

# 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](https://git.cs.dal.ca/jspatel/csci5408_s24_b00982253_jay_patel.git)

# Table of Content

|                      |    |
|----------------------|----|
| Task 1: MongoDB..... | 3  |
| Task 2: Neo4j.....   | 11 |

# Task 1: MongoDB

The screenshot shows the 'Deploy your cluster' page on the MongoDB Cloud console. The page has a dark header with the URL 'cloud.mongodb.com/v2/66807bfb820dca15904f850c#/clusters/starterTemplates'. The main content area is white and features three deployment options: 'M10' (\$0.08/hour), 'Serverless', and 'M0' (Free). The 'M0' option is selected with a blue radio button. Below the options, a green banner states: 'Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.' The 'Name' section has a text input field containing 'default-cluster'. Below this, there are two checked checkboxes: 'Automate security setup' and 'Preload sample dataset'. At the bottom, there are three buttons: 'I'll do this later', 'Go to Advanced Configuration', and 'Create Deployment'.

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

☐ M10 \$0.08/hour  
For production applications with sophisticated workload requirements.

☐ Serverless  
For application development and testing, or workloads with variable traffic.

☒ M0 Free  
For learning and exploring MongoDB in a cloud environment.

STORAGE RAM vCPU

10 GB 2 GB 2 vCPUs

Up to 1 TB Auto-scale Auto-scale

512 MB Shared Shared

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name  
You cannot change the name once the cluster is created.

default-cluster

☒ Automate security setup ⓘ

☒ Preload sample dataset ⓘ

I'll do this later Go to Advanced Configuration Create Deployment

Figure 1.1: Create a Cluster of name “default-cluster”

The screenshot shows the 'Choose Cloud Provider and Region' section of the MongoDB Cloud deployment page. The 'aws' provider is selected with a green border. Below the providers, the 'Region' dropdown is set to 'N. Virginia (us-east-1)'. There are two checkboxes: 'Recommended' (checked) and 'Low carbon emissions' (unchecked). Below this, there is a 'Tag (optional)' section with a text input field for the key and a text input field for the value. At the bottom, there are three buttons: 'I'll do this later', 'Go to Advanced Configuration', and 'Create Deployment'.

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Name  
You cannot change the name once the cluster is created.

default-cluster

☒ Automate security setup ⓘ

☒ Preload sample dataset ⓘ

Provider

aws Google Cloud Azure

Region

N. Virginia (us-east-1) ★

★ Recommended ⓘ Low carbon emissions ⓘ

Tag (optional)  
Create your first tag to categorize and label your resources; more tags can be added later. [Learn more.](#)

