.println("Something went Wrong: "+e);
132     }
133 }
134
135 public void upsertOffersToCustomers() {
136     try {
137         Bson filter = Filters.gte("points", 70);
138         Bson updateOperation = Updates.push("promoCode", "CASHBACK200");
139         UpdateResult result = collection.updateMany(filter, updateOperation);
140         System.out.println("Result is: "+result);
141     } catch (Exception e) {
142         System.out.println("Something went Wrong: "+e);
143     }
144 }
145 }
146 }

```

5.6 Go back to **App.java** and change the operation name to **upsertOffersToCustomers()**

```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App ");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 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 }
53 }

```

5.7 Save the file and run the code

```

16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 99999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B2B, ABC, Bangalore");
50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     operations.upsertOffersToCustomers[];
52
53 }

```

```

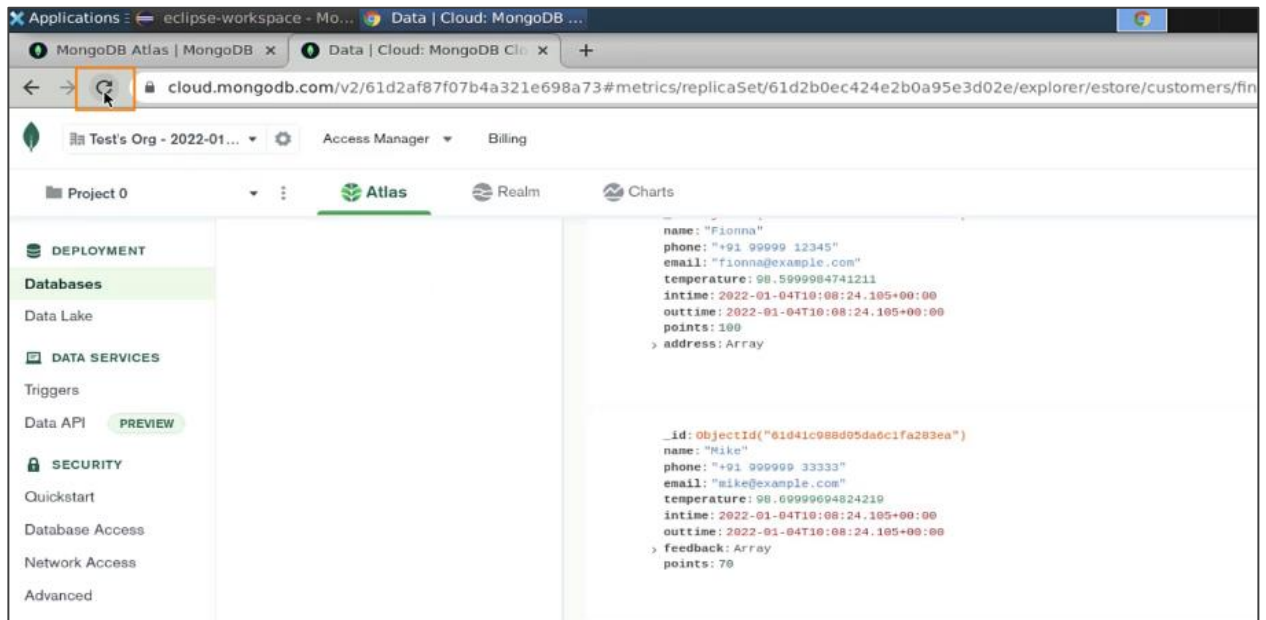
16 public static void main( String[] args )
17 {
18     System.out.println("MongoDB CRUD Operations App");
19     DBOperations operations = new DBOperations();
20
21     /*Customer customer = new Customer("John Watson", "+91 999999 11111", "john@example.com", 98.4f, new Date(), new Date());
22     System.out.println("Customer 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 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 99999 12345");
49     //operations.updateCustomer("fionna@example.com", "address", "2144 B2B, ABC, Bangalore");
50     //operations.upsertCustomer("leo@example.com", "feedback", "A wonderful learning experience");
51     operations.upsertOffersToCustomers[];
52
53 }

```

<terminated> App (3) [Java Application] /usr/eclipseplugins/org.eclipse.justi.openjdk.hotspot...
 MongoDB CRUD Operations App
 Jan 09, 2022 10:45:12 AM com.mongodb.diagnostics.logging.Logger 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 999999 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...
 Result is: AcknowledgedUpdateResult(matchedCount=3, modifiedCount=3, upser...

You can see that three documents were matched, and three documents were modified.

5.8 Go back to the database and refresh the screen

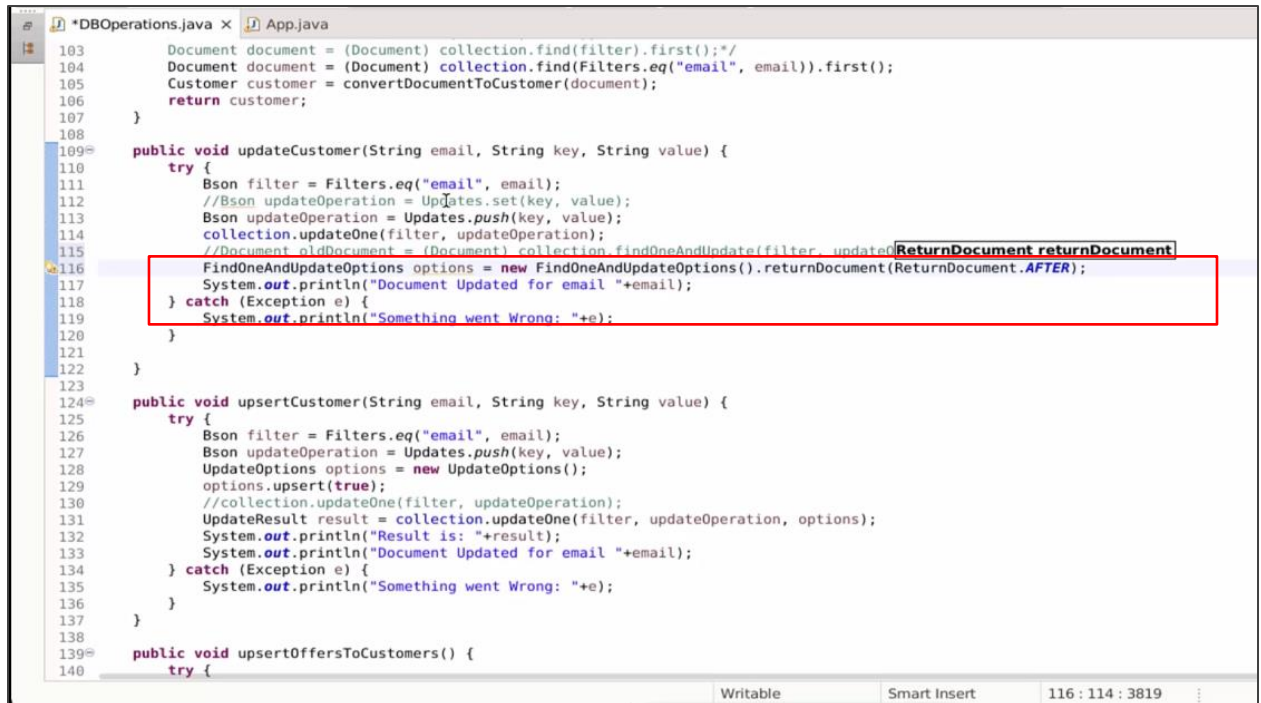


You can see a promo code for points **100** as the condition was greater than or equal to **70**.



Step 6: Write a new method

- 6.1 Go back to the **DBOperations.java**, create a new collection, and use the **findOneAndUpdate** method, which will take the update operation as input to update the document



