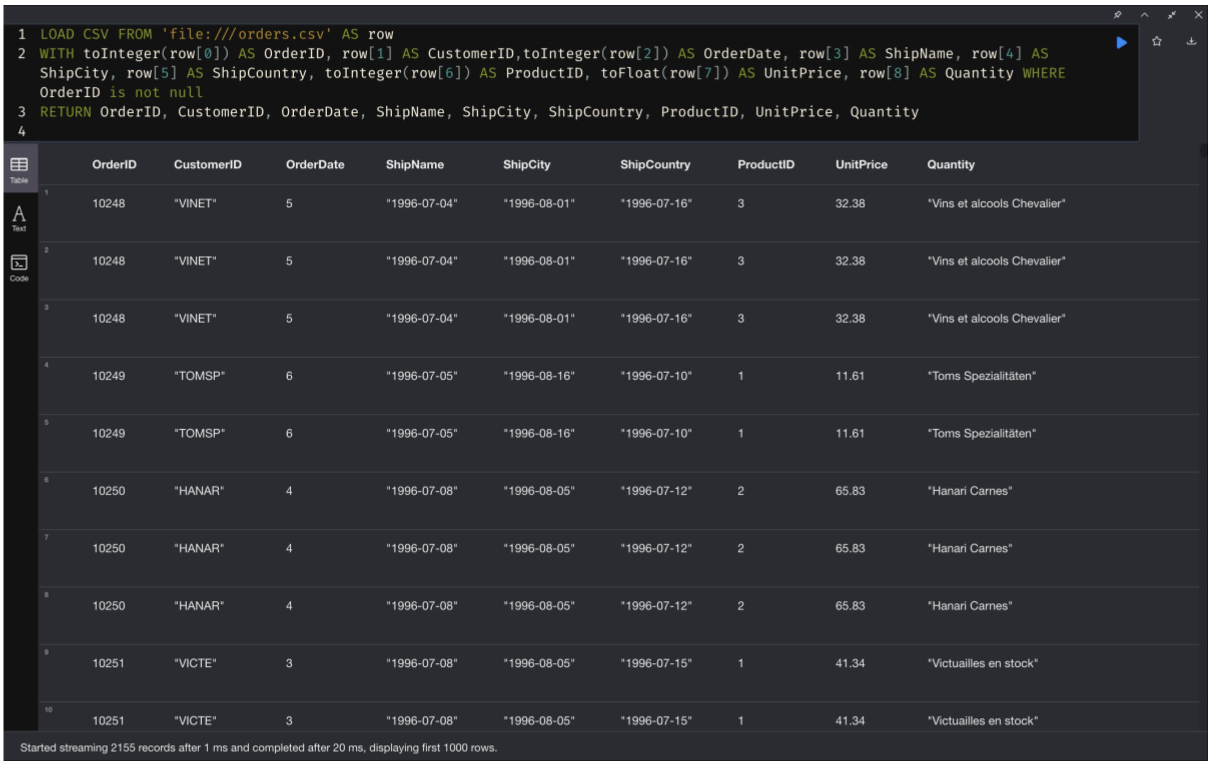
Data Science Infrastructures – Exercise 05

Assignment I: View CSV Data in Neo4J

After the using the Cypher Command for loading the CSV file for the specific columns, I also implemented the code snippet WHERE to prevent the errors. Also, the column with integer and float values were converted accordingly to their respective data type. The screen shot of the final output along with code is shown below.



```
LOAD CSV FROM 'file:///orders.csv' AS row
WITH toInteger(row[0]) AS OrderID, row[1] AS CustomerID,toInteger(row[2]) AS OrderDate, row[3] AS ShipName, row[4] AS
ShipCity, row[5] AS ShipCountry, toInteger(row[6]) AS ProductID, toFloat(row[7]) AS UnitPrice, row[8] AS Quantity WHERE
OrderID is not null
RETURN OrderID, CustomerID, OrderDate, ShipName, ShipCity, ShipCountry, ProductID, UnitPrice, Quantity
```