Select or enter key Select or enter value

I'll do this later Go to Advanced Configuration Create Deployment

Figure 1.2: Choose Cloud Provider and Region

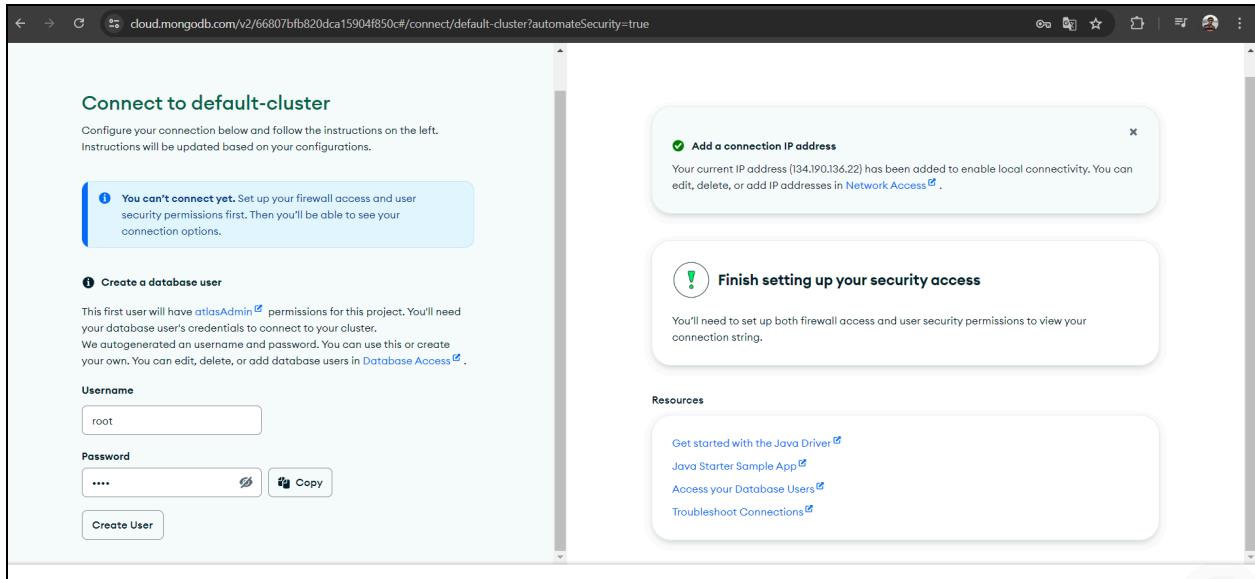


Figure 1.3: Create a new database user

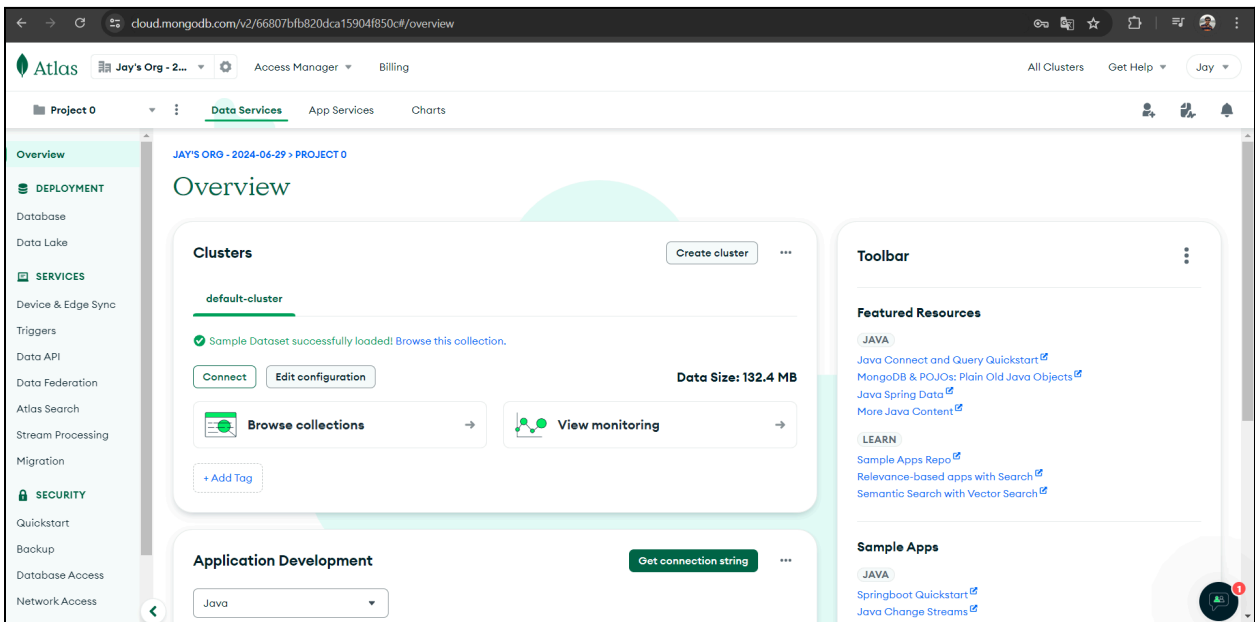
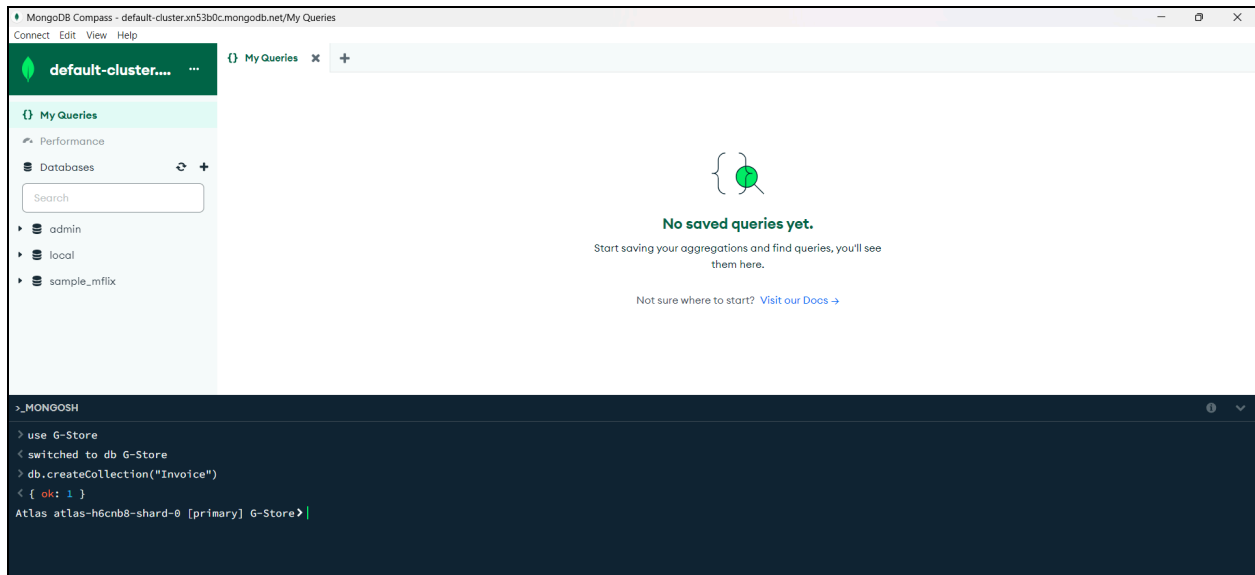