```
103     Document document = (Document) collection.find(filter).first();  
104     Document document = (Document) collection.find(Filters.eq("email", email)).first();  
105     Customer customer = convertDocumentToCustomer(document);  
106     return customer;  
107 }  
108  
109 public void updateCustomer(String email, String key, String value) {  
110     try {  
111         Bson filter = Filters.eq("email", email);  
112         //Bson updateOperation = Updates.set(key, value);  
113         Bson updateOperation = Updates.push(key, value);  
114         collection.updateOne(filter, updateOperation);  
115         //Document oldDocument = (Document) collection.findOneAndUpdate(filter, updateOperation, ReturnDocument.BEFORE);  
116         FindOneAndUpdateOptions options = new FindOneAndUpdateOptions().returnDocument(ReturnDocument.AFTER);  
117         System.out.println("Document Updated for email "+email);  
118     } catch (Exception e) {  
119         System.out.println("Something went Wrong: "+e);  
120     }  
121 }  
122  
123  
124 public void upsertCustomer(String email, String key, String value) {  
125     try {  
126         Bson filter = Filters.eq("email", email);  
127         Bson updateOperation = Updates.push(key, value);  
128         UpdateOptions options = new UpdateOptions();  
129         options.upsert(true);  
130         //collection.updateOne(filter, updateOperation);  
131         UpdateResult result = collection.updateOne(filter, updateOperation, options);  
132         System.out.println("Result is: "+result);  
133         System.out.println("Document Updated for email "+email);  
134     } catch (Exception e) {  
135         System.out.println("Something went Wrong: "+e);  
136     }  
137 }  
138  
139 public void upsertOffersToCustomers() {  
140     try {
```


The final code appears as shown below:

```

# DBOperations.java x App.java
103 Document document = (Document) collection.find(filter).first();/*
104 Document document = (Document) collection.find(Filters.eq("email", email)).first();
105 Customer customer = convertDocumentToCustomer(document);
106 return customer;
107 }
108
109 public void updateCustomer(String email, String key, String value) {
110     try {
111         Bson filter = Filters.eq("email", email);
112         //Bson updateOperation = Updates.set(key, value);
113         Bson updateOperation = Updates.push(key, value);
114         collection.updateOne(filter, updateOperation);
115         //Document oldDocument = (Document) collection.findOneAndUpdate(filter, updateOperation);
116         FindOneAndUpdateOptions options = new FindOneAndUpdateOptions().returnDocument(ReturnDocument.AFTER);
117         Document updatedDocument = (Document) collection.findOneAndUpdate(filter, updateOperation, options);
118         System.out.println("Document Updated for email "+email);
119         System.out.println("Updated Version of Document is: "+updatedDocument.toJson());
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

```

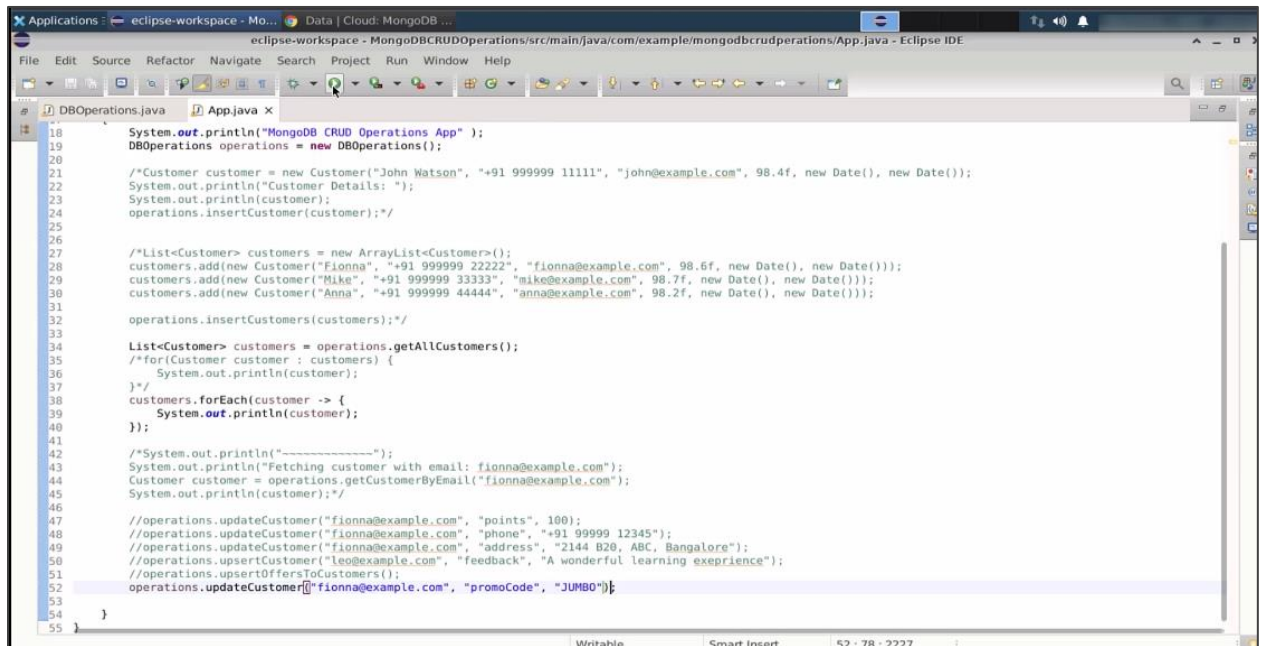
6.2 Go back to the **App.java** file and edit the promoCode as **JUMBO** in the **updateCustomer** method

