CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

LAB-6: NoSQL – MongoDB, GCP & Neo4j

GitLab Link: https://git.cs.dal.ca/jspatel/csci5408_s24_b00982253_jay_patel.git

Table of Content

Task 1: MongoDB

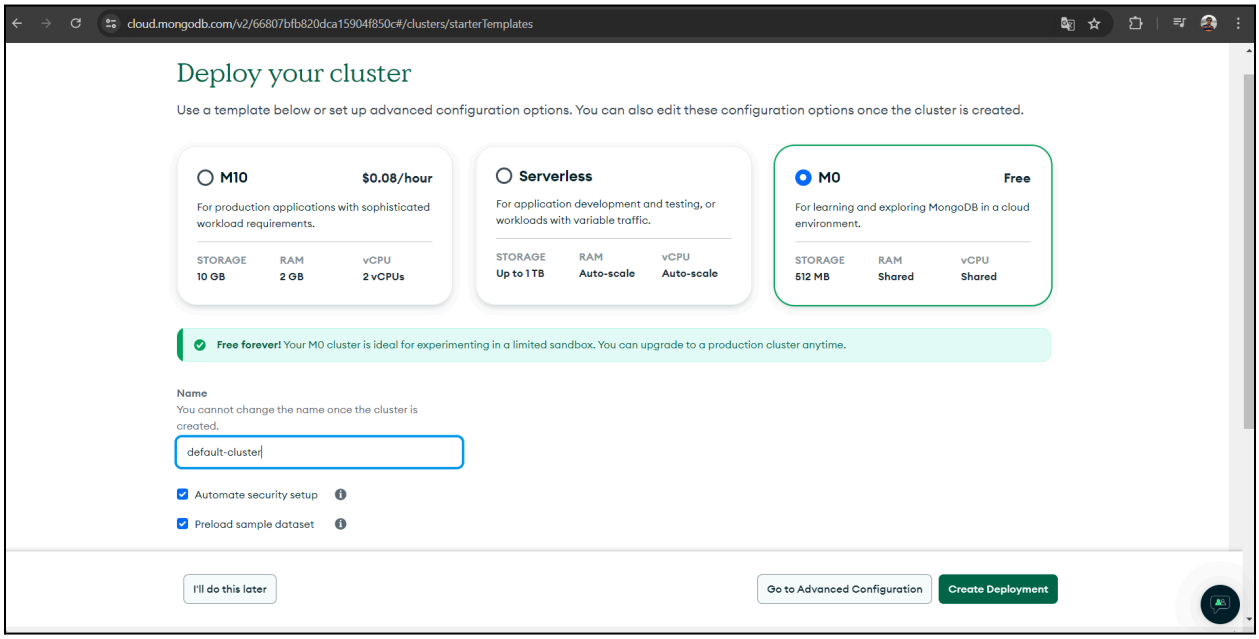

Figure 1.1: Create a Cluster of name "default-cluster"