Figure 1.4: New MongoDB Account and Cluster Creation is Successfully Done



*Figure 1.5: Create database and collection in MongoDB 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*

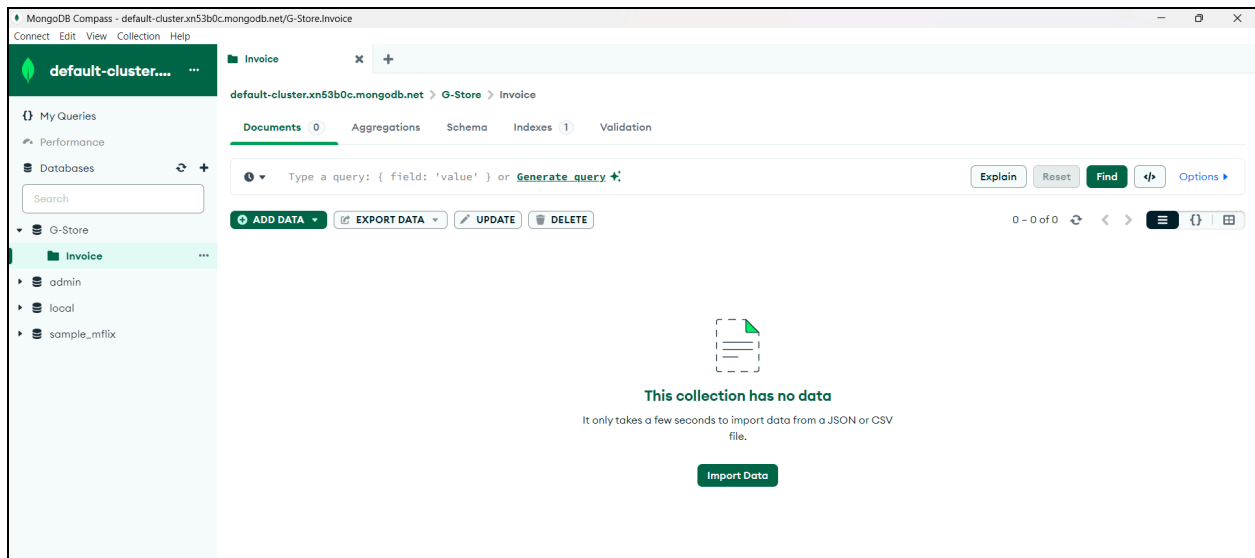


Figure 1.7: Database before insert query was performed

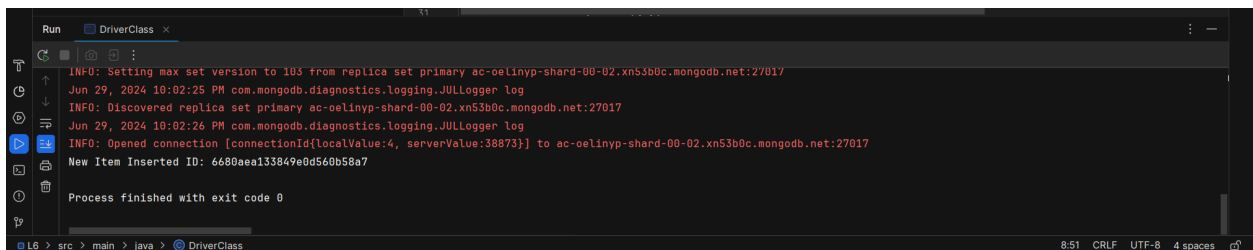


Figure 1.8: Run Insert Query

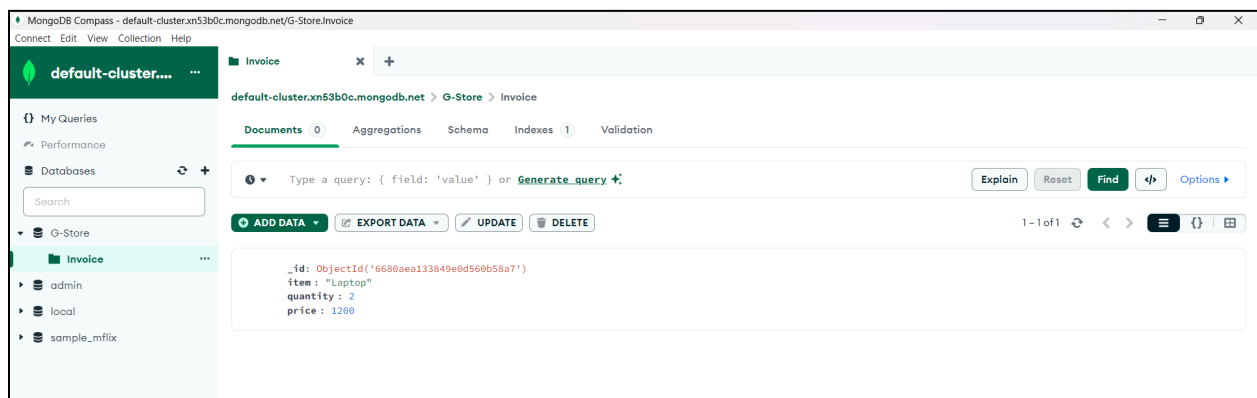


Figure 1.9: Database after insert query was performed

## GET

```
53     private static void get(String invoiceName){
54         Document doc = collection.find(eq("fieldName": "item", invoiceName)).first();
55         if (doc != null) {
56             Invoice invoice = new Invoice(doc.getString("key: \"item\""), doc.getInteger("key: \"quantity\""), doc.getInteger("key: \"price\""));
57             System.out.println("Item: " + invoice.getItem() + ", Quantity: " + invoice.getQuantity() + ", Price: " + invoice.getPrice());