	OrderID	CustomerID	OrderDate	ShipName	ShipCity	ShipCountry	ProductID	UnitPrice	Quantity
1	10248	"VINET"	5	"1996-07-04"	"1996-08-01"	"1996-07-16"	3	32.38	"Vins et alcools Chevalier"
2	10248	"VINET"	5	"1996-07-04"	"1996-08-01"	"1996-07-16"	3	32.38	"Vins et alcools Chevalier"
3	10248	"VINET"	5	"1996-07-04"	"1996-08-01"	"1996-07-16"	3	32.38	"Vins et alcools Chevalier"
4	10249	"TOMSP"	6	"1996-07-05"	"1996-08-16"	"1996-07-10"	1	11.61	"Toms Spezialitäten"
5	10249	"TOMSP"	6	"1996-07-05"	"1996-08-16"	"1996-07-10"	1	11.61	"Toms Spezialitäten"
6	10250	"HANAR"	4	"1996-07-08"	"1996-08-05"	"1996-07-12"	2	65.83	"Hanari Carnes"
7	10250	"HANAR"	4	"1996-07-08"	"1996-08-05"	"1996-07-12"	2	65.83	"Hanari Carnes"
8	10250	"HANAR"	4	"1996-07-08"	"1996-08-05"	"1996-07-12"	2	65.83	"Hanari Carnes"
9	10251	"VICTE"	3	"1996-07-08"	"1996-08-05"	"1996-07-15"	1	41.34	"Victuailles en stock"
10	10251	"VICTE"	3	"1996-07-08"	"1996-08-05"	"1996-07-15"	1	41.34	"Victuailles en stock"

```
Started streaming 2155 records after 1 ms and completed after 20 ms, displaying first 1000 rows.
```

Figure 1Loading the specific columns of 'orders.csv' in Neo4J

Assignment II – Create the Graph
The schema that represents customers, products, and orders as well as their relationships are shown below. Also, their necessary properties are adde to the nodes and relationships.

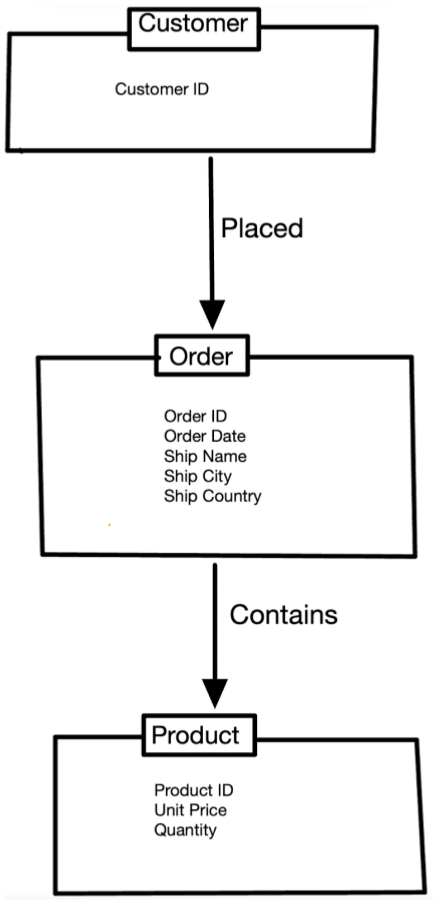
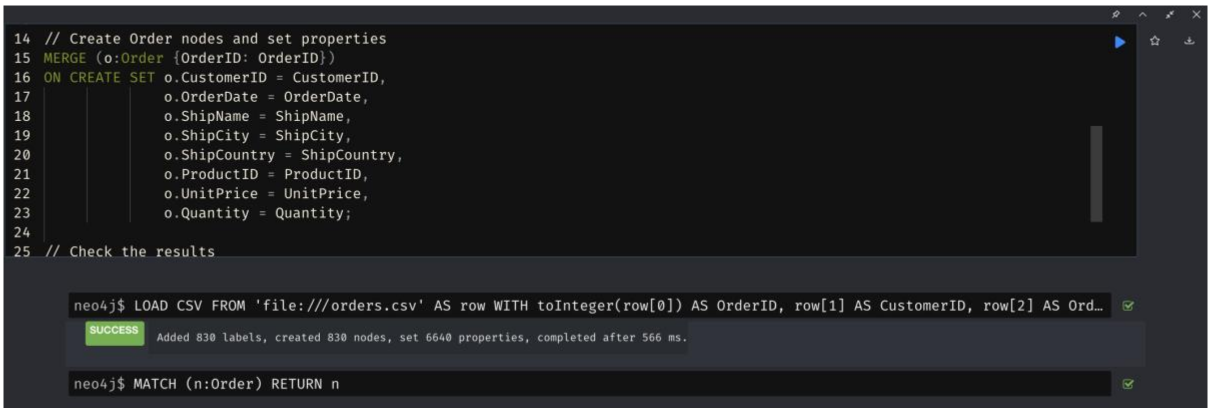


Figure 2 Schema that represents customer, products, and orders along with their relationships

Assignment II: B

a. Order nodes and properties according to the model using the MERGE command are created and shown below. Also, the result is checked by using the MATCH clause.