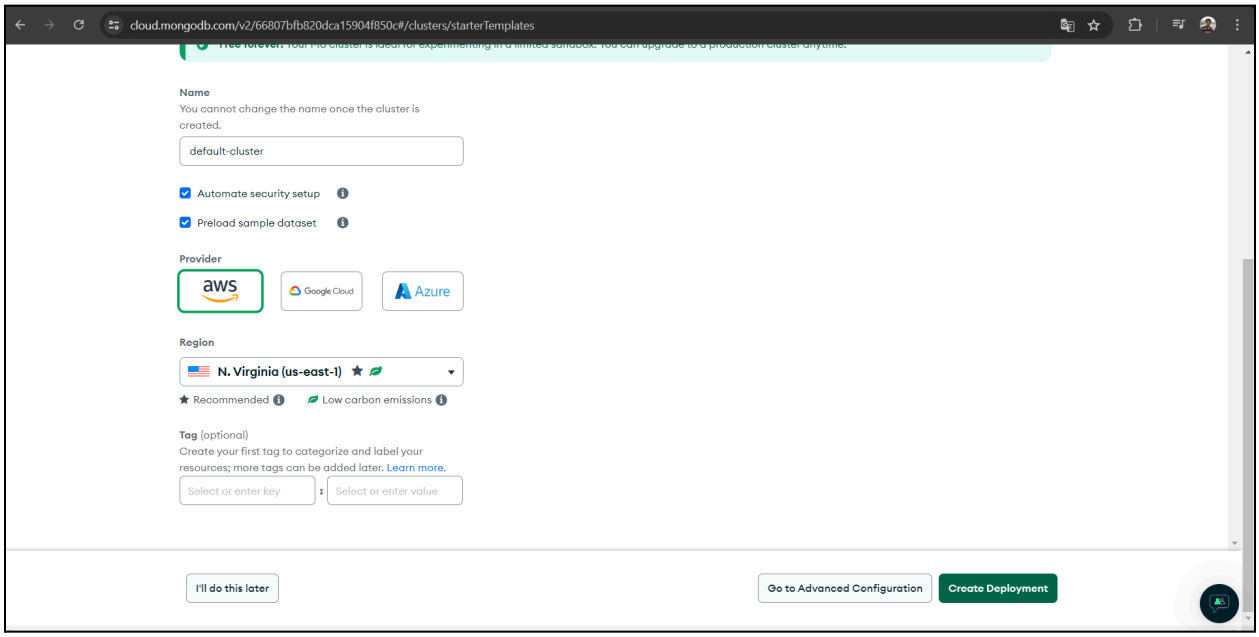

Figure 1.2: Choose Cloud Provider and Region

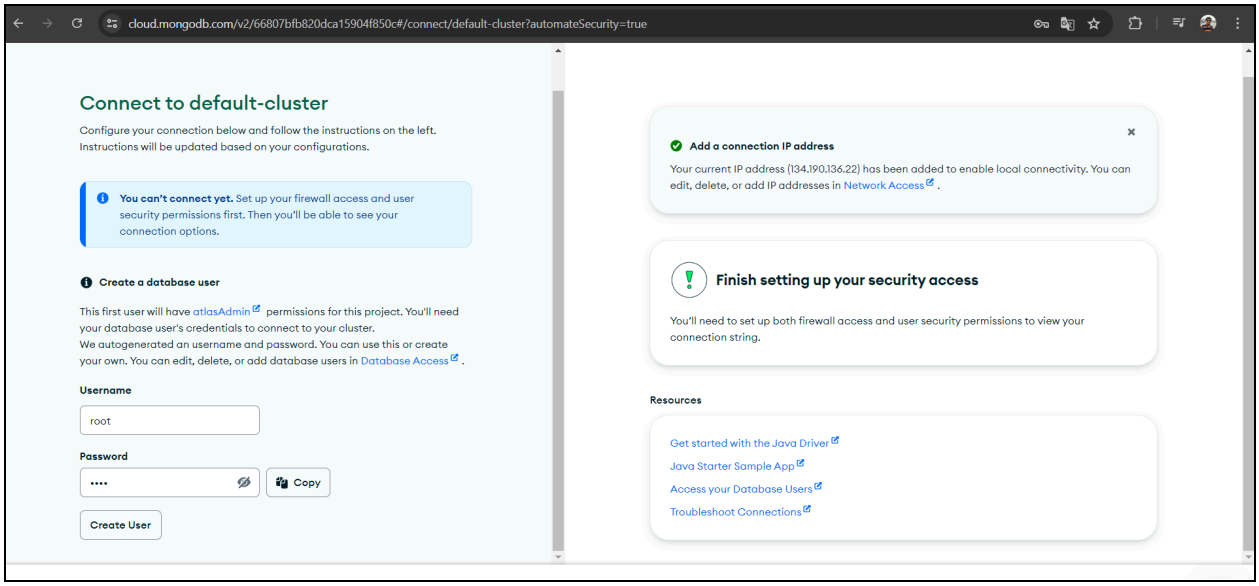

Figure 1.3: Create a new database user

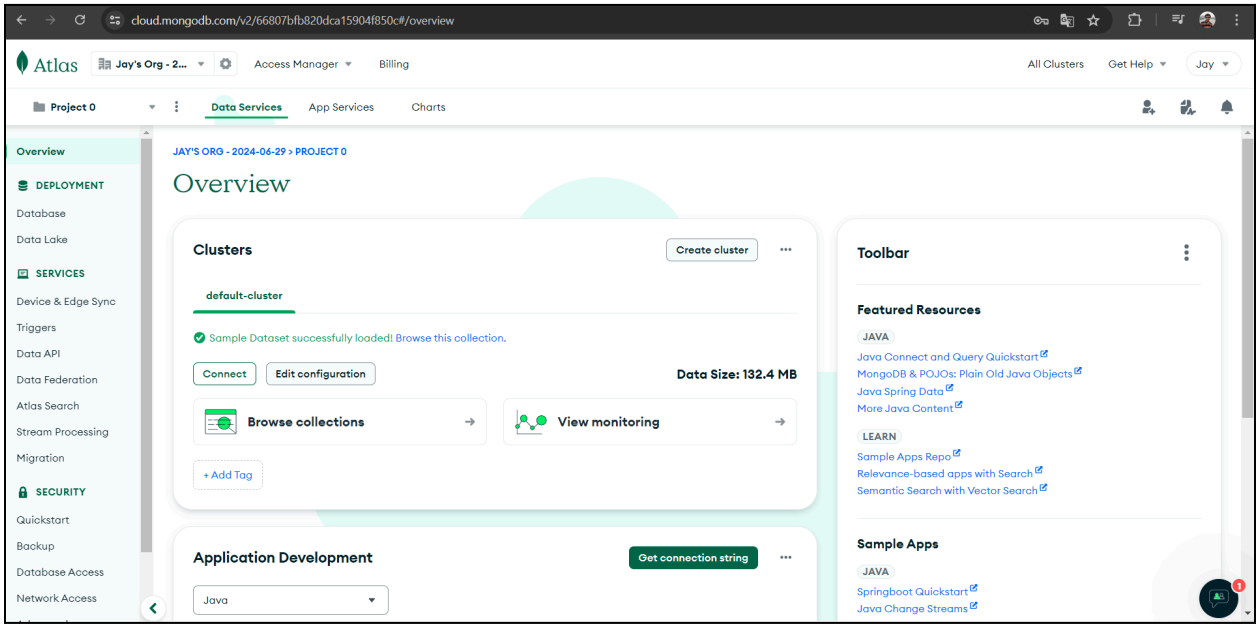

Figure 1.4: New MongoDB Account and Cluster Creation is Successfully Done

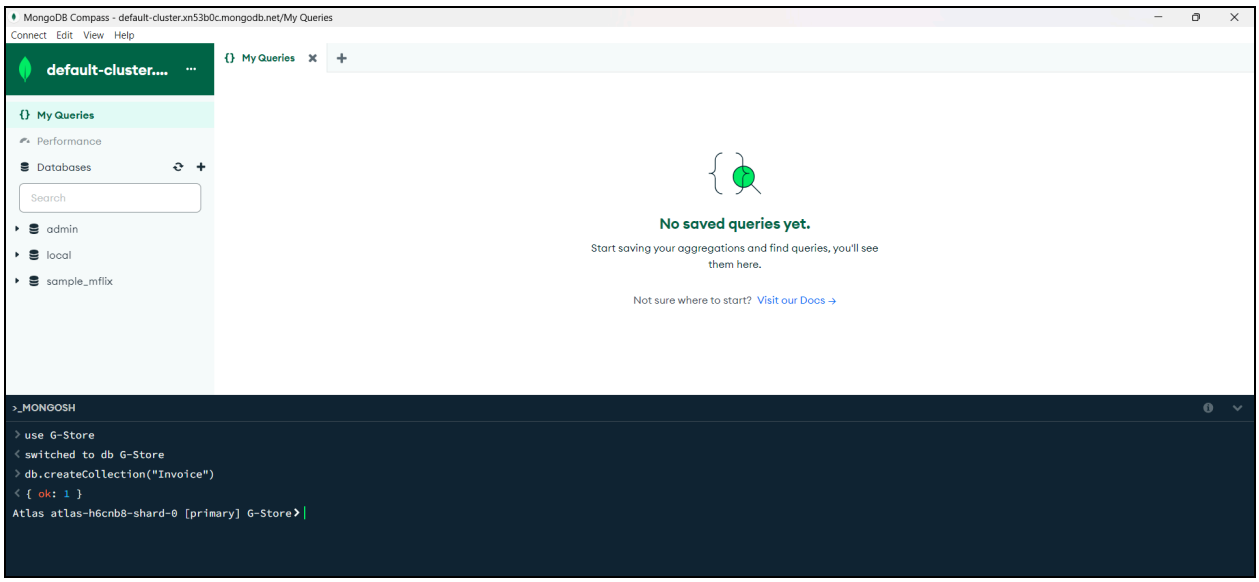

Figure 1.5: Create database and collection in MongoBD Compass

INSERT

```
no usages
32 @    private static void insert(Invoice invoice){
33          Document doc = new Document("item", invoice.getItem())
34                  .append("quantity", invoice.getQuantity())
35                          .append("price", invoice.getPrice());
36
37          collection.insertOne(doc);
38
39          ObjectId id = doc.getObjectId( key: "_id");
40          System.out.println("New Item Inserted ID: " + id.toHexString());
41      }
42
```