58         }
59         else {
60             System.out.println("Item not found!");
61         }
62     }
63 }
```

Figure 1.10: Get Query Code

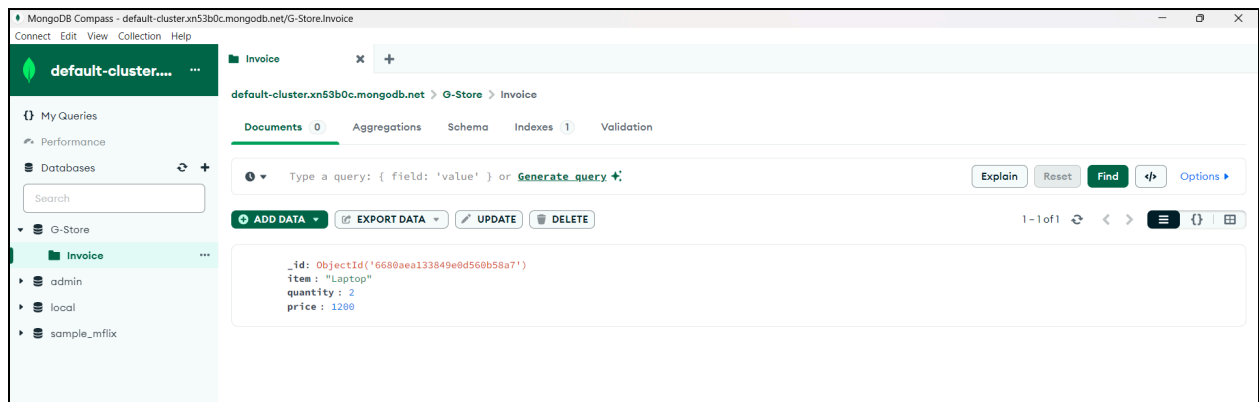


Figure 1.11: Database before get query was performed

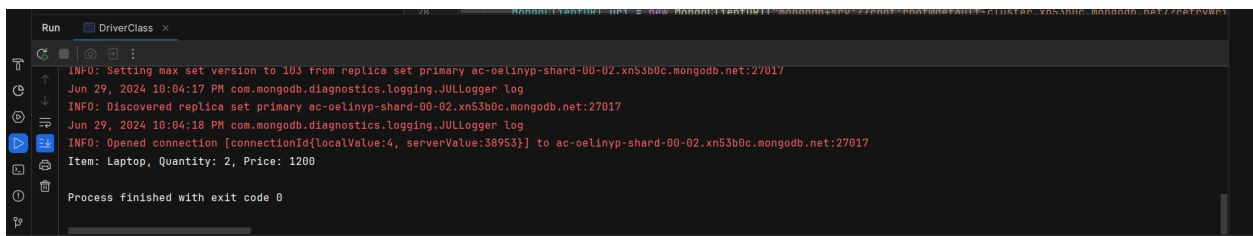


Figure 1.12: Run Get Query

## UPDATE

```
no usages
43 private static void update(String invoiceName, int updatePrice){
44     Document foundItem = collection.find(eq( fieldName: "item", invoiceName)).first();
45     if (foundItem != null) {
46         collection.updateOne(eq( fieldName: "item", invoiceName), new Document("$set", new Document("price", updatePrice)));
47         get(invoiceName);
48     } else {
49         System.out.println("Item not found, so it can't be updated!");
50     }
51 }
52 }
```

