**IMPLIMENTATION OF GRAPH DATABASE IN NEO4J**

**NEO4J:-**Network Exploration and Optimization 4 Java.

It uses the Cypher query language (CPL).

CPL is an Open and multi-vendor query language.

**GRAPH DATABASE :-**  It is a database designed to treat the relationship between data in dataset.

* It is composed of a nodes and relationships.
* It is intended to hold the data without preparing the model. Instead of we need to draw the data from dataset and show the how an each individual connected to each other and how they are related to each other.

We have taken the **SalesDB,** this database contains 4 tables:

* Customers
* Employees
* Products
* Sales

**5.GRAPH DATABASES**

**5.1 LOADING THE TABLE TO NEO4J:-**

LOAD CSV WITH HEADERS FROM 'file:///Customers.csv' as row CREATE (c:Customer) SET c = {customerId:row.CustomerID, name:row.FirstName, middle:row.MiddleInitial, lastname:row.LastName } return c



**5.2 CREATING THE RELATIONSHIP BETWEEN NODES**:

MATCH (e:Employees),(s:Sales) WHERE e.employeeId =s.salespersonId CREATE (e) - [r:Quantity Product Sold] -> (s) RETURN e,s,r

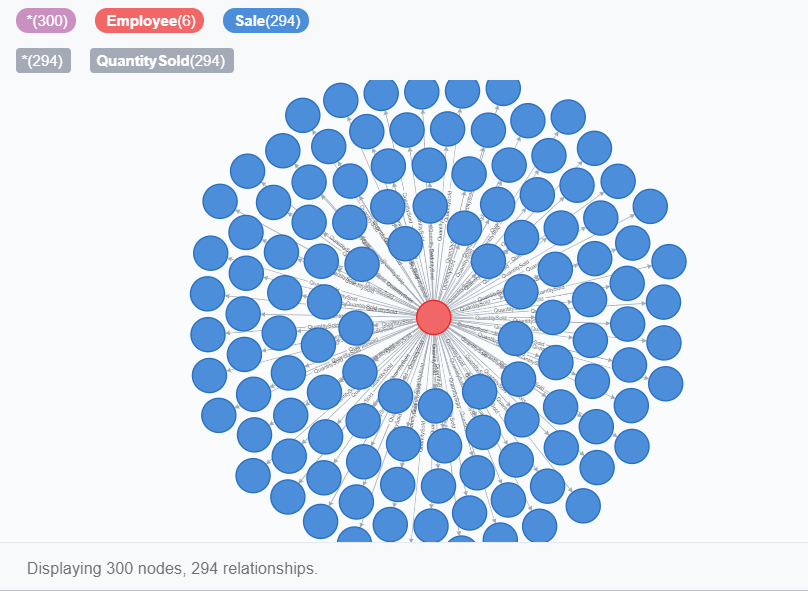


FIG 5.2.1 Relationship between Employees and Sales

MATCH (c:Customers),(s:Sales) WHERE c.customerID =s.customeriD CREATE (c) - [r:Purchased] -> (s) RETURN c,s,r

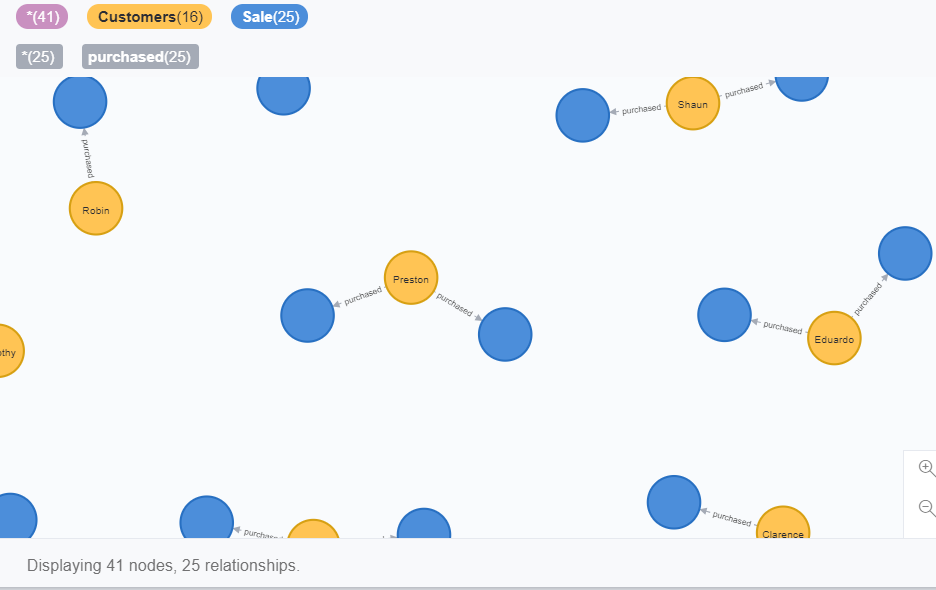


FIG 5.2.2 Relationship between Customers and Sales

MATCH (p:Products),(s:Sales) WHERE p.productId =s.productiD CREATE (p) - [r:ProductSold] -> (s) RETURN p,s,r