Figure 1.6: Insert Query Code

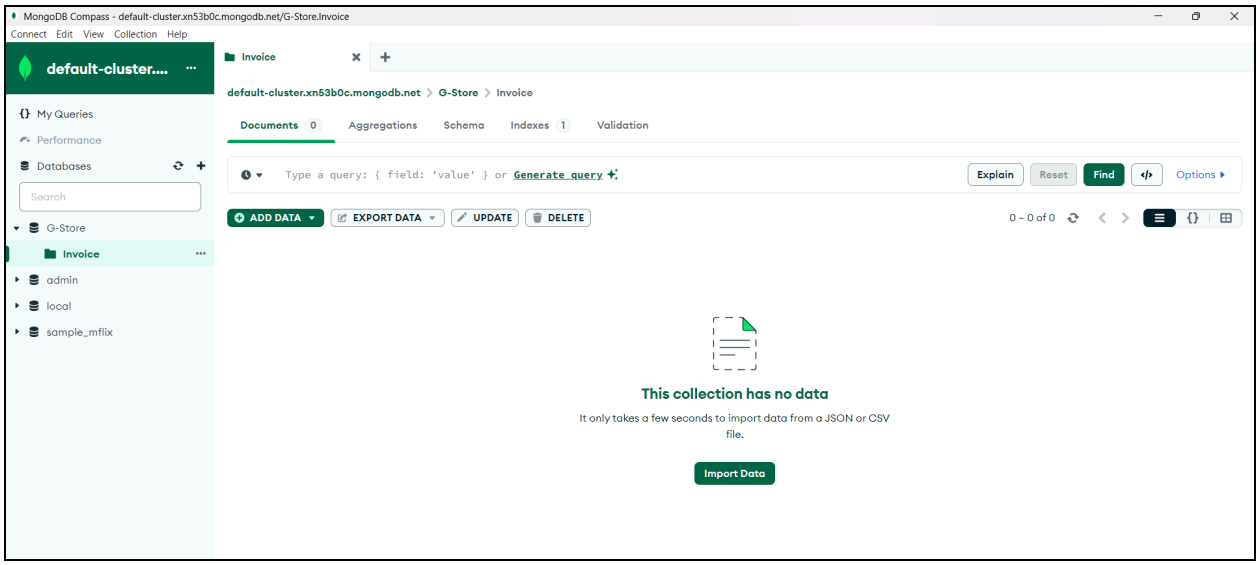

Figure 1.7: Database before insert query was performed

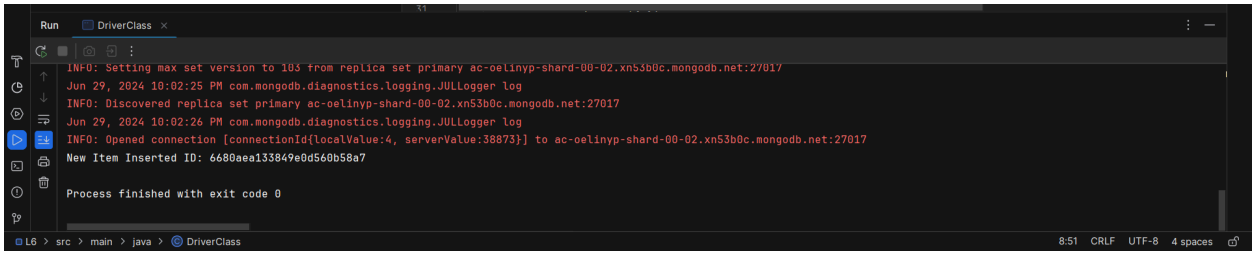

Figure 1.8: Run Insert Query

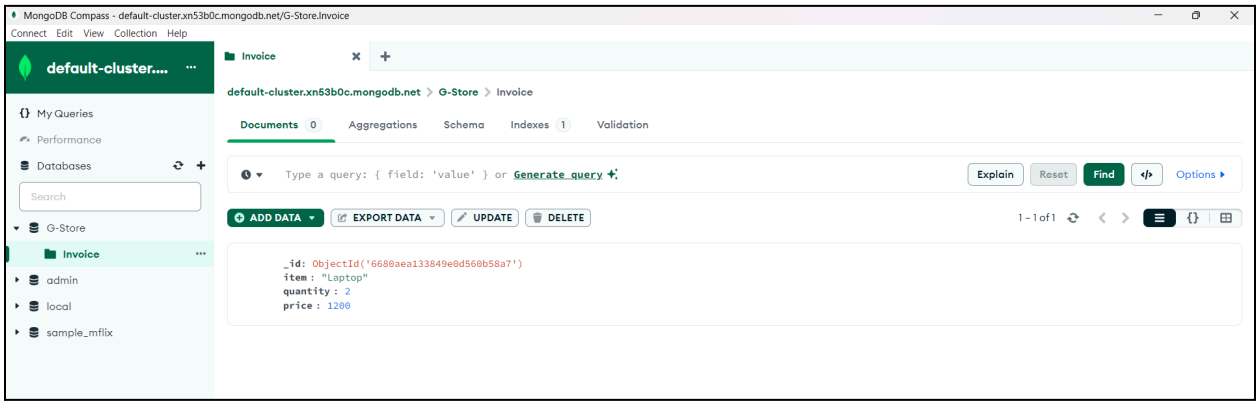

Figure 1.9: Database after insert query was performed

GET

```
1 usage
private static void get(String invoiceName){
    Document doc = collection.find(eq( fieldName: "item", invoiceName)).first();
    if (doc != null) {
        Invoice invoice = new Invoice(doc.getString( key: "item"), doc.getInteger( key: "quantity"), doc.getInteger( key: "price"));
        System.out.println("Item: " + invoice.getItem() + ", Quantity: " + invoice.getQuantity() + ", Price: " + invoice.getPrice());
    }
    else {
        System.out.println("Item not found!");
    }
}
```

Figure 1.10: Get Query Code

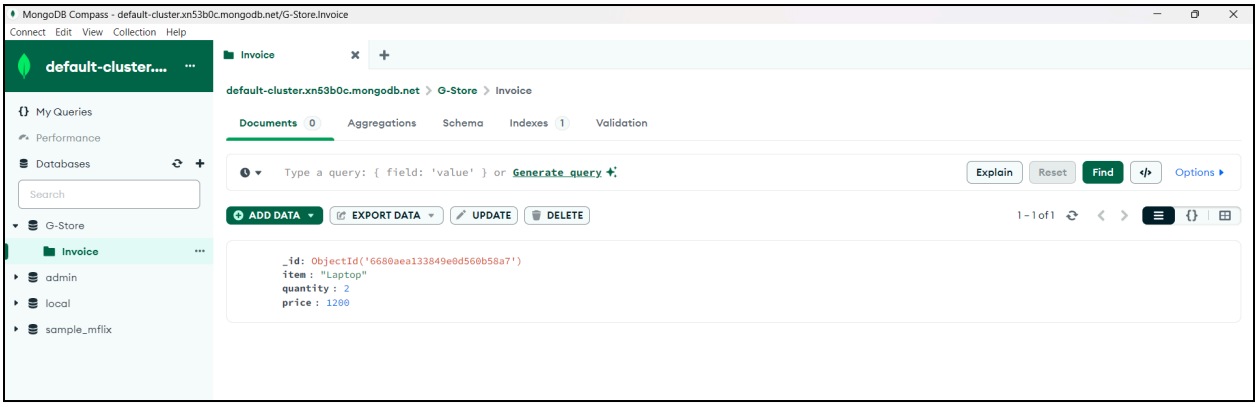

Figure 1.11: Database before get query was performed

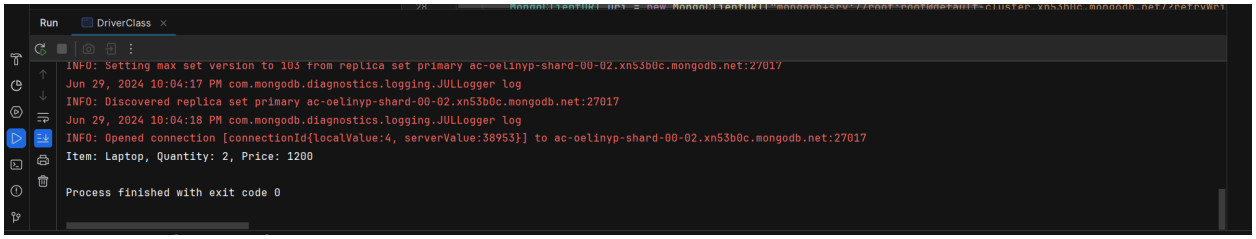

Figure 1.12: Run Get Query

UPDATE

```
no usages
private static void update(String invoiceName, int updatePrice){
    Document foundItem = collection.find(eq( fieldName: "item", invoiceName)).first();
    if (foundItem != null) {
        collection.updateOne(eq( fieldName: "item", invoiceName), new Document("$set", new Document("price", updatePrice)));
        get(invoiceName);
    } else {
        System.out.println("Item not found, so it can't be updated!");
    }
}
```

