Question 1.1. Inside a new database (name it Netflix), create a collection (name it latestdata) whose schema and data values are given in the table below: (Insert these records using Mongo Shell)

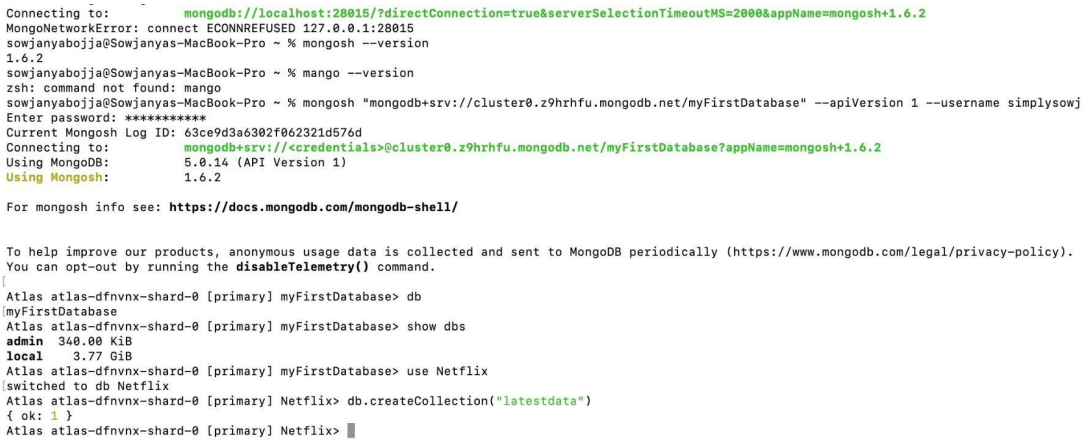
Syntax for using New database Netflix:

use Netflix

Syntax for create a collection (name it latestdata):

db.createCollection("latestdata")

Screenshot:



Syntax for inserting records:

db.latestdata.insertMany([{Type:"Movie",Acotor_name:"Michel",Country:"Mexico",Date_of_Release:"23-DEC-2016",
Year_of_Release:"2016"},{Type:"Movie",Acotor_name:"Gilbert",Country:"Singapor",Date_of_Release:"20-DEC-2018",Year_of_release:"2011"},{Type:"Movie",Acotor_name:"Shane",Country:"US",Date_of_Release:"16-NOV-2017",Year_of_Release:"2009"},{Type:"Movie",Acotor_name:"Robert",Country:"US",Date_of_Release:"1-jan-2020",Year_of_Release:"2008"}])

db.latestdata.insertMany([{Type:"TVShow",Show_name:"13 reasons why",Country:"US",DOR:"23-AUG-2019",YOR:"2016"},{ Type:"TVShow",Show_name:"20 minutes",Country:"Turkey",DOR:"15-Aug-2017",YOR:"2011"},{Type:"TVShow",Show_name:"13 reasons why",Country:"UK",DOR:"1-JUL-2020",YOR:"2009"},{Type:"TVShow",Show_name:"Elite",Country:"India",DOR:"1-DEC-2020",YOR:"2008"}])

Screenshot:

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.insertMany([{Type:"Movie",Acotor_name:"Michel",Country:"Mexico",Date_of_Release:"23-DEC-2016",
... Year_of_Release:"2016"},{Type:"Movie",Acotor_name:"Gilbert",Country:"Singapor",Date_of_Release:"20-DEC-2018",Year_of_release:"2011"},{Type:"Movie",Acotor_name:"Shane",Country:"
S",Date_of_Release:"16-NOV-2017",Year_of_Release:"2009"},{Type:"Movie",Acotor_name:"Robert",Country:"US",Date_of_Release:"1-jan-2020",Year_of_Release:"2008"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63cf3ceb6302f062321d577a"),
    '1': ObjectId("63cf3ceb6302f062321d577b"),
    '2': ObjectId("63cf3ceb6302f062321d577c"),
    '3': ObjectId("63cf3ceb6302f062321d577d")
  }
}
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>  db.latestdata.insertMany([{Type:"TVShow",Show_name:"13 reasons why",Country:"US",DOR:"23-AUG-2019",YOR:"2016"},{ Type:"TVShow",Show_n
me:"20 minutes",Country:"Turkey",DOR:"15-Aug-2017",YOR:"2011"},{Type:"TVShow",Show_name:"13 reasons why",Country:"UK",DOR:"1-JUL-2020",YOR:"2009"},{Type:"TVShow",Show_name:"Elite",
ountry:"India",DOR:"1-DEC-2020",YOR:"2008"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63cf3e7a6302f062321d577e"),
    '1': ObjectId("63cf3e7a6302f062321d577f"),
    '2': ObjectId("63cf3e7a6302f062321d5780"),
    '3': ObjectId("63cf3e7a6302f062321d5781")
  }
}
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question 1.2. Write a query to return all the data of TV shows and movies.:

db.latestdata.find()

Screenshot:

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.find()
[
  {
    _id: ObjectId("63cf3ceb6302f062321d577a"),
    Type: 'Movie',
    Acotor_name: 'Michel',
    Country: 'Mexico',
    Date_of_Release: '23-DEC-2016',
    Year_of_Release: '2016'
  },
  {
    _id: ObjectId("63cf3ceb6302f062321d577b"),
    Type: 'Movie',
    Acotor_name: 'Gilbert',
    Country: 'Singapor',
    Date_of_Release: '20-DEC-2018',
    Year_of_release: '2011'
  },
  {
    _id: ObjectId("63cf3ceb6302f062321d577c"),
    Type: 'Movie',
    Acotor_name: 'Shane',
    Country: 'US',
    Date_of_Release: '16-NOV-2017',
    Year_of_Release: '2009'
  },
  {
    _id: ObjectId("63cf3ceb6302f062321d577d"),
    Type: 'Movie',
    Acotor_name: 'Robert',
    Country: 'US',
    Date_of_Release: '1-jan-2020',
    Year_of_Release: '2008'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d577e"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'US',
    DOR: '23-AUG-2019',
    YOR: '2016'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d577f"),
    Type: 'TVShow',
    Show_name: '20 minutes',
    Country: 'Turkey',
    DOR: '15-Aug-2017',
    YOR: '2011'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d5780"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'UK',
    DOR: '1-JUL-2020',


    Type: 'Movie',
    Acotor_name: 'Gilbert',
    Country: 'Singapor',
    Date_of_Release: '20-DEC-2018',
    Year_of_release: '2011'
  },
  {
    _id: ObjectId("63cf3ceb6302f062321d577c"),
    Type: 'Movie',
    Acotor_name: 'Shane',
    Country: 'US',
    Date_of_Release: '16-NOV-2017',
    Year_of_Release: '2009'
  },
  {
    _id: ObjectId("63cf3ceb6302f062321d577d"),
    Type: 'Movie',
    Acotor_name: 'Robert',
    Country: 'US',
    Date_of_Release: '1-jan-2020',
    Year_of_Release: '2008'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d577e"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'US',
    DOR: '23-AUG-2019',
    YOR: '2016'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d577f"),
    Type: 'TVShow',
    Show_name: '20 minutes',
    Country: 'Turkey',
    DOR: '15-Aug-2017',
    YOR: '2011'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d5780"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'UK',
    DOR: '1-JUL-2020',
    YOR: '2009'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d5781"),
    Type: 'TVShow',
    Show_name: 'Elite',
    Country: 'India',
    DOR: '1-DEC-2020',
    YOR: '2008'
  }
]
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question1.3. Write a query to return data about 13 Reasons Why:

Syntax:

db.latestdata.find({Show_name:"13 reasons why"})

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.find({Show_name:"13 reasons why"})
[
  {
    _id: ObjectId("63cf3e7a6302f062321d577e"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'US',
    DOR: '23-AUG-2019',
    YOR: '2016'
  },
  {
    _id: ObjectId("63cf3e7a6302f062321d5780"),
    Type: 'TVShow',
    Show_name: '13 reasons why',
    Country: 'UK',
    DOR: '1-JUL-2020',
    YOR: '2009'
  }
]
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question 1.4. Update the release year of the movie starring Robert from 2008 to 2020.

Syntax:

db.latestdata.updateOne({ Acotor_name: "Robert" }, { $set: {Year_of_Release:"2020" } })

Screenshot:

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.find({Acotor_name:"Robert"})
[
  {
    _id: ObjectId("63cf3ceb6302f062321d577d"),
    Type: 'Movie',
    Acotor_name: 'Robert',
    Country: 'US',
    Date_of_Release: '1-jan-2020',
    Year_of_Release: '2008'
  }
]
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.updateOne( { Acotor_name: "Robert" }, { $set: {Year_of_Release:"2020" } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.find({Acotor_name:"Robert"})
[
  {
    _id: ObjectId("63cf3ceb6302f062321d577d"),
    Type: 'Movie',
    Acotor_name: 'Robert',
    Country: 'US',
    Date_of_Release: '1-jan-2020',
    Year_of_Release: '2020'
  }
]
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question 1.5. Delete the data of the movies that were released in 2020.

Syntax:

db.latestdata.deleteMany({ Year_of_Release: "2020" })

Screenshot:

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.find({Year_of_Release:"2020"})
[
  {
    _id: ObjectId("63cf3ceb6302f062321d577d"),
    Type: 'Movie',
    Acotor_name: 'Robert',
    Country: 'US',
    Date_of_Release: '1-jan-2020',
    Year_of_Release: '2020'
  }
]
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.deleteMany({ Year_of_Release: "2020" })
{ acknowledged: true, deletedCount: 1 }
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.deleteMany({ Year_of_Release: "2020" })
{ acknowledged: true, deletedCount: 0 }
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question 1.6. Delete the collection through the shell.