Figure 1.13: Update Query Code

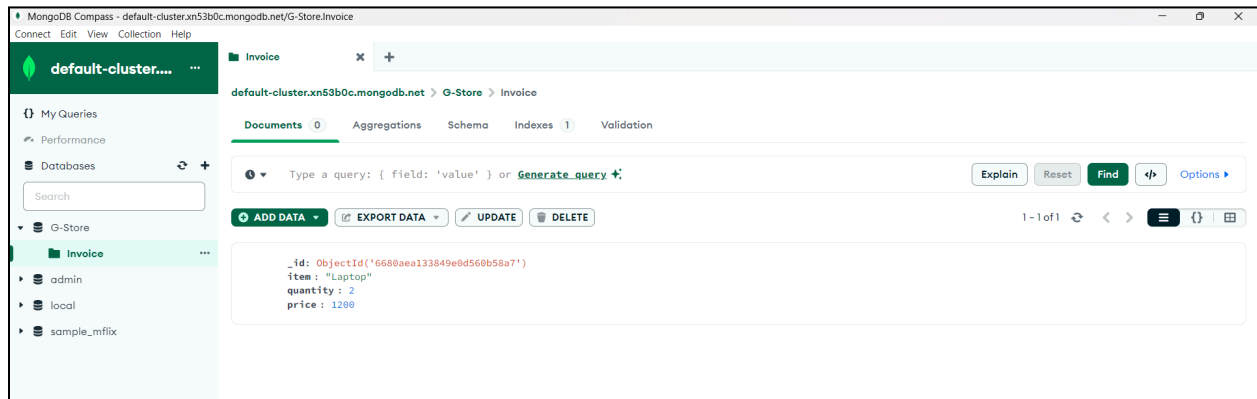


Figure 1.14: Database before update query was performed

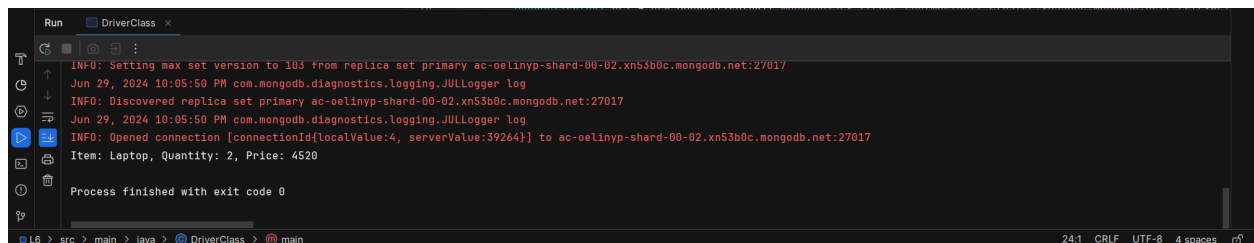


Figure 1.15: Run Update Query



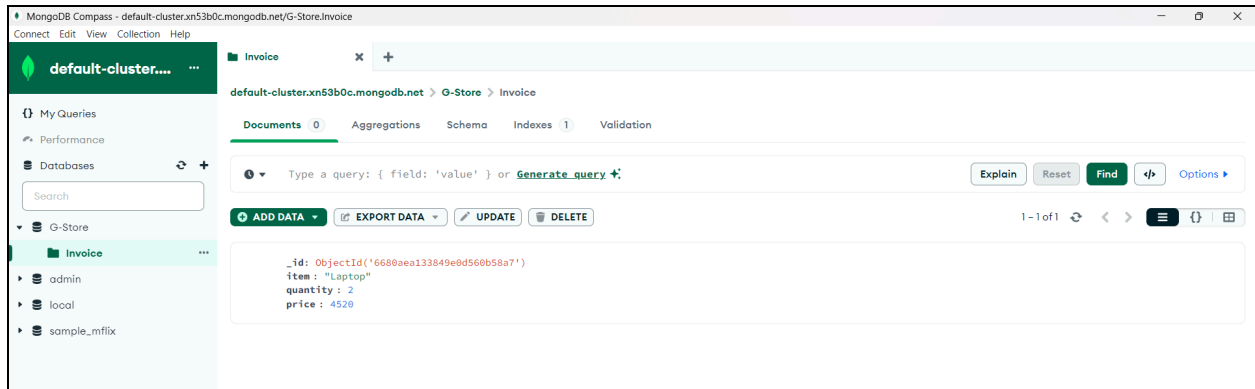


Figure 1.16: Database after update query was performed

## DELETE

```

1 usage
64 private static void delete(String invoiceName){
65     Document foundItem = collection.find(eq( fieldName: "item", invoiceName)).first();
66     if (foundItem != null) {
67         ObjectId id = foundItem.getObjectId( key: "_id");
68         collection.deleteOne(eq( fieldName: "_id", id));
69         System.out.println("Item deleted successfully!");
70     } else {
71         System.out.println("Item not found, so it can't be deleted!");
72     }
73 }
74 }
75