Figure 1.13: Update Query Code

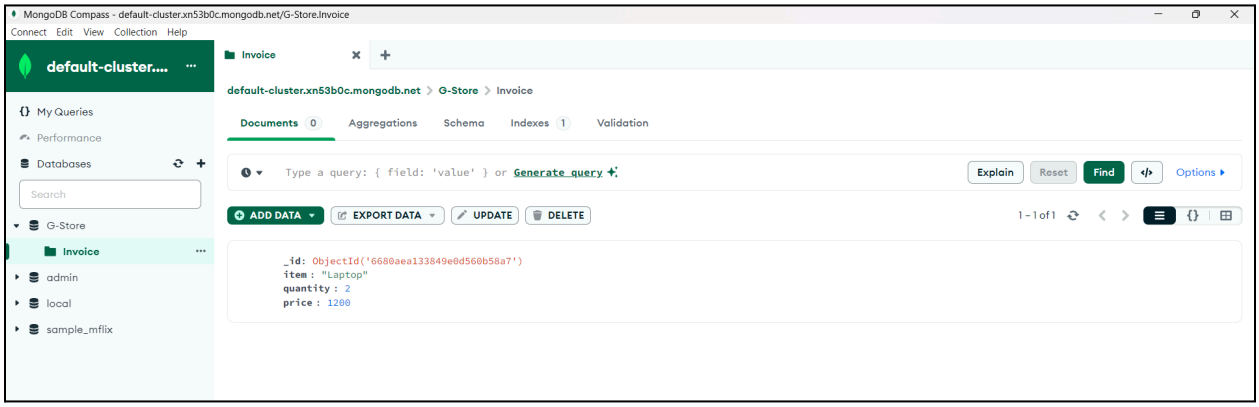

Figure 1.14: Database before update query was performed

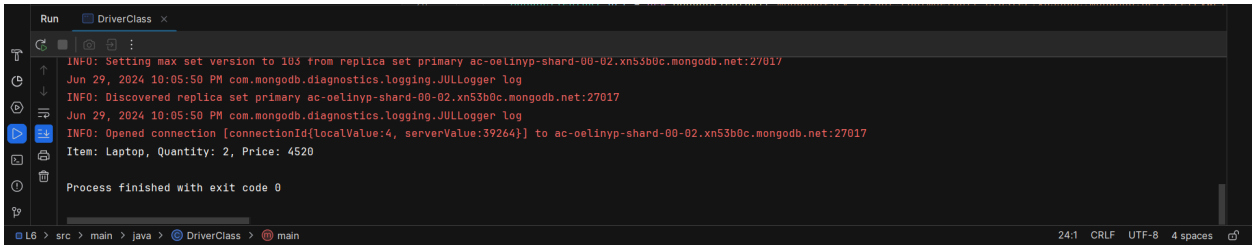

Figure 1.15: Run Update Query

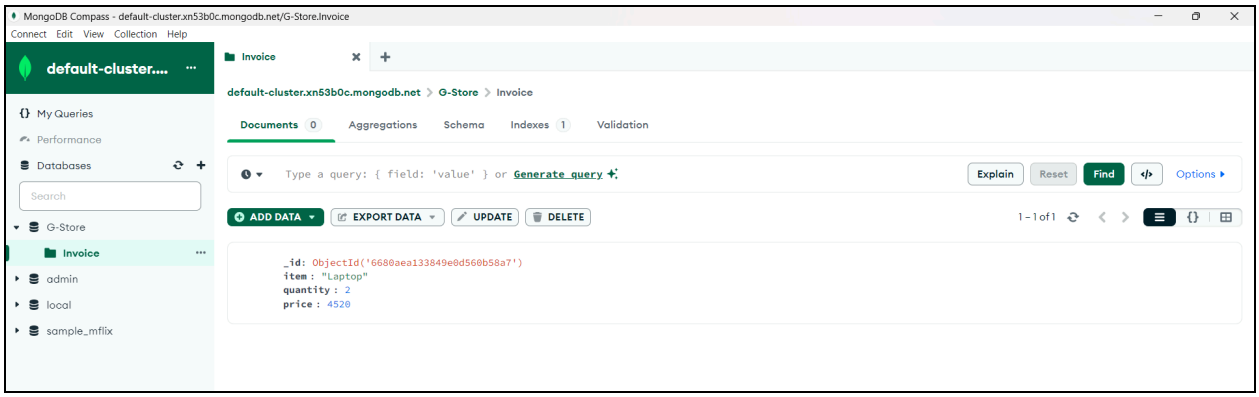

Figure 1.16: Database after update query was performed

DELETE

```
1 usage
    private static void delete(String invoiceName){
        Document foundItem = collection.find(eq( fieldName: "item", invoiceName)).first();
        if (foundItem != null) {
            ObjectId id = foundItem.getObjectId( key: "_id");
            collection.deleteOne(eq( fieldName: "_id", id));
            System.out.println("Item deleted successfully!");
        } else {
            System.out.println("Item not found, so it can't be deleted!");
        }
    }
}
```