Syntax:

db.latestdata.drop()

```
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> show collections
latestdata
posts
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db
Netflix
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> db.latestdata.drop()
true
Atlas atlas-dfnvnx-shard-0 [primary] Netflix> show collections
posts
Atlas atlas-dfnvnx-shard-0 [primary] Netflix>
```

Question 2:

Suppose the owner of an online clothing website wants to maintain data of its customers and the products they buy from his website. A single customer can buy multiple products and can return the products purchased. Insert the following data and perform the operations given below using HBase.

2.1:
Create a table named ‘Customer’ with two column families ‘Personal_Info’ and ‘Product_Info’. The version count for Personal_Info should be 2 and Product_Info should be 4 and should be changed in case needed. Customer_Number is the rowkey of the table.

Syntax:

```
create 'Customer', 'Personal_Info', {VERSIONS => 2}, 'Product_Info', {VERSIONS => 4}
```

2.2:
Add the following data into the created table ‘Customer’ ● 001 (rowkey) Personal_Info:Customer_Name = ‘Sam’ Personal_Info:City = ‘Pune’ Personal_Info:Contact = 9877656789 Personal_Info:Contact = 8765421345 Product_Info:Name = ‘Jeans’ Product_Info:Size= ‘M’ Product_Info:Price = 1200 ● 002 (rowkey) Personal_Infor:Customer_Name = ‘Rahul’ Personal_Infor:City = ‘Mumbai’ Personal_Info:Contact = 9844215263 Product_Info:Name = ‘T-shirt’ Product_Info:Size= ‘L’ Product_Info:Price = 800 Product_Info:Name = ‘Shirt’ Product_Info:Code= S Product_Info:Price = 1000 Product_Info:Name = ‘Jeans’ Product_Info:Size= ‘L’ Product_Info:Price = 1200 ● 003 (rowkey) Personal_Infor:Customer_Name = ‘Shalini’ Personal_Infor:City = ‘Patiala’ Personal_Info:Contact = 8765423456 Product_Info:Name = ‘Dress’ © Product_Info:Size = ‘S’ Product_Info:Price = 2300 Product_Info:Name = ‘T-shirt’ Product_Info:Size = ‘S’ Product_Info:Price = 800 ● 004 (rowkey) Personal_Infor:Customer_Name = ‘Sakshi’ Personal_Infor:City = ‘Ludhiana’ Personal_Info:Contact = 6789023456 Product_Info:Name = ‘Top’ Product_Info:Size = ‘M’ Product_Info:Price = 1300 Product_Info:Name = ‘T-shirt’ Product_Info:Size = ‘M’ Product_Info:Price = 800 Product_Info:Name = ‘Shirt’ Product_Info:Size= ‘S’ Product_Info:Price = 1000 Product_Info:Name = ‘Jeans’ Product_Info:Size= ‘S’ Product_Info:Price = 1200 ● 005 (rowkey) Personal_Infor:Customer_Name = ‘Riya’ Personal_Infor:City = ‘Delhi’ Personal_Info:Contact = 7785823459 Product_Info:Name = ‘Top’ Product_Info:SIze = ‘S’ Product_Info:Price = 1300

Syntax:

```
put 'Customer', '001', 'Personal_Info:Customer_Name', 'Sam'
put 'Customer', '001', 'Personal_Info:City', 'Pune'
put 'Customer', '001', 'Personal_Info:Contact', '9877656789'
put 'Customer', '001', 'Personal_Info:Contact', '8765421345'
put 'Customer', '001', 'Product_Info:Name', 'Jeans'
put 'Customer', '001', 'Product_Info:Size', 'M'
put 'Customer', '001', 'Product_Info:Price', '1200'
put 'Customer', '002', 'Personal_Info:Customer_Name', 'Rahul'
put 'Customer', '002', 'Personal_Info:City', 'Mumbai'
put 'Customer', '002', 'Personal_Info:Contact', '9844215263'
put 'Customer', '002', 'Product_Info:Name', 'T-shirt'
put 'Customer', '002', 'Product_Info:Size', 'L'
put 'Customer', '002', 'Product_Info:Price', '800'
put 'Customer', '002', 'Product_Info:Name', 'Shirt'
put 'Customer', '002', 'Product_Info:Code', 'S'
put 'Customer', '002', 'Product_Info:Price', '1000'
put 'Customer', '002', 'Product_Info:Name', 'Jeans'
put 'Customer', '002', 'Product_Info:Size', 'L'
put 'Customer', '002', 'Product_Info:Price', '1200'
put 'Customer', '003', 'Personal_Info:Customer_Name', 'Shalini'
put 'Customer', '003', 'Personal_Info:City', 'Patiala'
put 'Customer', '003', 'Personal_Info:Contact', '8765423456'
put 'Customer', '003', 'Product_Info:Name', 'Dress'
put 'Customer', '003', 'Product_Info:Size', 'S'
put 'Customer', '003', 'Product_Info:Price', '2300'
put 'Customer', '003', 'Product_Info:Name', 'T-shirt'
put 'Customer', '003', 'Product_Info:Size', 'S'
put 'Customer', '003', 'Product_Info:Price', '800'
put 'Customer', '004', 'Personal_Info:Customer_Name', 'Sakshi'
put 'Customer', '004', 'Personal_Info:City', 'Ludhiana'
put 'Customer', '004', 'Personal_Info:Contact', '6789023456'
put 'Customer', '004', 'Product_Info:Name', 'Top'
put 'Customer', '004', 'Product_Info:Size', 'M'
put 'Customer', '004', 'Product_Info:Price', '1300'
put 'Customer', '004', 'Product_Info:Name', 'T-shirt'
put 'Customer', '004', 'Product_Info:Size', 'M'
put 'Customer', '004', 'Product_Info:Price', '800'
put 'Customer', '004', 'Product_Info:Name', 'Shirt'
put 'Customer', '004', 'Product_Info:Size', 'S'
put 'Customer', '004', 'Product_Info:Price', '1000'
put 'Customer', '004', 'Product_Info:Name', 'Jeans'
put 'Customer', '004', 'Product_Info:Size', 'S'
put 'Customer', '004', 'Product_Info:Price', '1200'
```

```
put 'Customer', '005', 'Personal_Info:Customer_Name', 'Riya'
put 'Customer', '005', 'Personal_Info:City', 'Delhi'
put 'Customer', '005', 'Personal_Info:Contact', '7785823459'
put 'Customer', '005', 'Product_Info:Name', 'Top'
put 'Customer', '005', 'Product_Info:Size', 'S'
put 'Customer', '005', 'Product_Info:Price', '1300'
```

2.3
The customer named ‘Sam’ wants to return all the products purchased. Delete the data associated with the customer named ‘Sam’.

Syntax:

```
scan 'Customer', {FILTER =>
"SingleColumnValueFilter('Personal_Info','Customer_Name',=,'binary:Sam')"}

deleteall 'Customer', 'rowkey'
```

2.4
The website owner wants to change the bandwidth of the products bought by customers from 4 to 6. Use a Shell command to change the version to 6.

```
alter 'Customer', {NAME => 'Product_Info', VERSIONS => 6}

describe 'Customer'
```

2.5
Riya wants to buy a pair of jeans of size ‘M’ of price 1200. Add the relevant product information.

```
put 'Customer', '005', 'Product_Info:Name', 'Jeans'
put 'Customer', '005', 'Product_Info:Size', 'M'
put 'Customer', '005', 'Product_Info:Price', '1200'
```

2.6:Calculate the number of customers.

count 'Customer'

2.7Fetch the details of customers who have bought items with a price less than or equal to 1000.

scan 'Customer', {FILTER => "SingleColumnValueFilter('Product_Info','Price',<=,1000)"}

For specific version:

scan 'Customer', {FILTER => "SingleColumnValueFilter('Product_Info','Price',<=,1000)", VERSIONS => 1}

scan 'Customer', {COLUMNS => 'Personal_Info:Customer_Name'}

For specific version:
scan 'Customer', {COLUMNS => 'Personal_Info:Customer_Name', VERSIONS => 1}

2.8 Fetch the details of the products sold.

scan 'Customer', {COLUMNS => ['Product_Info:Name', 'Product_Info:Size', 'Product_Info:Price']}

scan 'Customer', {COLUMNS => ['Product_Info:Name', 'Product_Info:Size', 'Product_Info:Price'], VERSIONS => 1}

Question 3 Create a GraphDB using Neo4J desktop client and perform the following operations:

Create 4 customer nodes with the data given below. Customer Name Email Id Age Daniel Johnston dan_j@example.com 25 Alex McGyver mcgalex@example.com 25 Allison York ally_york1@example.com 31 Joe Baxton joeee_baxton@example.com 24