```

Figure 1.17: Delete Query Code

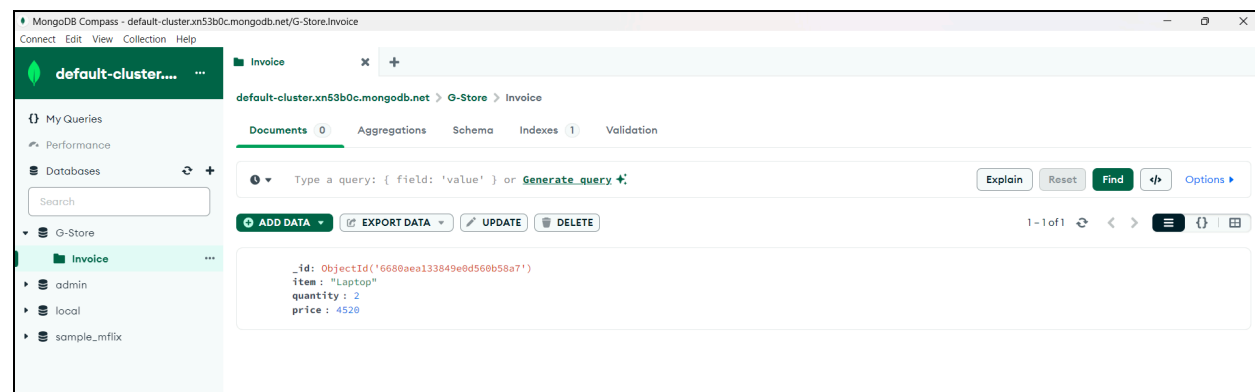


Figure 1.18: Database before delete query was performed

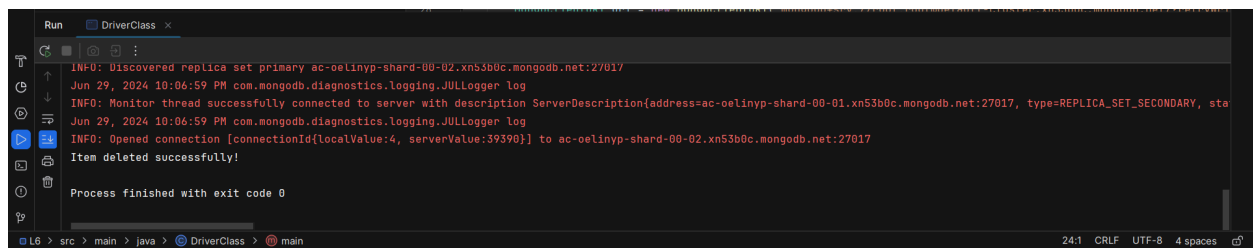


Figure 1.19: Run Delete Query