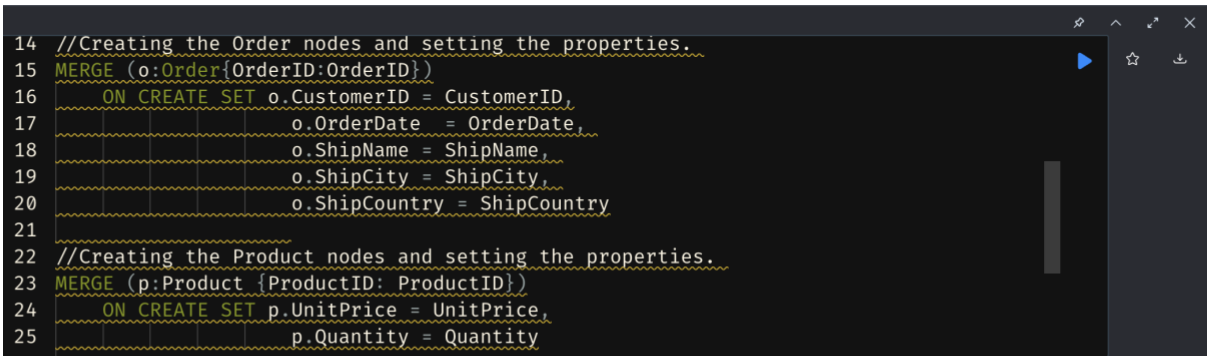


```
// Create Order nodes and set properties
MERGE (o:Order {OrderID: OrderID})
ON CREATE SET o.CustomerID = CustomerID,
              o.OrderDate = OrderDate,
              o.ShipName = ShipName,
              o.ShipCity = ShipCity,
              o.ShipCountry = ShipCountry,
              o.ProductID = ProductID,
              o.UnitPrice = UnitPrice,
              o.Quantity = Quantity;

// Check the results
```

```
neo4j$ LOAD CSV FROM 'file:///orders.csv' AS row WITH toInteger(row[0]) AS OrderID, row[1] AS CustomerID, row[2] AS Ord…
SUCCESS  Added 830 labels, created 830 nodes, set 6640 properties, completed after 566 ms.

neo4j$ MATCH (n:Order) RETURN n
```