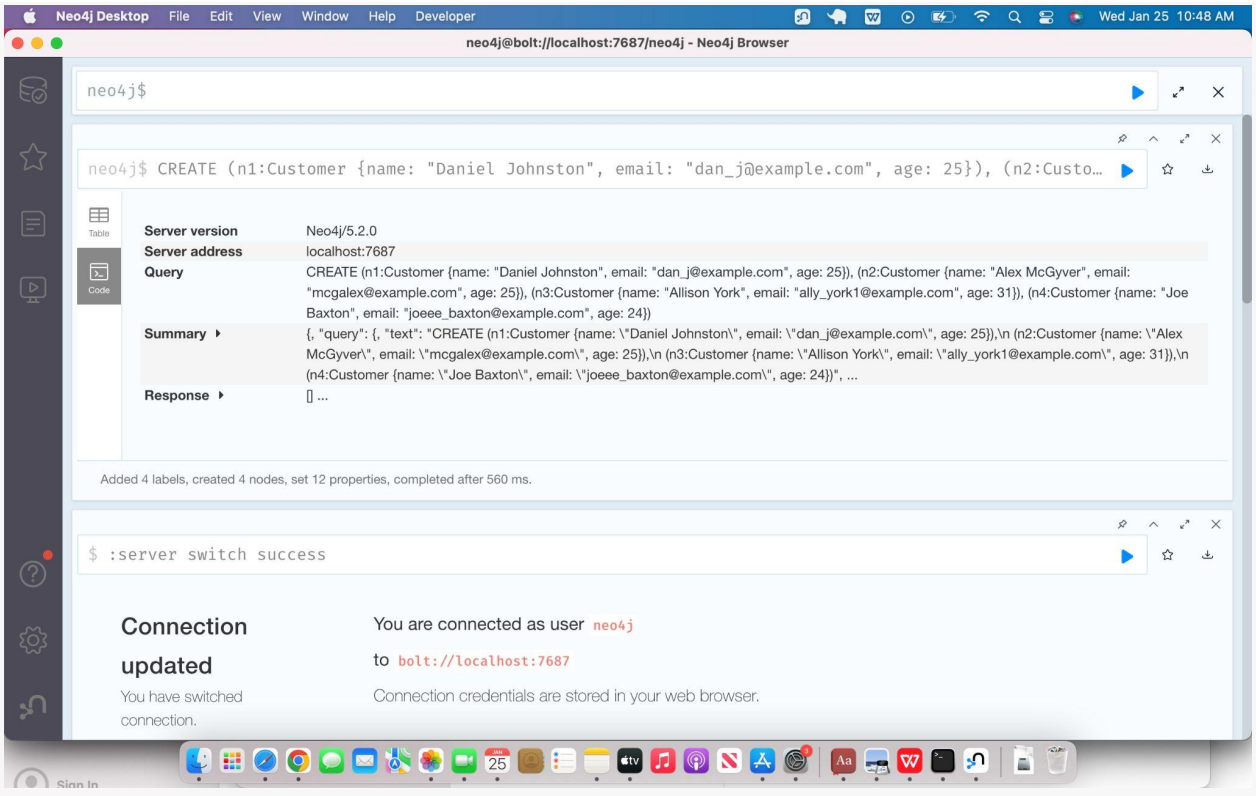
To create 4 customer nodes with the data given, you can use the Cypher query:

CREATE (n1:Customer {name: "Daniel Johnston", email: "dan_j@example.com", age: 25}),

(n2:Customer {name: "Alex McGyver", email: "mcgalex@example.com", age: 25}),
(n3:Customer {name: "Allison York", email: "ally_york1@example.com", age: 31}),
(n4:Customer {name: "Joe Baxton", email: "joeee_baxton@example.com", age: 24})

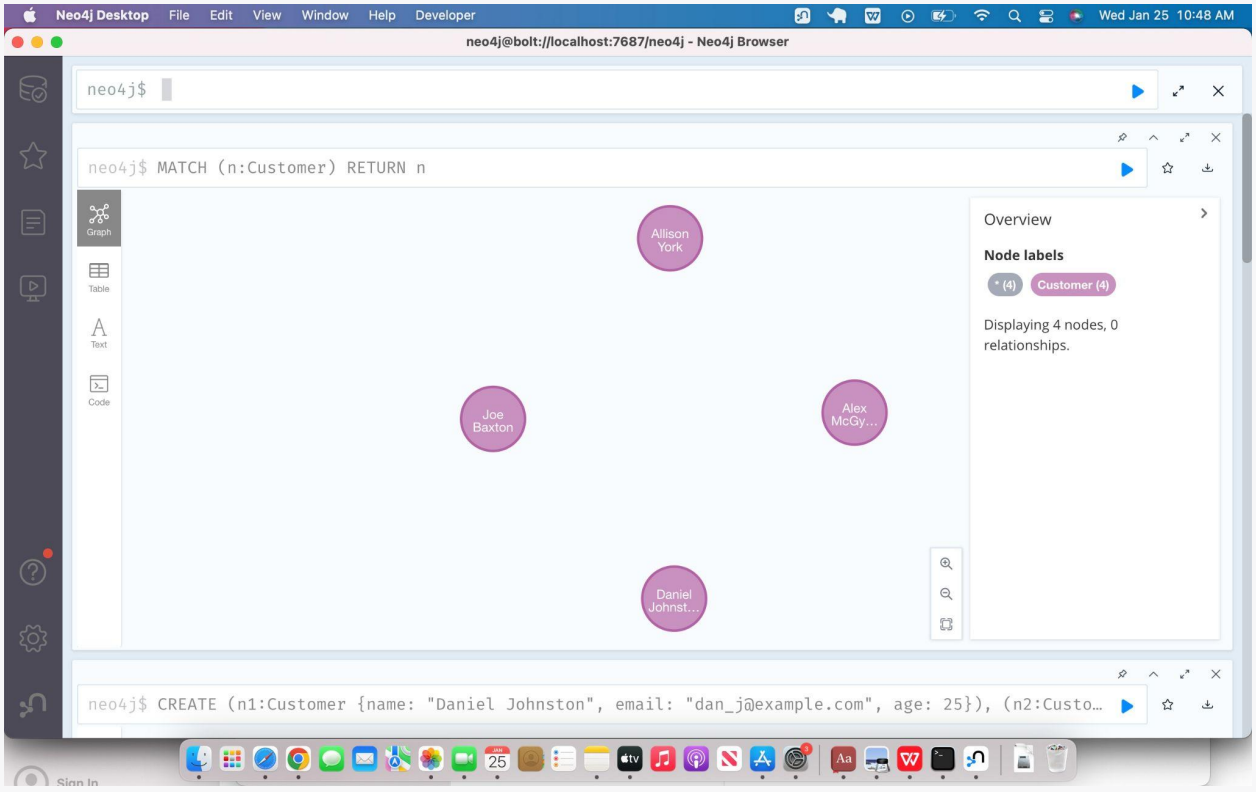
This will create 4 nodes with label Customer and properties name, email, and age with the corresponding values.

Screenshot:



You can then query the newly created nodes using Cypher query:
MATCH (n:Customer)
RETURN n

This will return all the customer nodes with their properties.

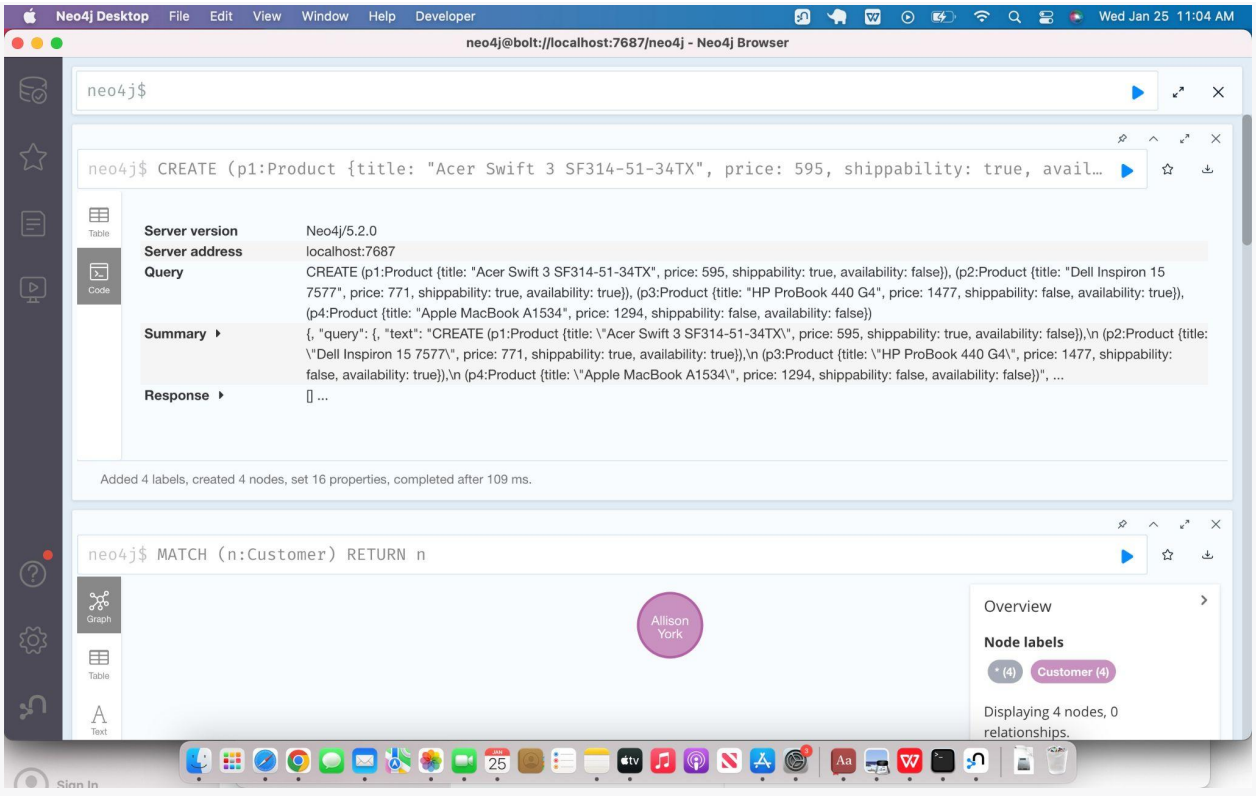


2. Create 4 product nodes with data of laptops as given in the table below and establish the relationship named 'BOUGHT' between the customer and product nodes. Product Title Price Shippabilit y Availability Customers who brought it Acer Swift 3 SF314-51-34TX 595 True False Alex McGyver Dell Inspiron 15 7577 771 True True Joe Baxton HP ProBook 440 G4 1477 False True Allison York Apple MacBook A1534 12 1294 False False Daniel Johnston

To create 4 product nodes with the given data and establish the relationship 'BOUGHT' between the customer and product nodes, you can use the Cypher query:

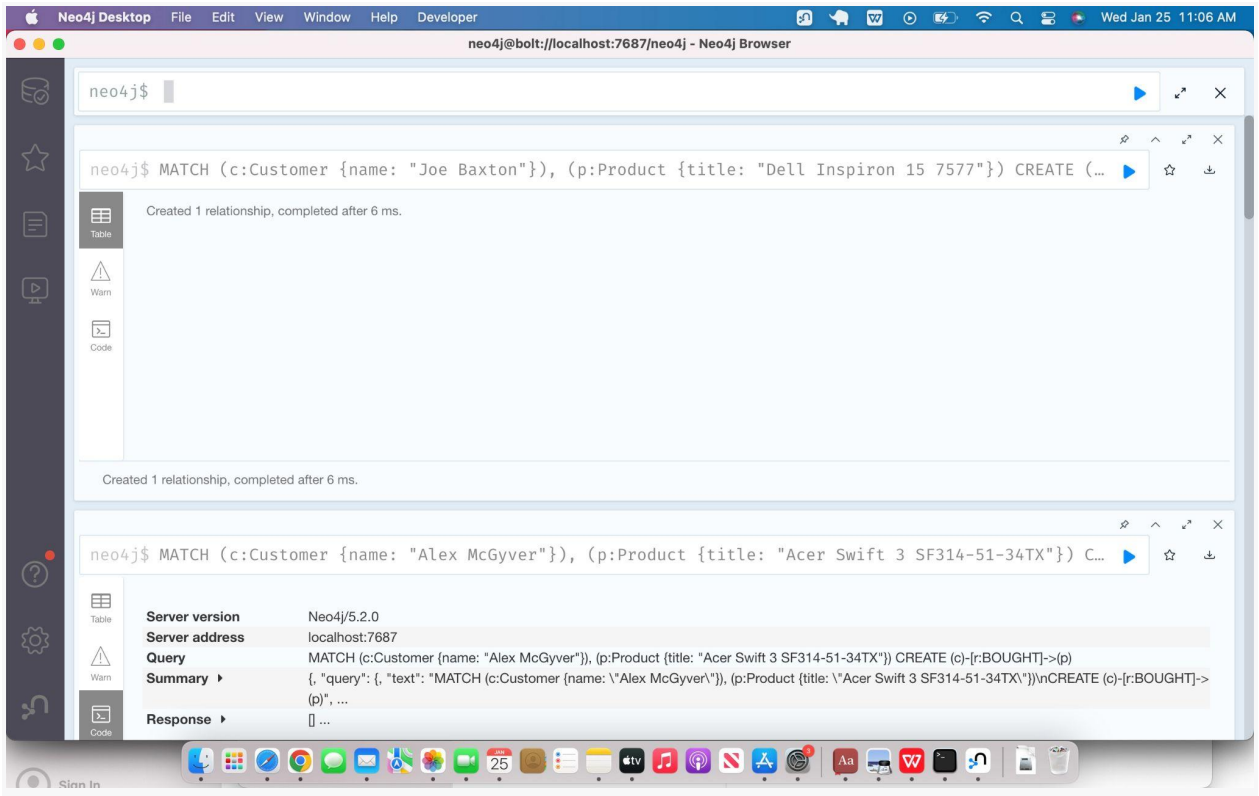
```
CREATE (p1:Product {title: "Acer Swift 3 SF314-51-34TX", price: 595, shippability: true, availability: false}),
       (p2:Product {title: "Dell Inspiron 15 7577", price: 771, shippability: true, availability: true}),
       (p3:Product {title: "HP ProBook 440 G4", price: 1477, shippability: false, availability: true}),
       (p4:Product {title: "Apple MacBook A1534", price: 1294, shippability: false, availability: false})
```

This will create 4 nodes with label Product and properties title, price, shippability and availability with the corresponding values.



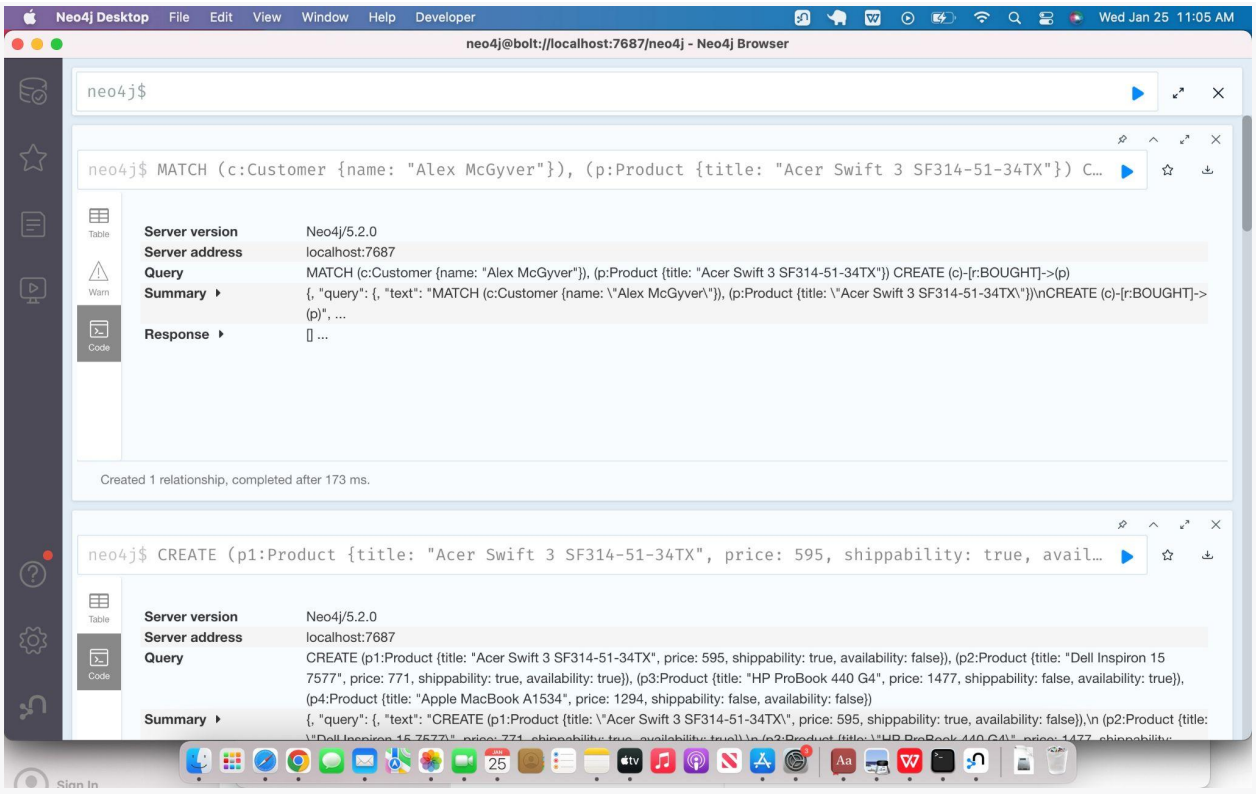
Then to create the relationships between the customer and product nodes, you can use the Cypher query:

```
MATCH (c:Customer {name: "Alex McGyver"}), (p:Product {title: "Acer Swift 3 SF314-51-34TX"})
CREATE (c)-[r:BOUGHT]->(p)
```

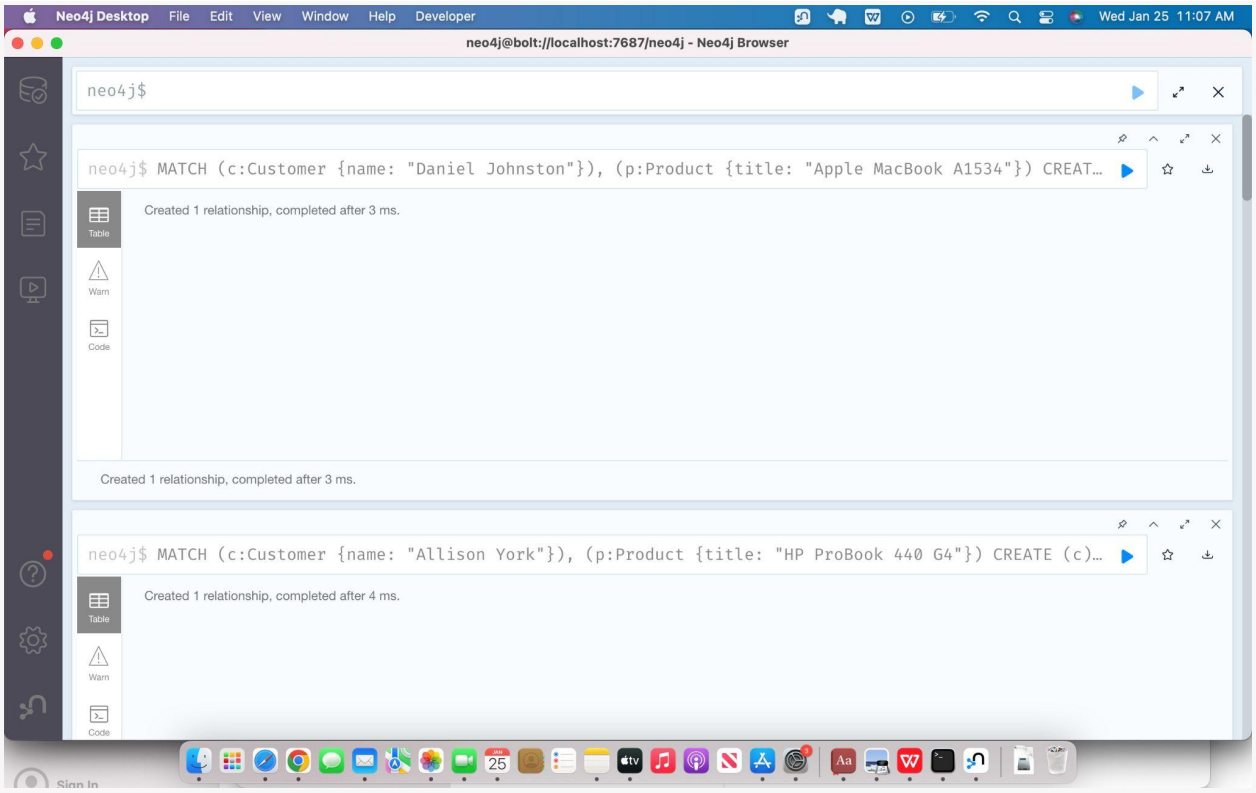



```
MATCH (c:Customer {name: "Joe Baxton"}), (p:Product {title: "Dell Inspiron 15 7577"})
CREATE (c)-[r:BOUGHT]->(p)
```

```
MATCH (c:Customer {name: "Daniel Johnston"}), (p:Product {title: "Apple MacBook A1534"})
CREATE (c)-[r:BOUGHT]->(p)
```