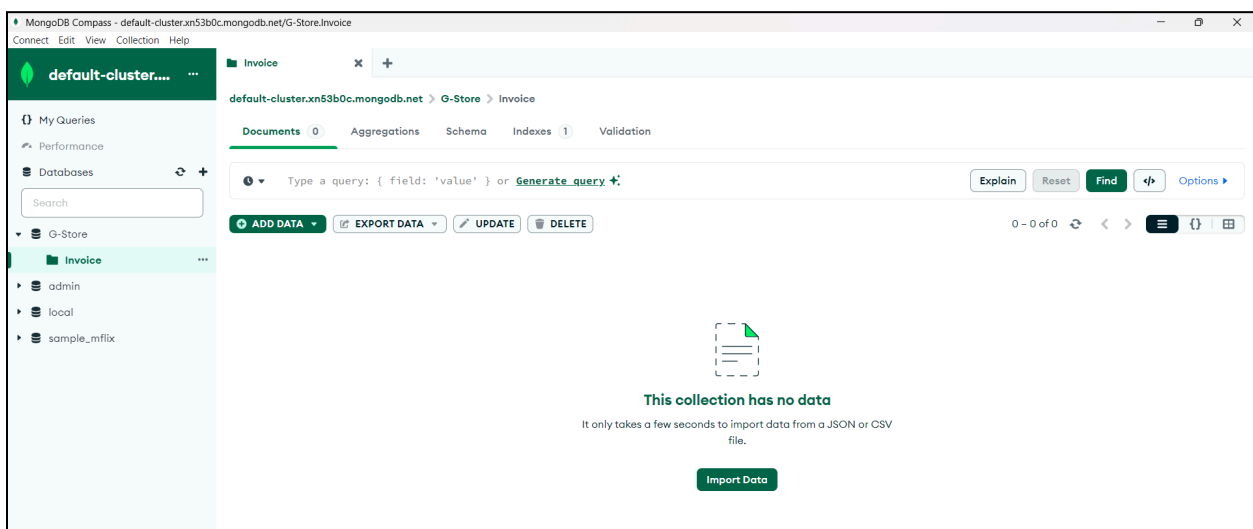


Figure 1.20: Database after delete query was performed

## Task 2: Neo4j

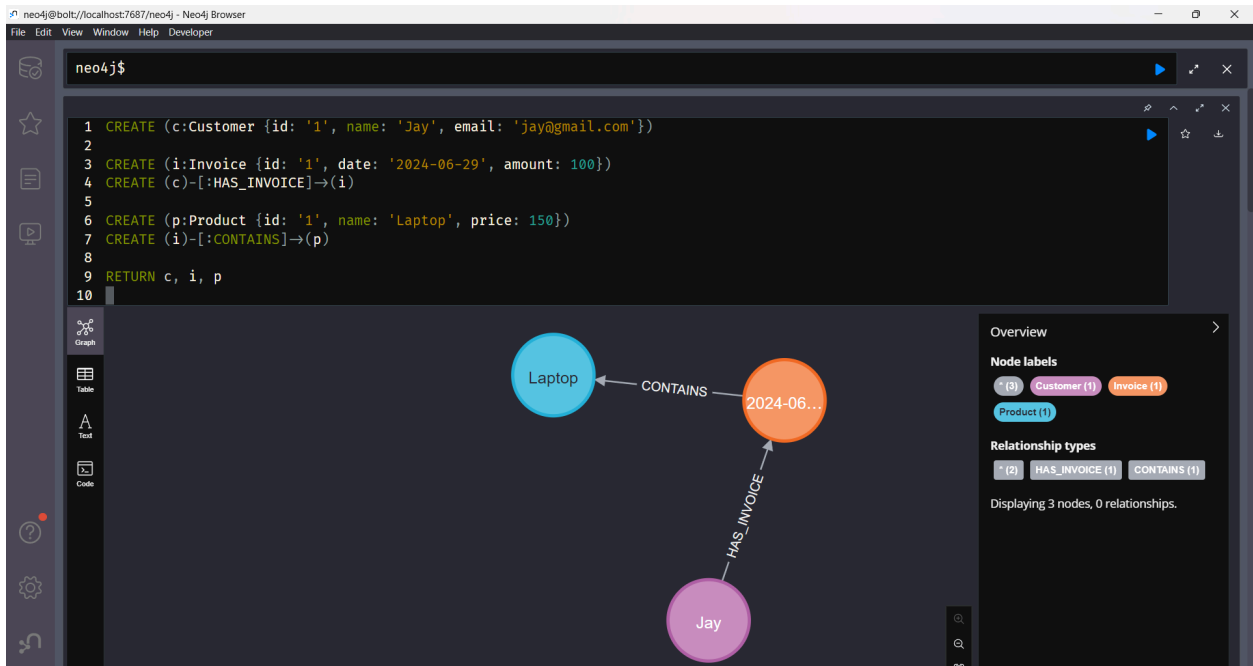


Figure 2.1: Create a graph

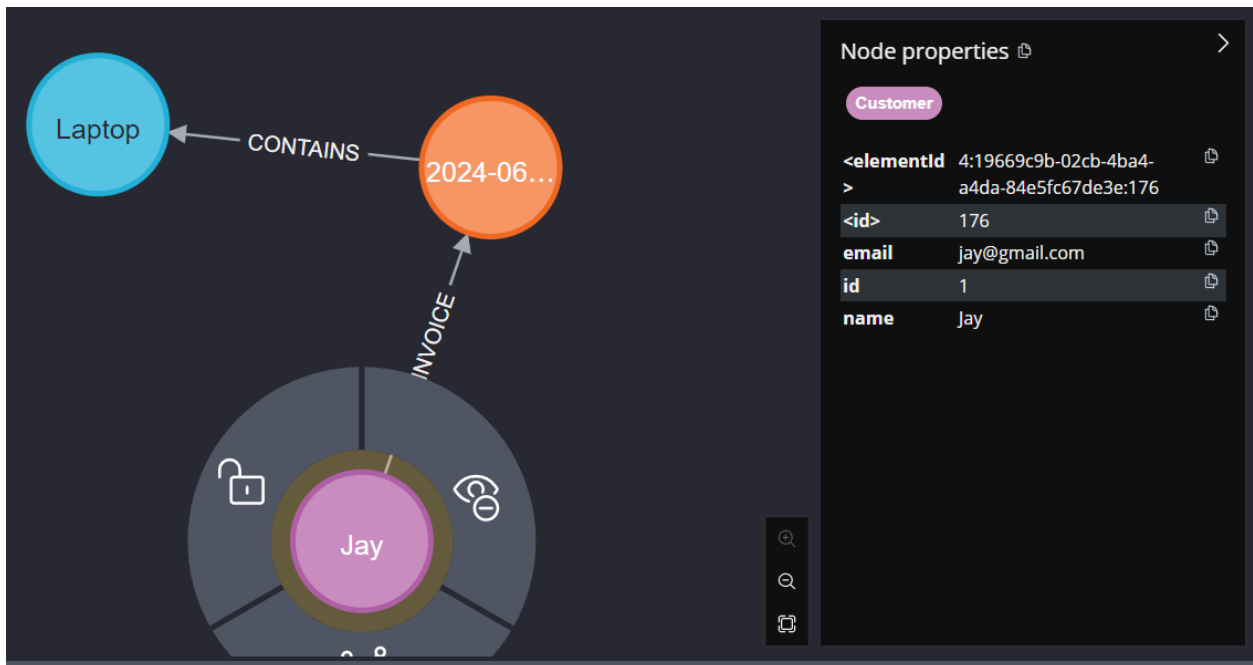