Figure 3 Creating Order nodes and setting their properties

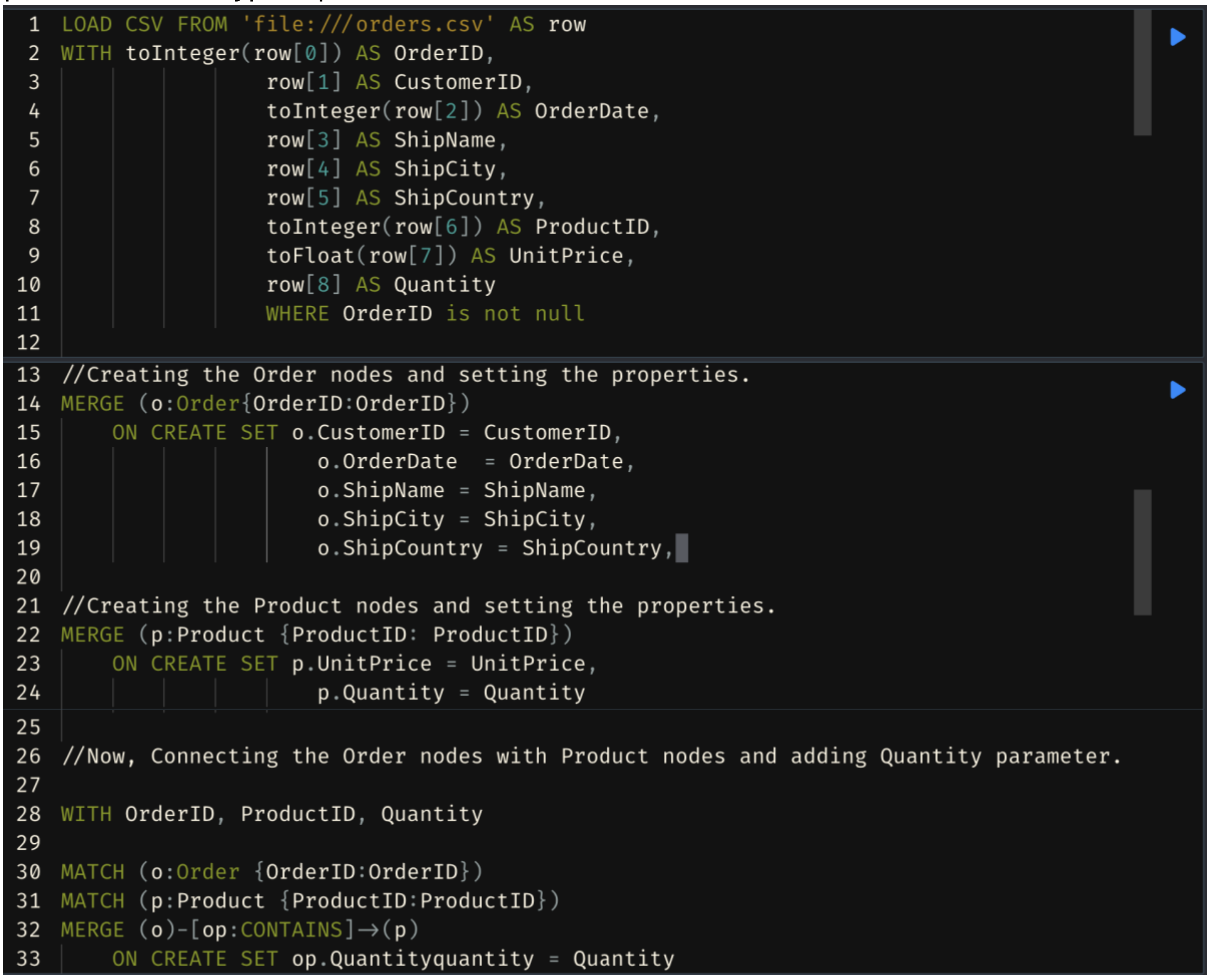
b. Now, the results after extending the Cypher to create the Product Nodes is shown below.



```
//Creating the Order nodes and setting the properties.
MERGE (o:Order{OrderID:OrderID})
    ON CREATE SET o.CustomerID = CustomerID,
                  o.OrderDate  = OrderDate,
                  o.ShipName = ShipName,
                  o.ShipCity = ShipCity,
                  o.ShipCountry = ShipCountry

//Creating the Product nodes and setting the properties.
MERGE (p:Product {ProductID: ProductID})
    ON CREATE SET p.UnitPrice = UnitPrice,
                  p.Quantity = Quantity
```

Figure 4 Cypher extended to create Product nodes.