Figure 1.17: Delete Query Code

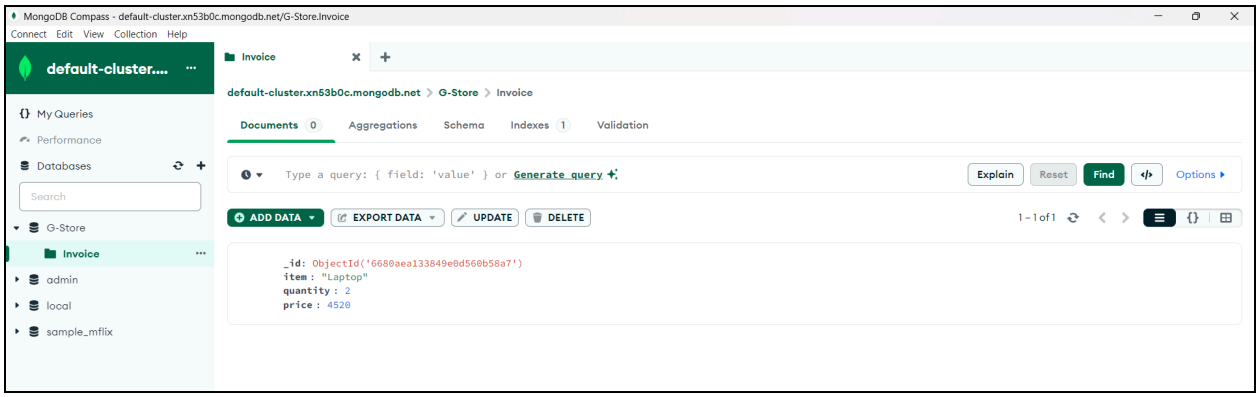

Figure 1.18: Database before delete query was performed

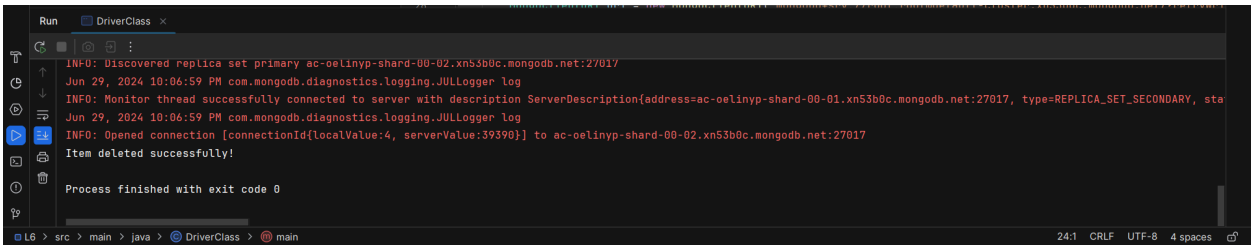

Figure 1.19: Run Delete Query