MATCH (c:Customer {name: "Allison York"}), (p:Product {title: "HP ProBook 440 G4"}) CREATE (c)-[r:BOUGHT]->(p)

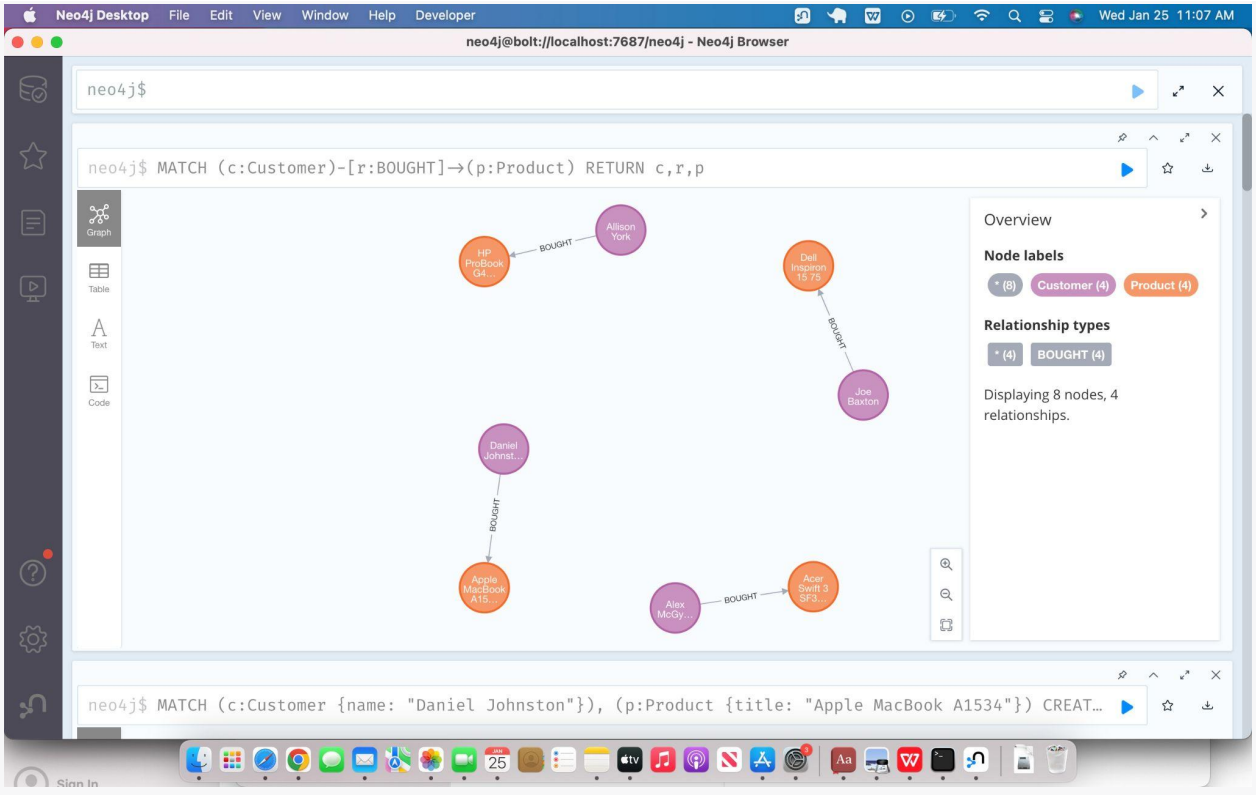


This will create a relationship 'BOUGHT' between the customer and product nodes.

You can then query the newly created relationship using Cypher query

```
MATCH (c:Customer)-[r:BOUGHT]->(p:Product)
RETURN c,r,p
```

This will return all the customer nodes with the relationship 'BOUGHT' and the product nodes they bought.



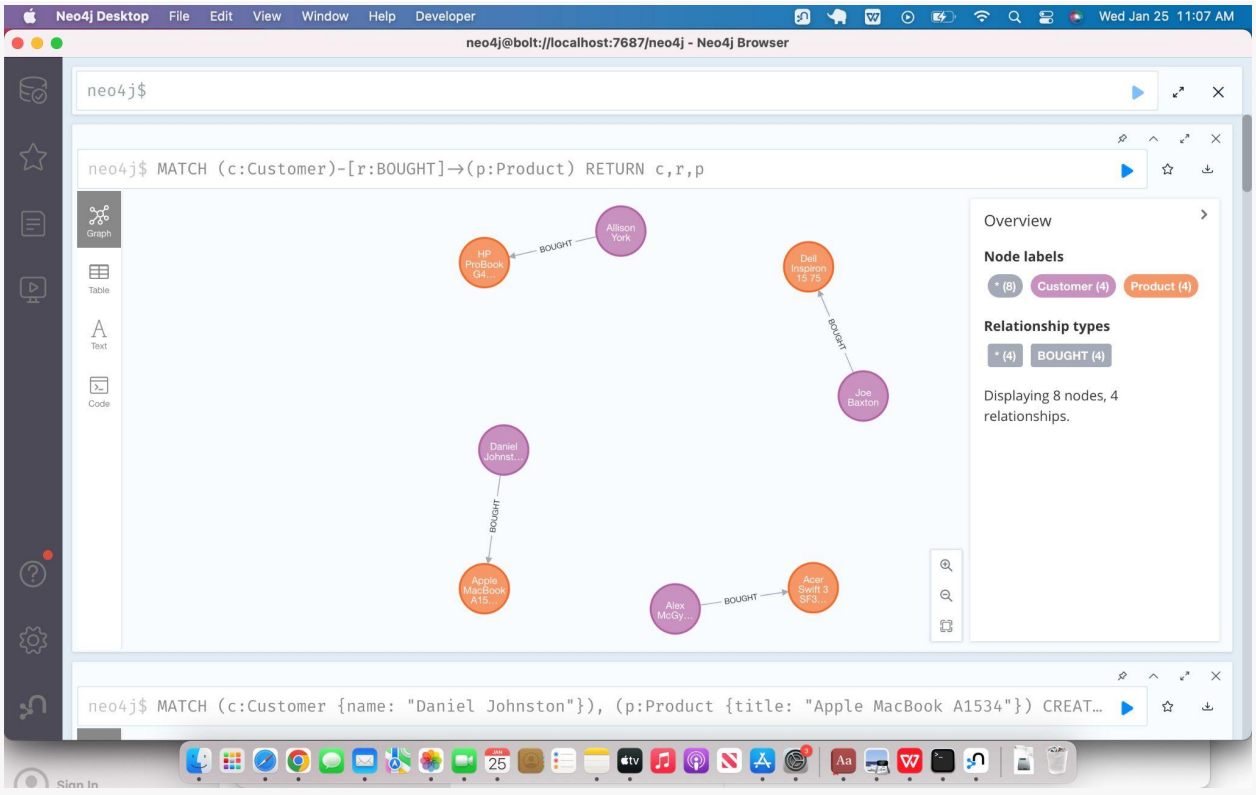
3. Now, assume customers have bought these items (one customer buys only one laptop). Convey this through the graph by displaying the above graph data.

To convey that customers have bought these items (one customer buys only one laptop) through the graph, you can use the Cypher query:

```
MATCH (c:Customer)-[r:BOUGHT]->(p:Product)
RETURN c,r,p
```

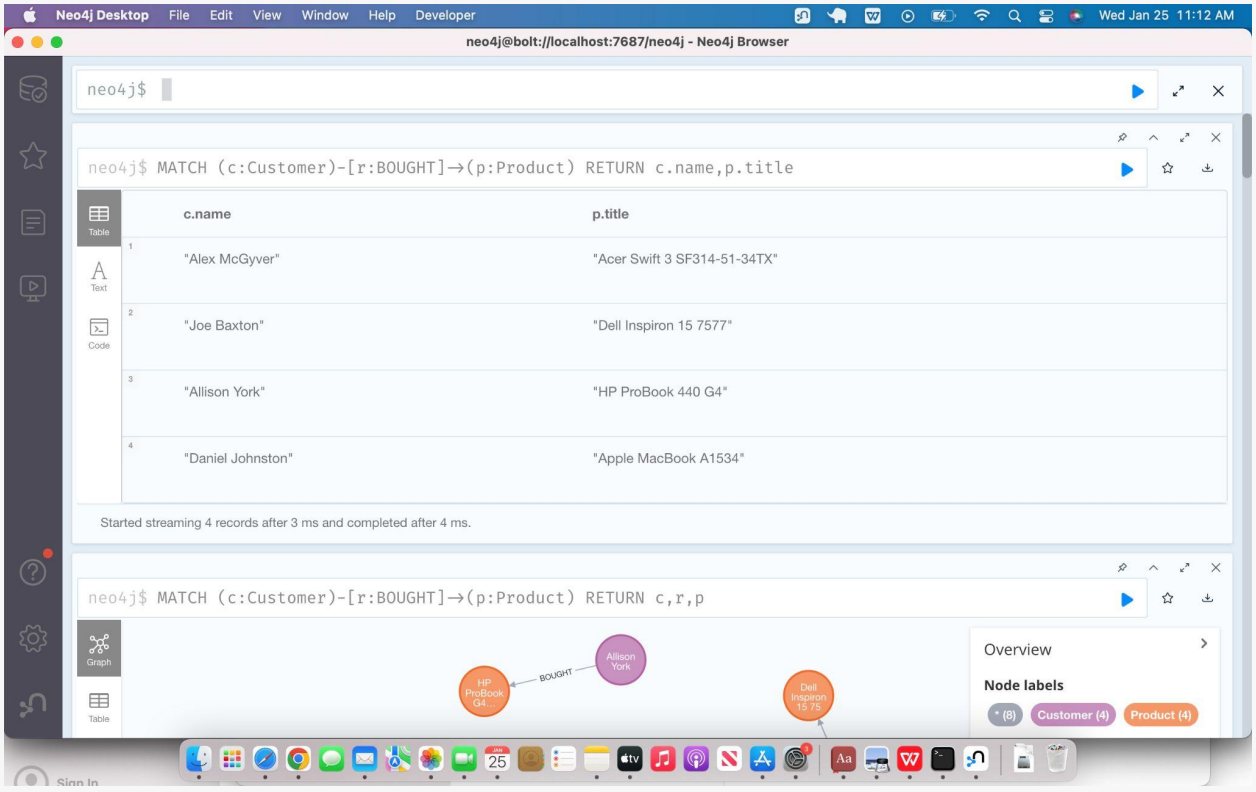
This query will match all the customer nodes that have the relationship 'BOUGHT' with the product nodes, and will return all the customer nodes, the relationship 'BOUGHT' and the product nodes they bought. This will display the graph data of customers who have bought the laptops.

```
MATCH (c:Customer)-[r:BOUGHT]->(p:Product)
RETURN c,r,p
```