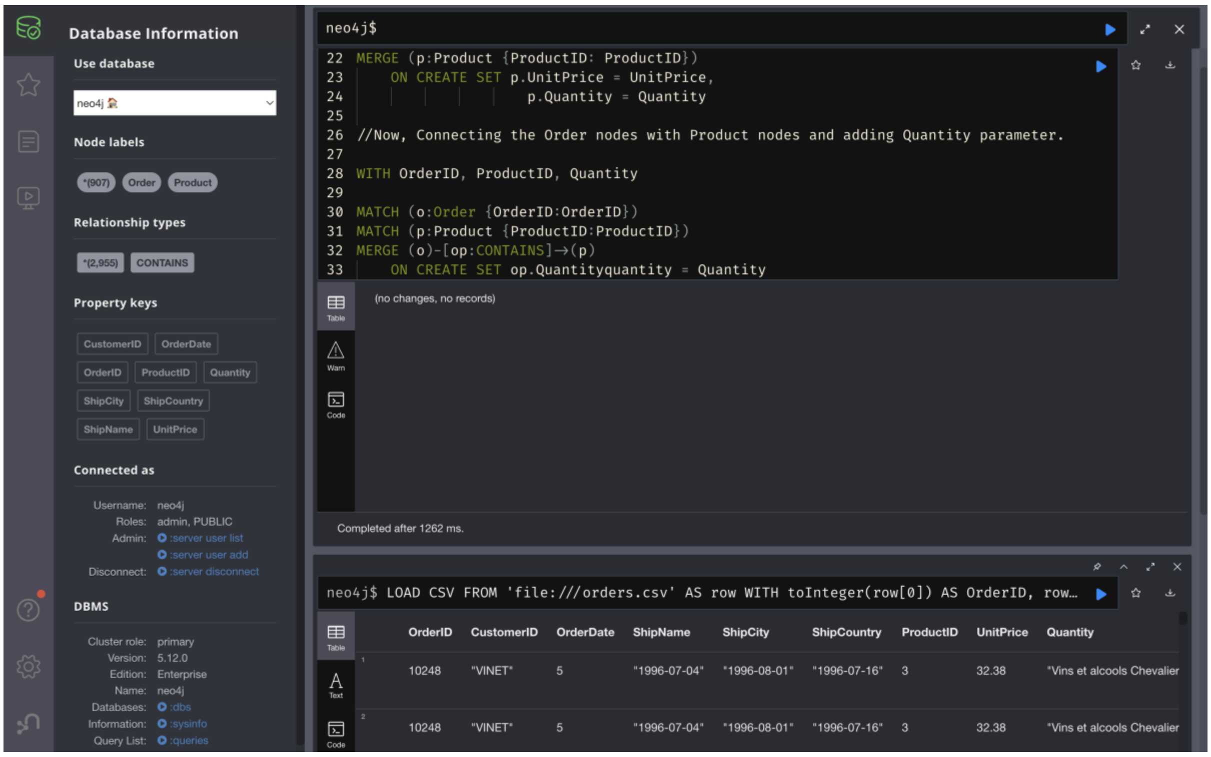
c. After connecting Order nodes with the Product nodes and adding the Quantity parameter, the Cypher parts look like this.



```
LOAD CSV FROM 'file:///orders.csv' AS row
WITH toInteger(row[0]) AS OrderID,
                row[1] AS CustomerID,
                toInteger(row[2]) AS OrderDate,
                row[3] AS ShipName,
                row[4] AS ShipCity,
                row[5] AS ShipCountry,
                toInteger(row[6]) AS ProductID,
                toFloat(row[7]) AS UnitPrice,
                row[8] AS Quantity
                WHERE OrderID is not null

//Creating the Order nodes and setting the properties.
MERGE (o:Order{OrderID:OrderID})
    ON CREATE SET o.CustomerID = CustomerID,
                  o.OrderDate  = OrderDate,
                  o.ShipName = ShipName,
                  o.ShipCity = ShipCity,
                  o.ShipCountry = ShipCountry,

//Creating the Product nodes and setting the properties.
MERGE (p:Product {ProductID: ProductID})
    ON CREATE SET p.UnitPrice = UnitPrice,
                  p.Quantity = Quantity

//Now, Connecting the Order nodes with Product nodes and adding Quantity parameter.

WITH OrderID, ProductID, Quantity

MATCH (o:Order {OrderID:OrderID})
MATCH (p:Product {ProductID:ProductID})
MERGE (o)-[op:CONTAINS]→(p)
    ON CREATE SET op.Quantityquantity = Quantity
```

Figure 5 Final Cyper part after connecting Order nodes with Product nodes

The result containing the order, and their related products is shown here. We can see that there are 907 node labels (Order and Product) with 2,955 relationship types (CONTAINS). The Property keys also reflect all the properties that were considered.



Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
Disconnect: :server disconnect

Cluster role: primary
Version: 5.12.0
Edition: Enterprise
Name: neo4j
Databases: :dbs
Information: :sysinfo
Query List: :queries

```
neo4j$
MERGE (p:Product {ProductID: ProductID})
    ON CREATE SET p.UnitPrice = UnitPrice,
                  p.Quantity = Quantity

//Now, Connecting the Order nodes with Product nodes and adding Quantity parameter.

WITH OrderID, ProductID, Quantity

MATCH (o:Order {OrderID:OrderID})
MATCH (p:Product {ProductID:ProductID})
MERGE (o)-[op:CONTAINS]→(p)
    ON CREATE SET op.Quantityquantity = Quantity
```

(no changes, no records)

Completed after 1262 ms.

```
neo4j$ LOAD CSV FROM 'file:///orders.csv' AS row WITH toInteger(row[0]) AS OrderID, row…
```

	OrderID	CustomerID	OrderDate	ShipName	ShipCity	ShipCountry	ProductID	UnitPrice	Quantity
1	10248	"VINET"	5	"1996-07-04"	"1996-08-01"	"1996-07-16"	3	32.38	"Vins et alcools Chevalier
2	10248	"VINET"	5	"1996-07-04"	"1996-08-01"	"1996-07-16"	3	32.38	"Vins et alcools Chevalier

Figure 6 Final outlook of Cypher part