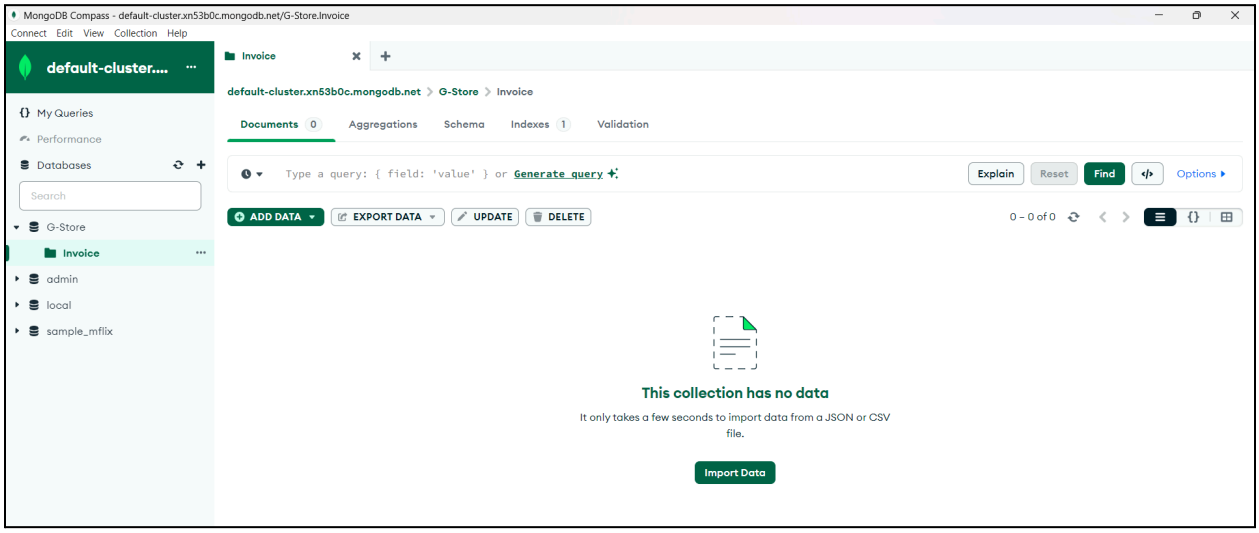

Figure 1.20: Database after delete query was performed

Task 2: Neo4j

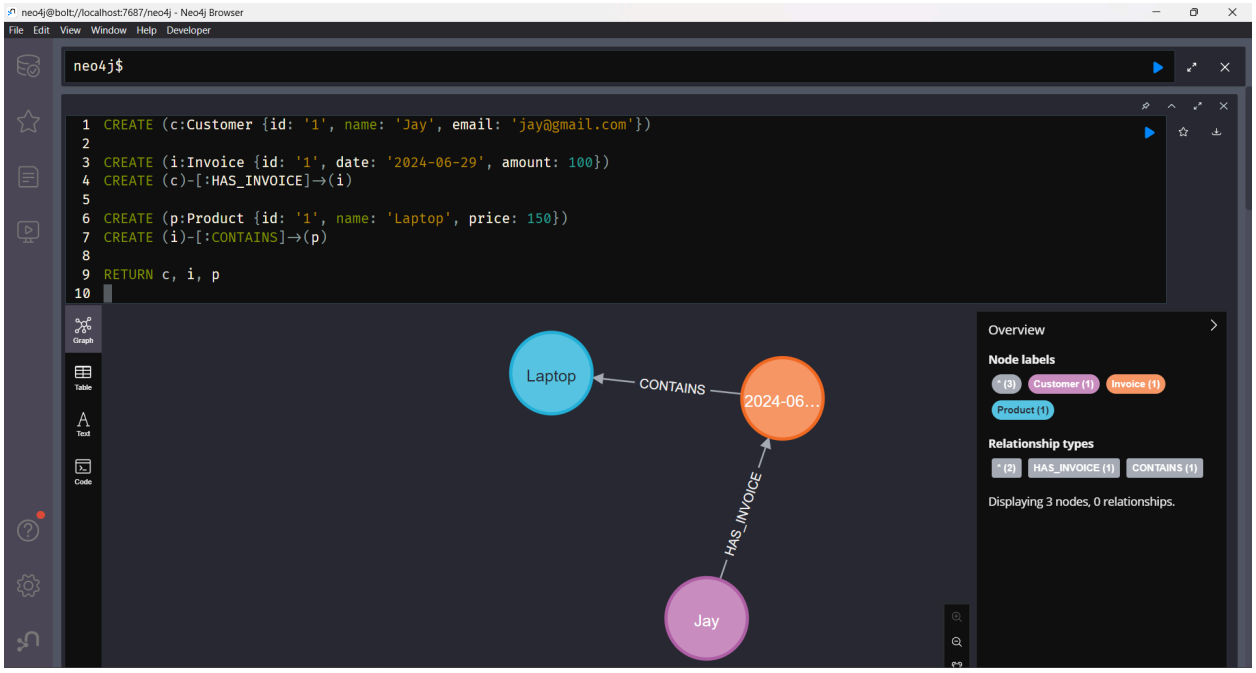

Figure 2.1: Create a graph

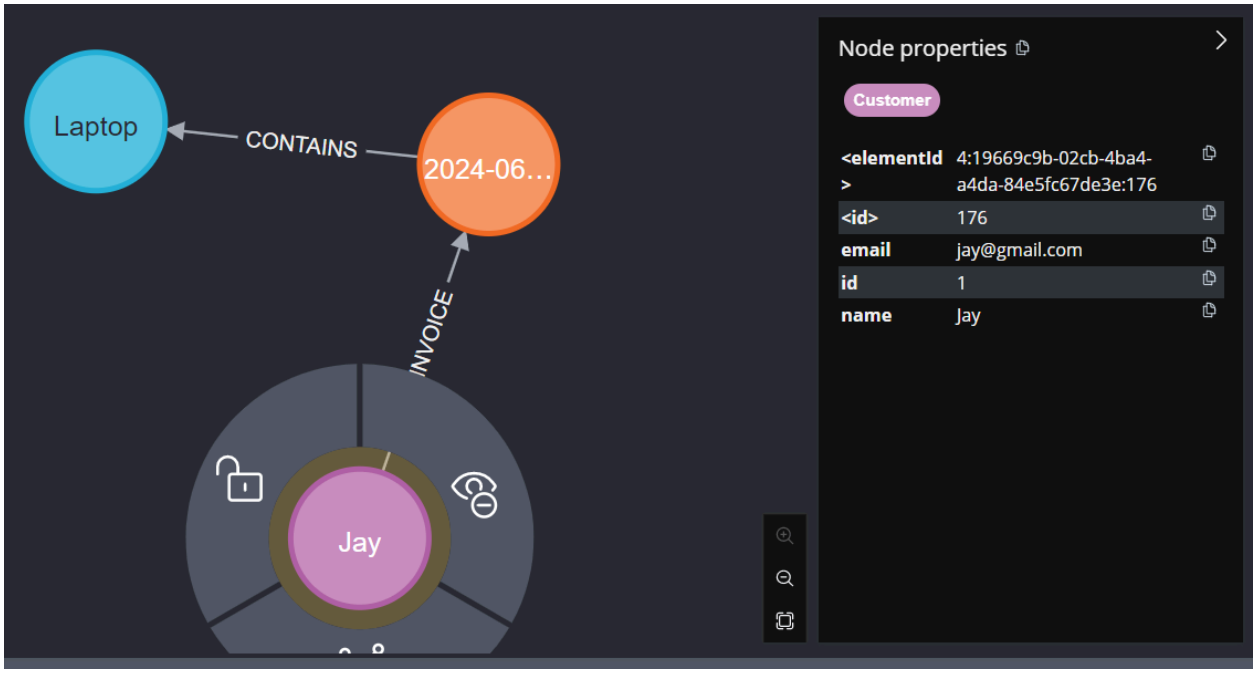