You can also use Cypher query to visualize the graph

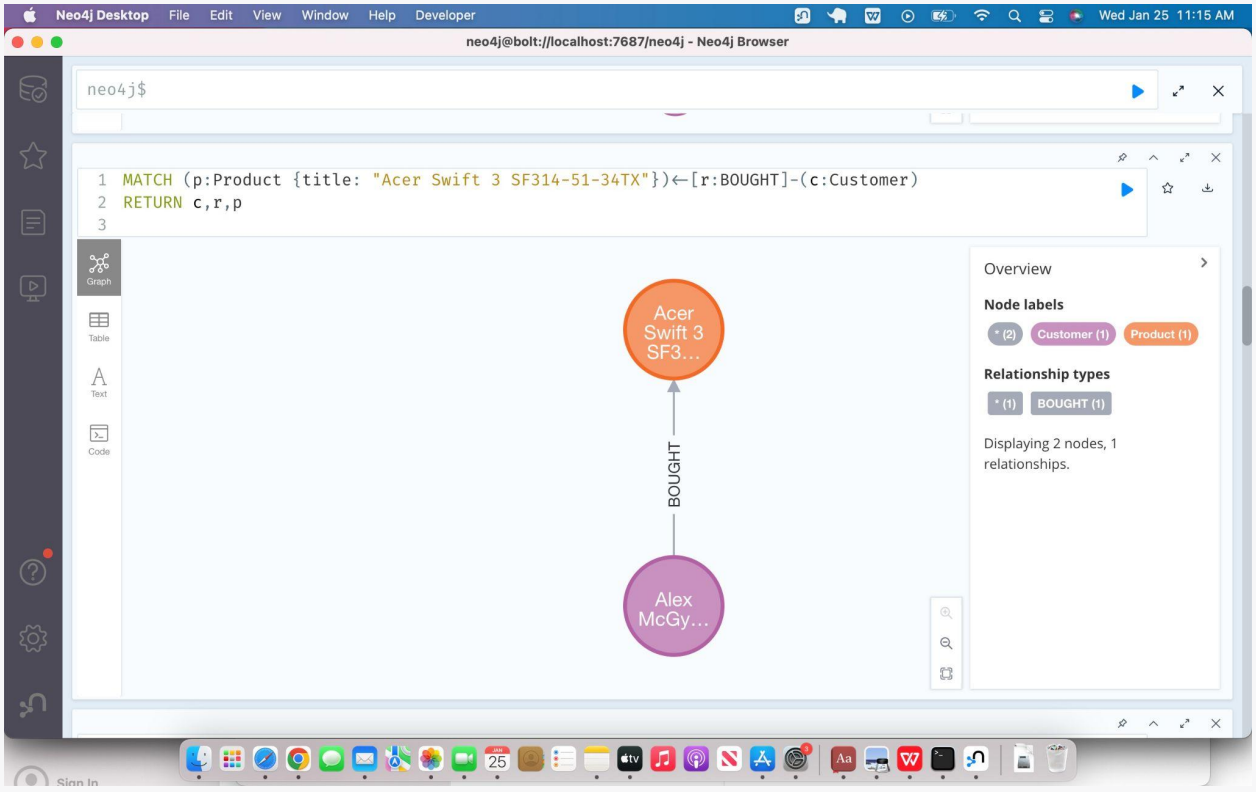
```
MATCH (c:Customer)-[r:BOUGHT]->(p:Product)
RETURN c.name,p.title
```



4. Query the graph database and find the list of customers who bought 'Acer Swift 3SF314- 51-34TX'.

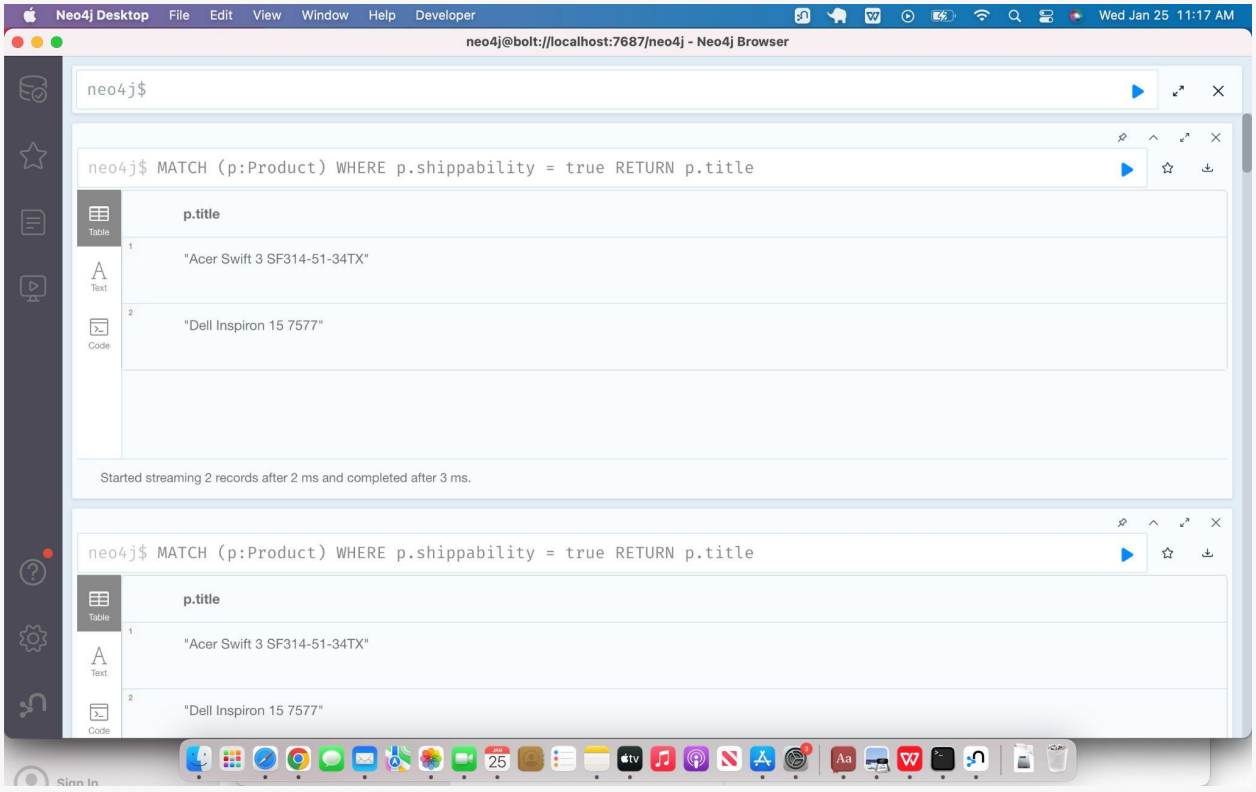
To query the graph database and find the list of customers who bought 'Acer Swift 3 SF314-51-34TX', you can use the Cypher query:

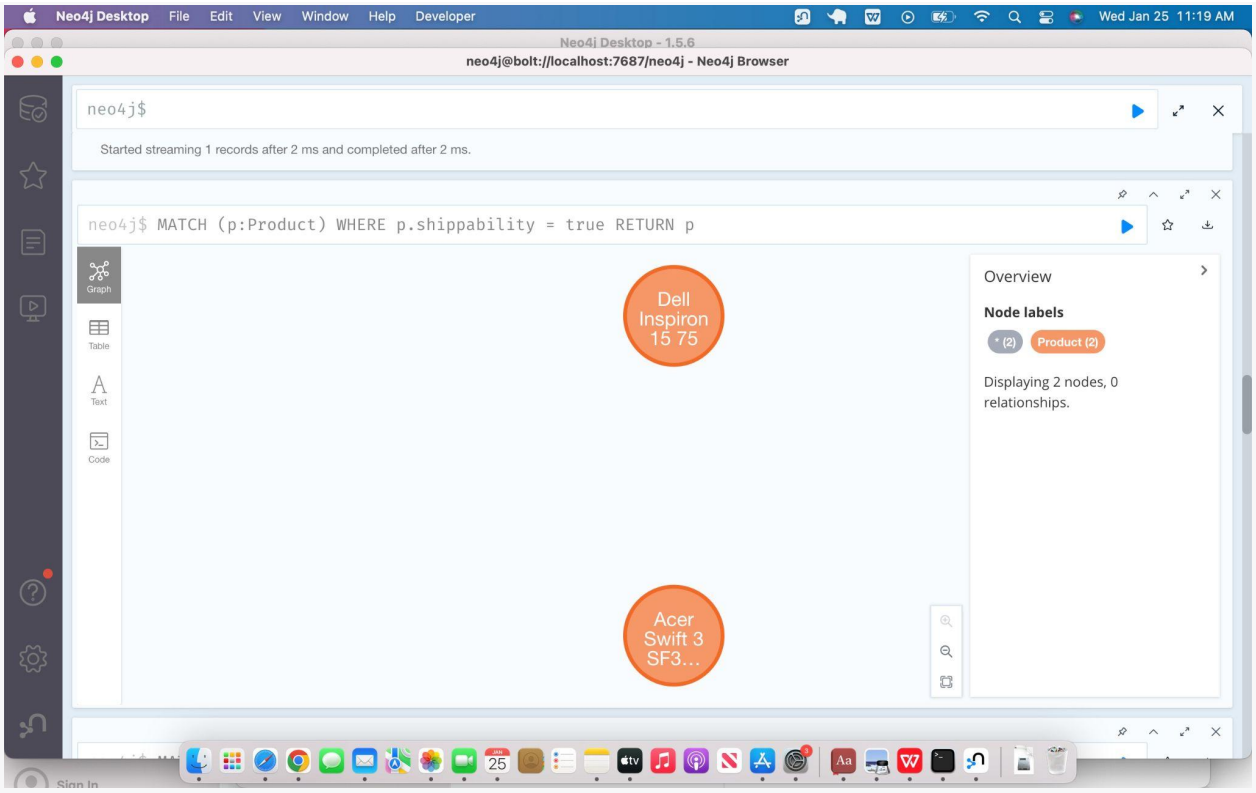
```
MATCH (p:Product {title: "Acer Swift 3 SF314-51-34TX"})<-[r:BOUGHT]-(c:Customer)
RETURN c,r,p
```