Figure 2.2: Show properties of the "Customer" node

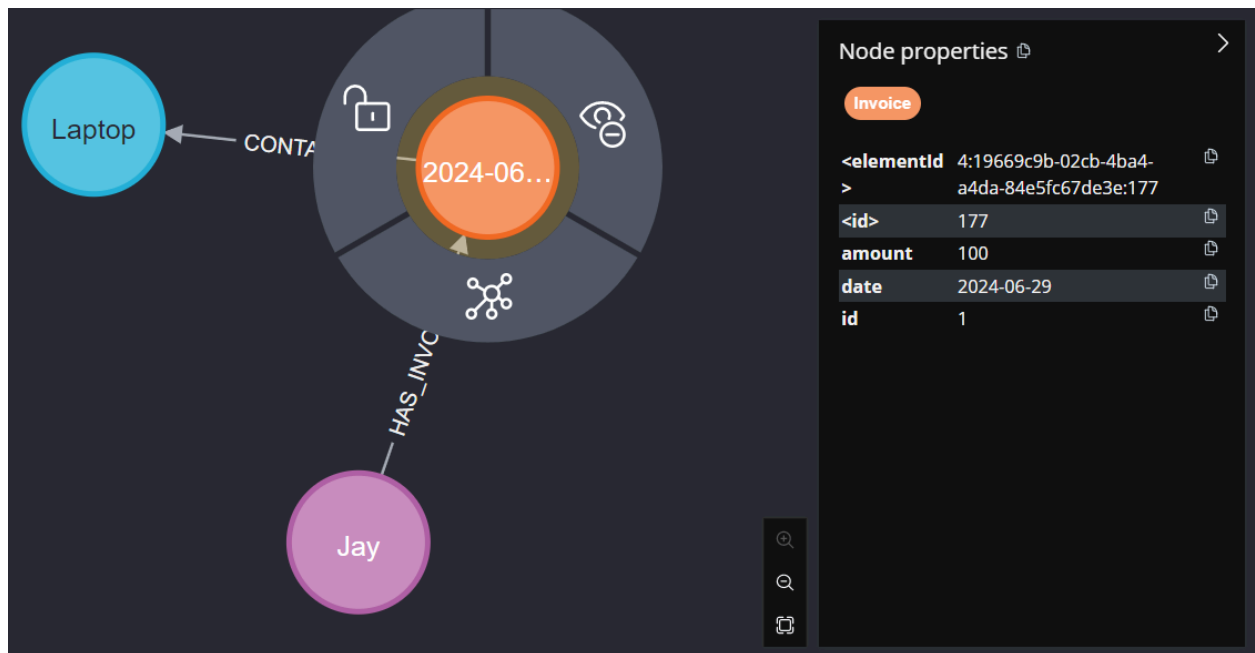


Figure 2.3: Show properties of the “Invoice” node

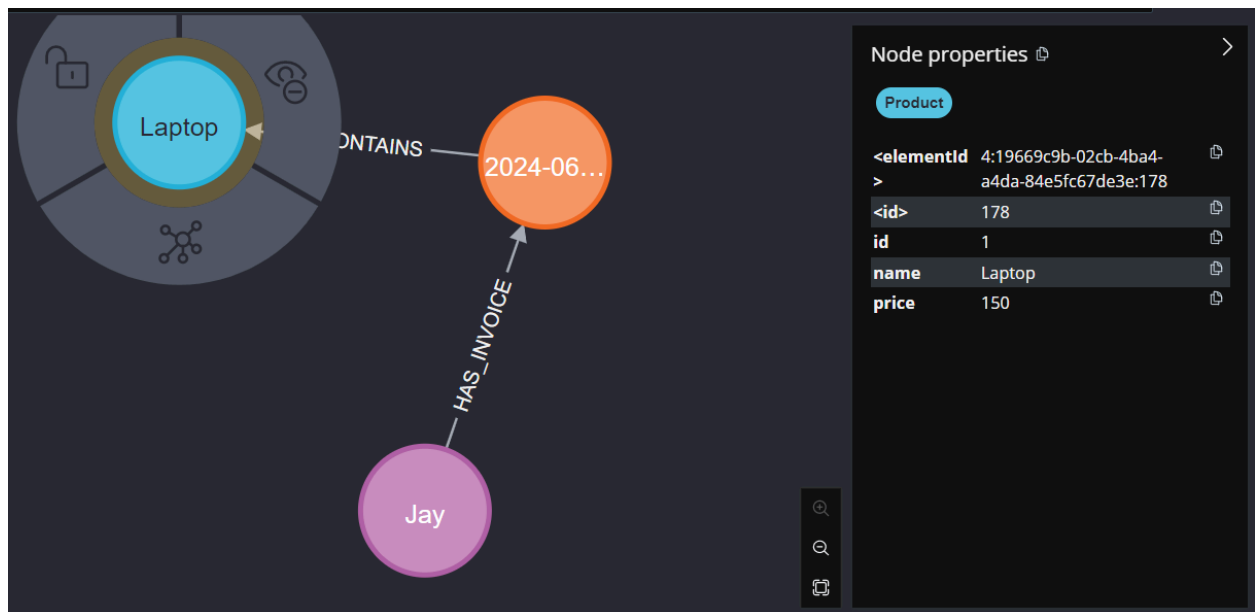


Figure 2.4: Show properties of the “Product” node