Figure 2.2: Show properties of the "Customer" node

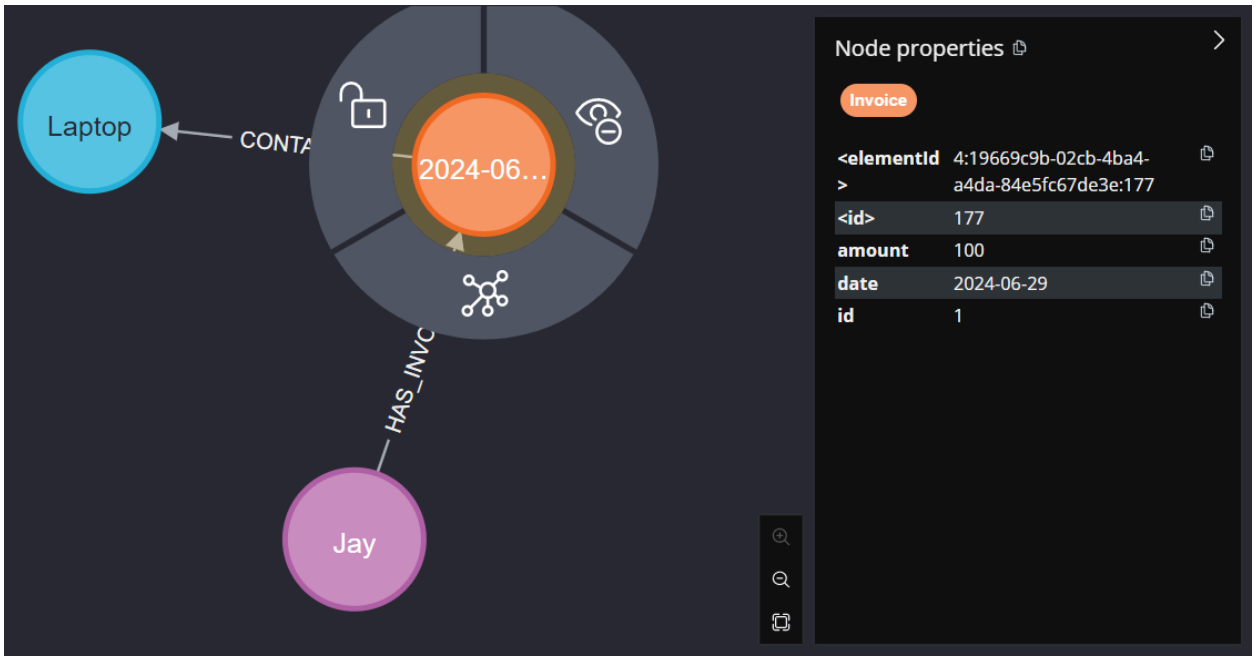

Figure 2.3: Show properties of the "Invoice" node

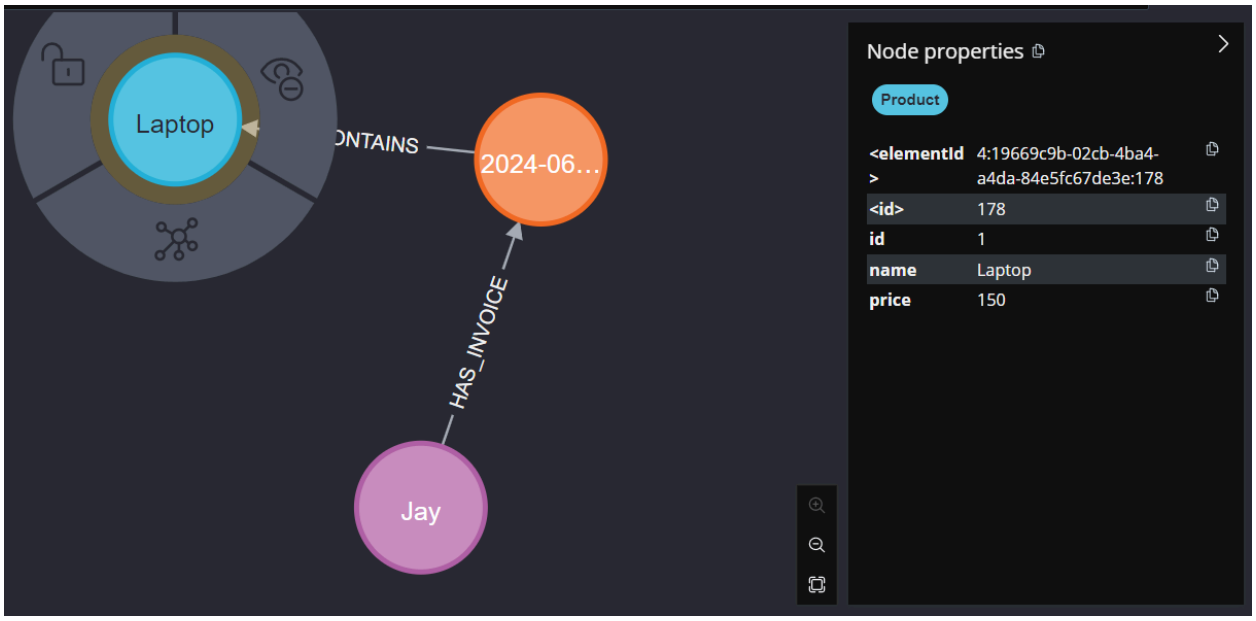

Figure 2.4: Show properties of the "Product" node