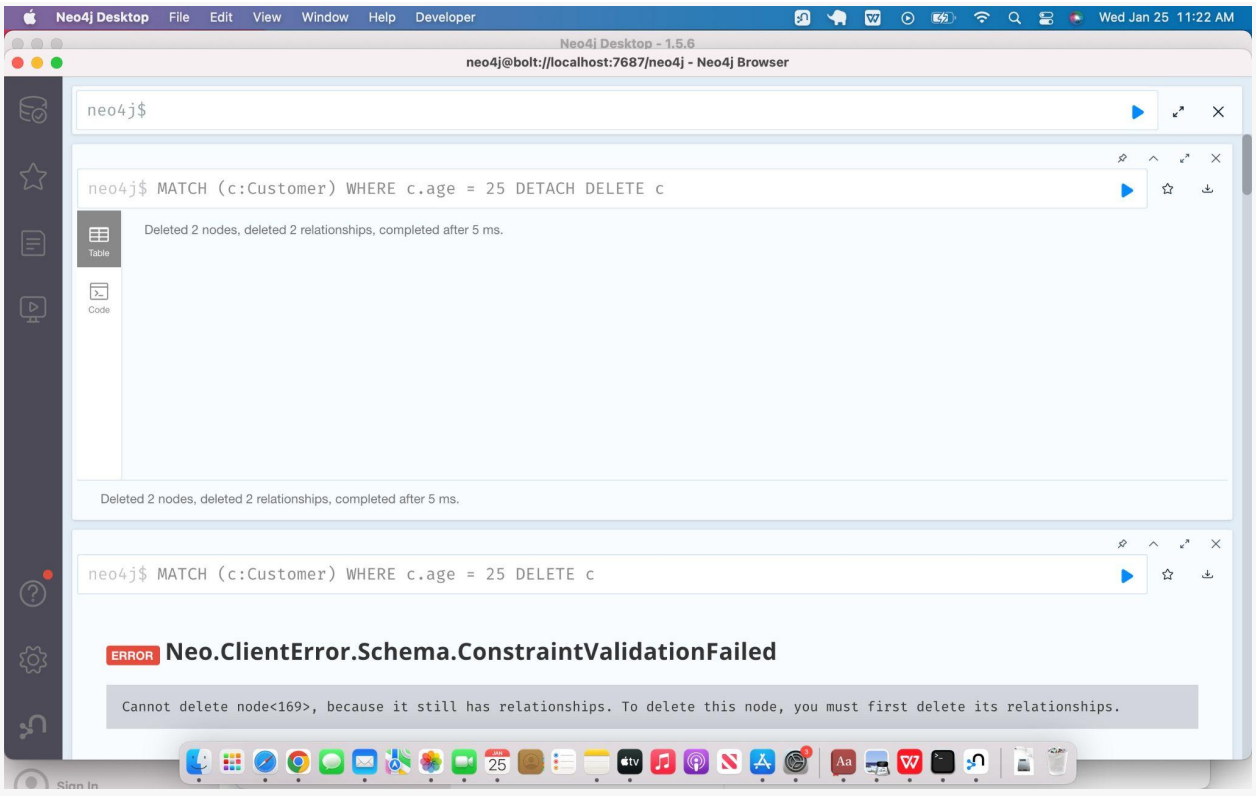
5. Find all the laptops whose shippability is true.

To find all the laptops whose shippability is true, you can use the Cypher query:
MATCH (p:Product)
WHERE p.shippability = true
RETURN p.title