```

# DBOperations.java x App.java x
18 System.out.println("MongoDB CRUD Operations App" );
19 DBOperations operations = new DBOperations();
20
21 /*Customer customer = new Customer("John Watson", "+91 999999 11111", "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 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 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 }
55

```

6.3 Save the file and run the code

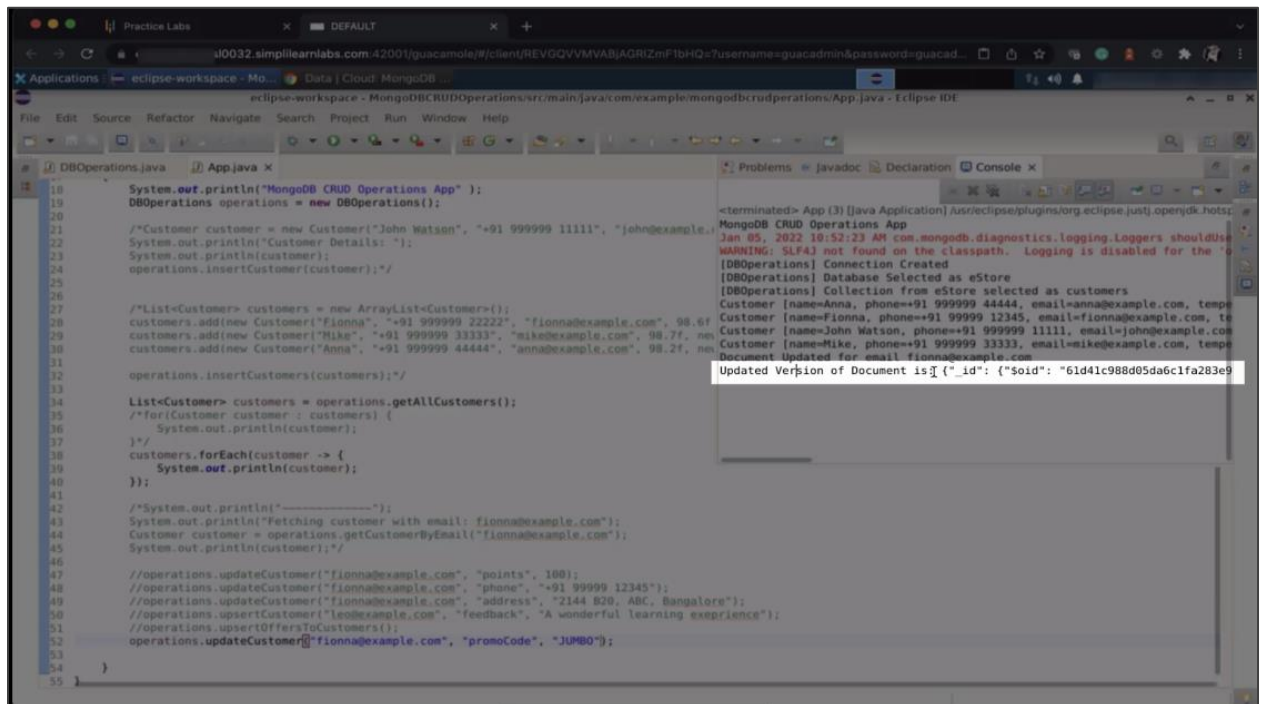


```

18      System.out.println("MongoDB CRUD Operations App");
19      DBOperations operations = new DBOperations();
20
21      /*Customer customer = new Customer("John Watson", "+91 99999 11111", "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 99999 22222", "fionna@example.com", 98.6f, new Date(), new Date()));
29      customers.add(new Customer("Mike", "+91 99999 33333", "mike@example.com", 98.7f, new Date(), new Date()));
30      customers.add(new Customer("Anna", "+91 99999 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 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", "JUMB0");
53
54  }
55  }

```


You can see the output as the **Updated Version of Document** is along with other details.



```

18 System.out.println("MongoDB CRUD Operations App");
19 DBOperations operations = new DBOperations();
20
21 /*Customer customer = new Customer("John Watson", "+91 99999 11111", "john@example.com");
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 99999 22222", "fionna@example.com", 98.6f);
29 customers.add(new Customer("Mike", "+91 99999 33333", "mike@example.com", 98.7f, null);
30 customers.add(new Customer("Anna", "+91 99999 44444", "anna@example.com", 98.2f, null);
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 829, ABC, Bangalore");
50 //operations.upsertCustomer("leona@example.com", "feedback", "A wonderful learning experience");
51 //operations.upsertOffersToCustomers();
52 operations.updateCustomer("fionna@example.com", "promoCode", "JUMB0");
53
54 }
55

```

```

<terminated> App (3) [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
MongoDB CRUD Operations App
Jan 05, 2022 10:52:23 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 99999 44444, email=anna@example.com, temp...
Customer [name=Fionna, phone=+91 99999 12345, email=fionna@example.com, te...
Customer [name=John Watson, phone=+91 99999 11111, email=john@example.com...
Customer [name=Mike, phone=+91 99999 33333, email=mike@example.com, temp...
Document Updated for email fionna@example.com
Updated Version of Document is { "_id": {"$oid": "61d41c988d05da6c1fa283e9"

```

This is all about the update operations. You can perform an update on a single document using the **UpdateOne** function and an **upsert** operation which is a combination of update and insert. The **updateMany** operation can be used with some filters or conditions.

By following these steps, you have successfully worked with the CRUD operations and updated the documents for existing and non-existing customers.