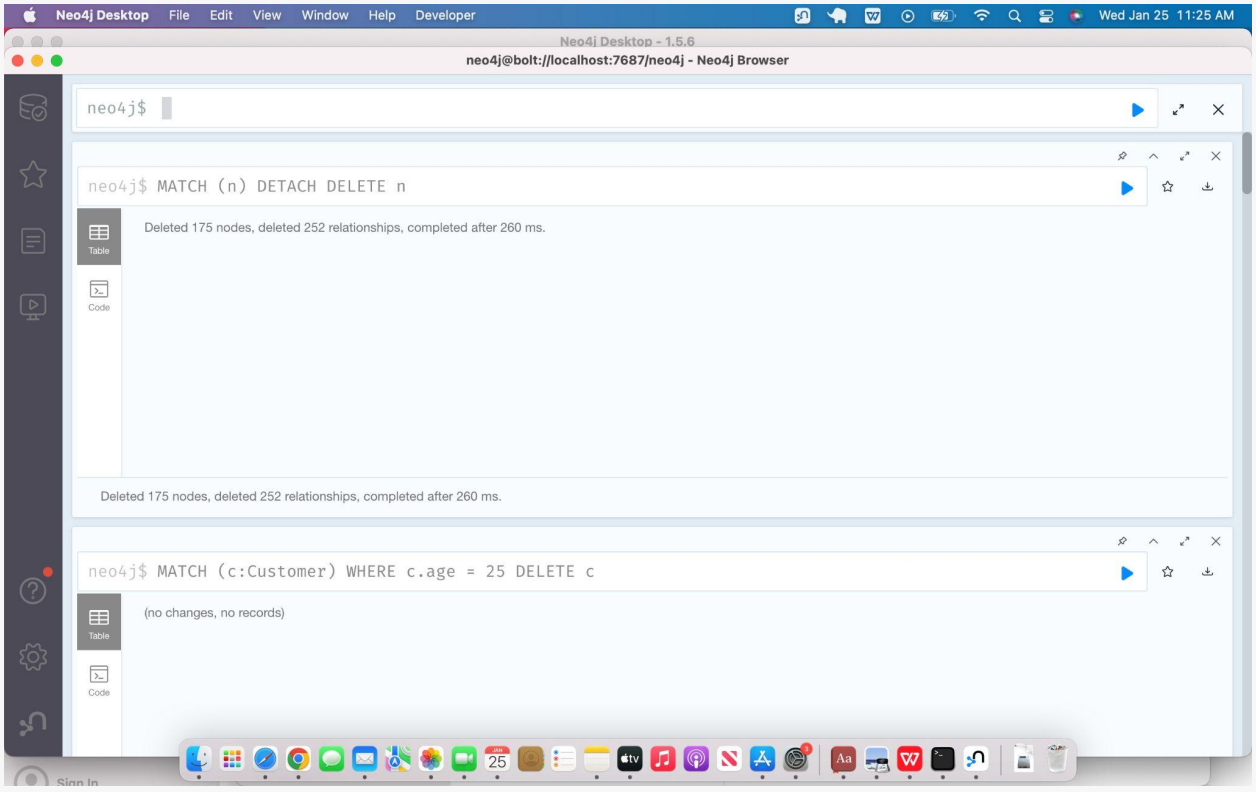


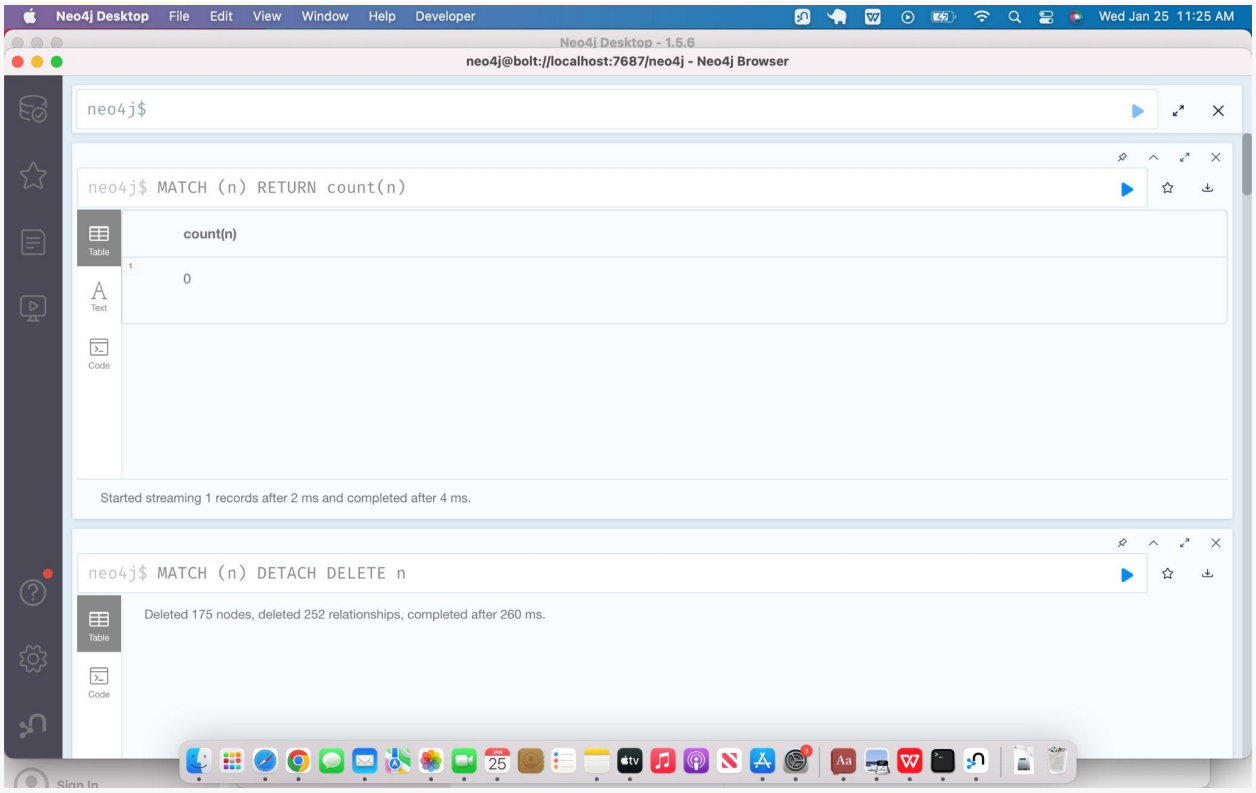
6. Delete all customers whose age is 25.MATCH (c:Customer)
WHERE c.age = 25
DETACH DELETE c

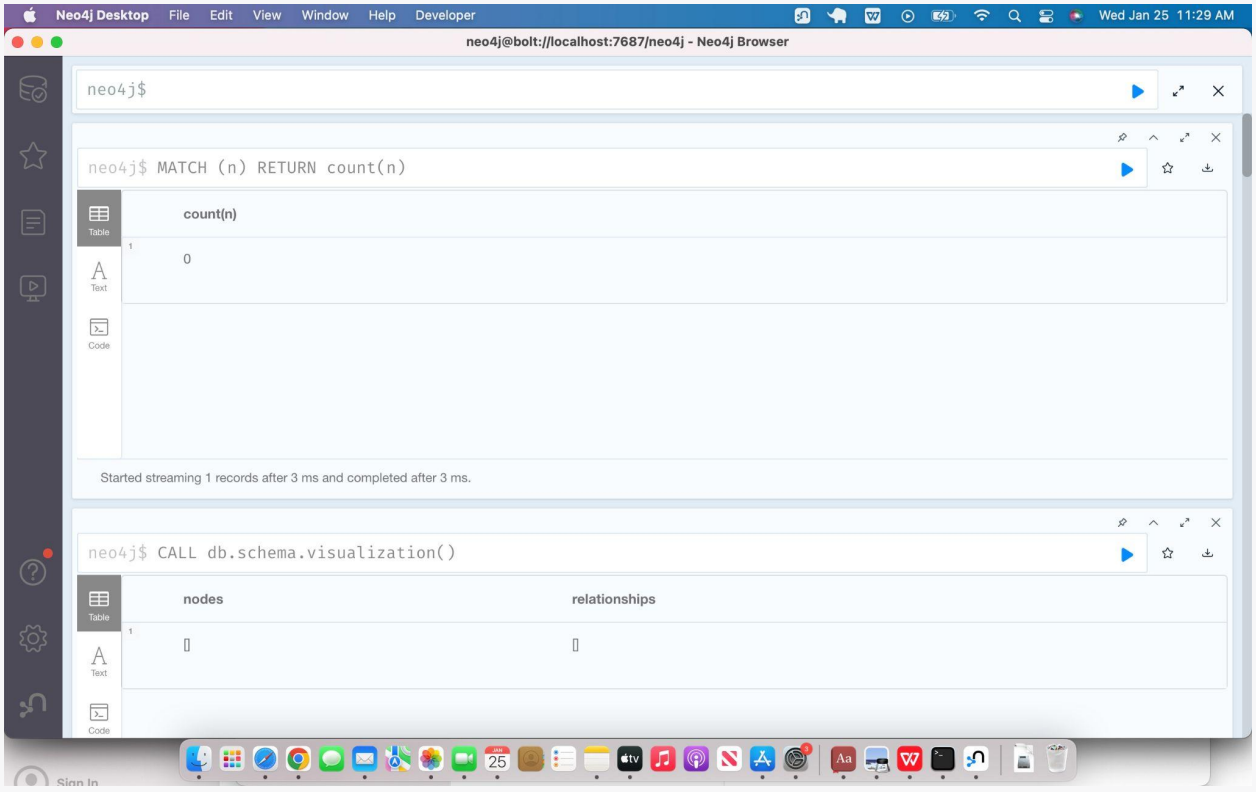


7. Delete all nodes along with their relationships. Also, prove that the nodes have been

deletedMATCH (n)
DETACH DELETE n


Neo4j Desktop File Edit View Window Help Developer
Wed Jan 25 11:25 AM
Neo4j Desktop - 1.5.6
neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
neo4j$
neo4j$ MATCH (n) DETACH DELETE n
Table
Code
Deleted 175 nodes, deleted 252 relationships, completed after 260 ms.
Deleted 175 nodes, deleted 252 relationships, completed after 260 ms.
neo4j$ MATCH (c:Customer) WHERE c.age = 25 DELETE c
Table
Code
(no changes, no records)
Sign In


Neo4j Desktop File Edit View Window Help Developer
Wed Jan 25 11:25 AM
Neo4j Desktop - 1.5.6
neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
neo4j$
neo4j$ MATCH (n) RETURN count(n)
Table
Text
Code
count(n)
1
0
Started streaming 1 records after 2 ms and completed after 4 ms.
neo4j$ MATCH (n) DETACH DELETE n
Table
Code
Deleted 175 nodes, deleted 252 relationships, completed after 260 ms.
Sign In

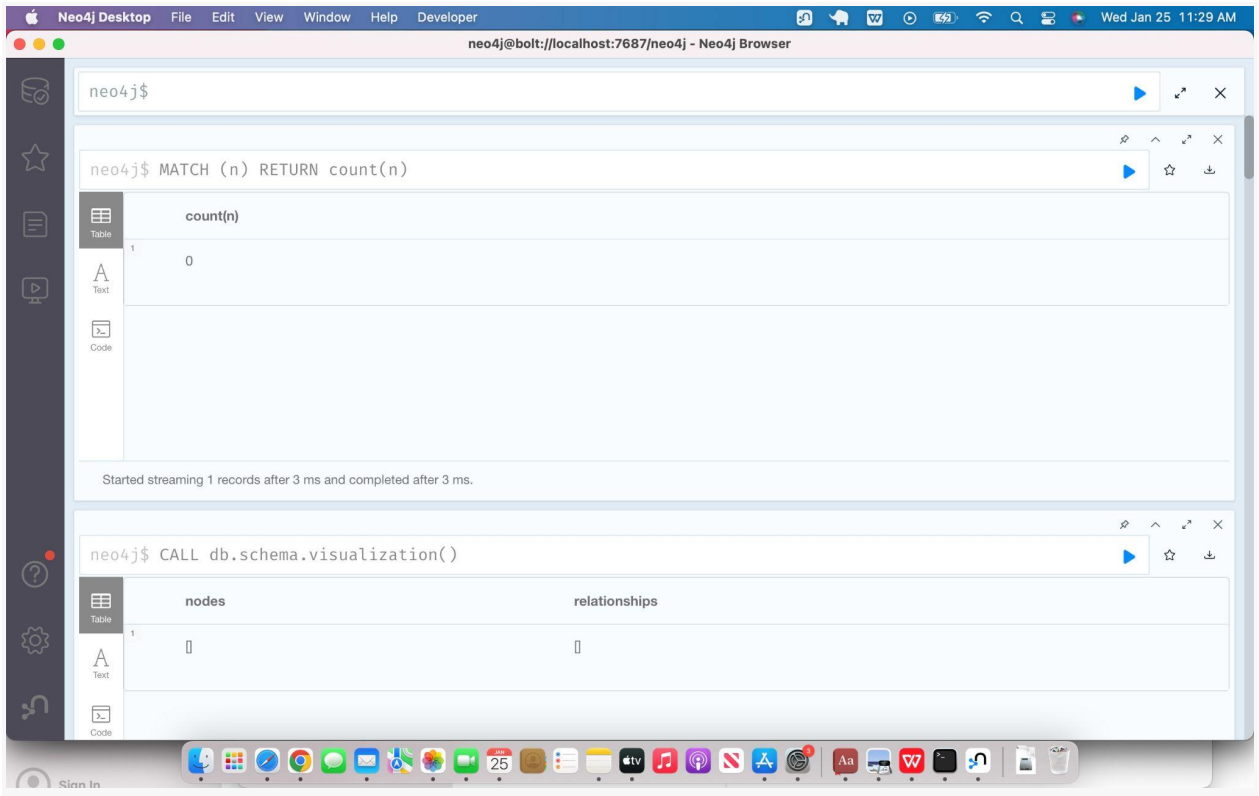


8. Visualise the inbuilt movie database of Neo4J and count the number of nodes in the movie database.

```
CALL db.schema.visualization()
```

```
MATCH (n)
RETURN count(n)
```

```
CALL db.labels()
```


Neo4j Desktop File Edit View Window Help Developer
Wed Jan 25 11:29 AM
neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
neo4j$
neo4j$ MATCH (n) RETURN count(n)
Table
Text
Code
count(n)
1
0
Started streaming 1 records after 3 ms and completed after 3 ms.
neo4j$ CALL db.schema.visualization()
Table
Text
Code
nodes
relationships
1
[]
[]