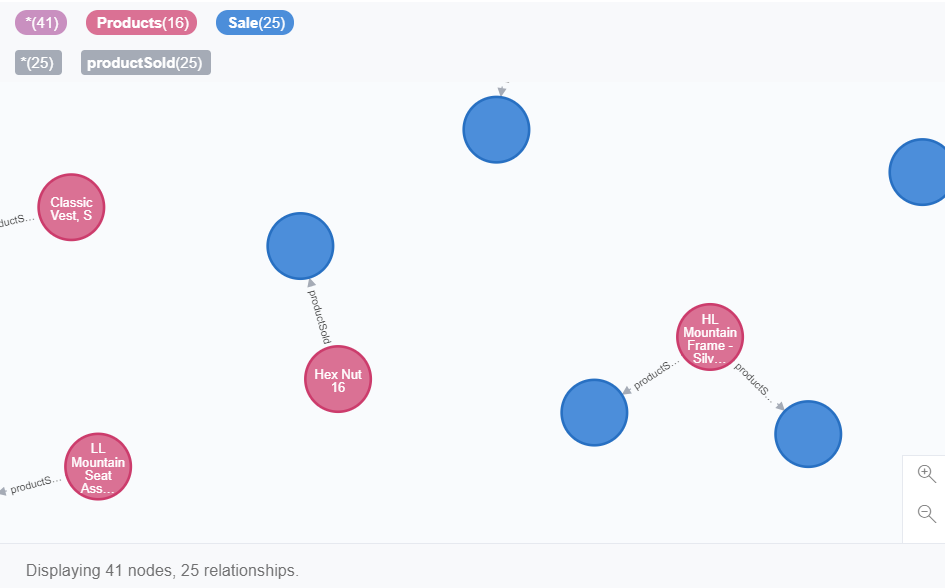
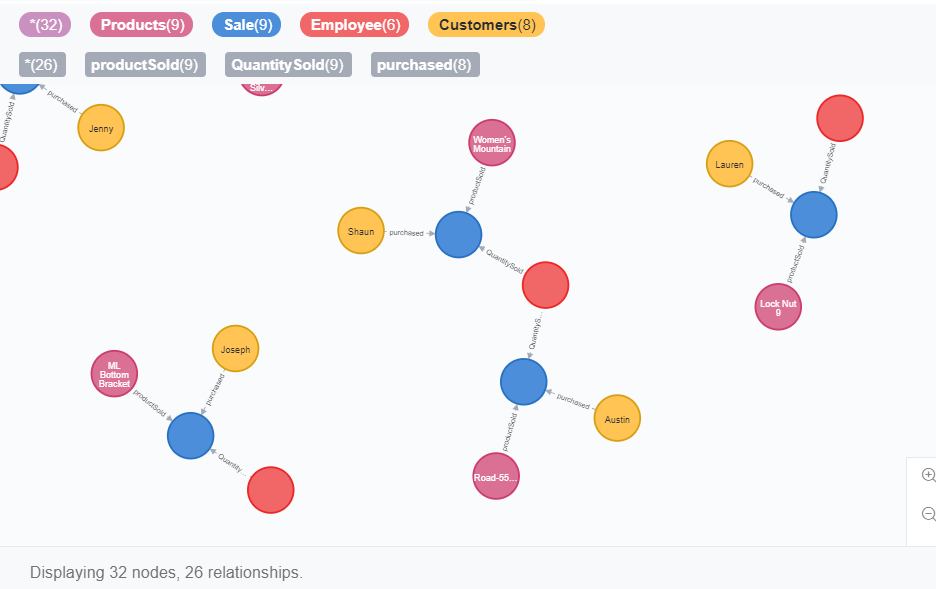


FIG 5.2.3 Relationship between Employees and Sales

FIG 5.2.4 Graph of the Database

**5.3COMPARING THE SQL AND CQL:**

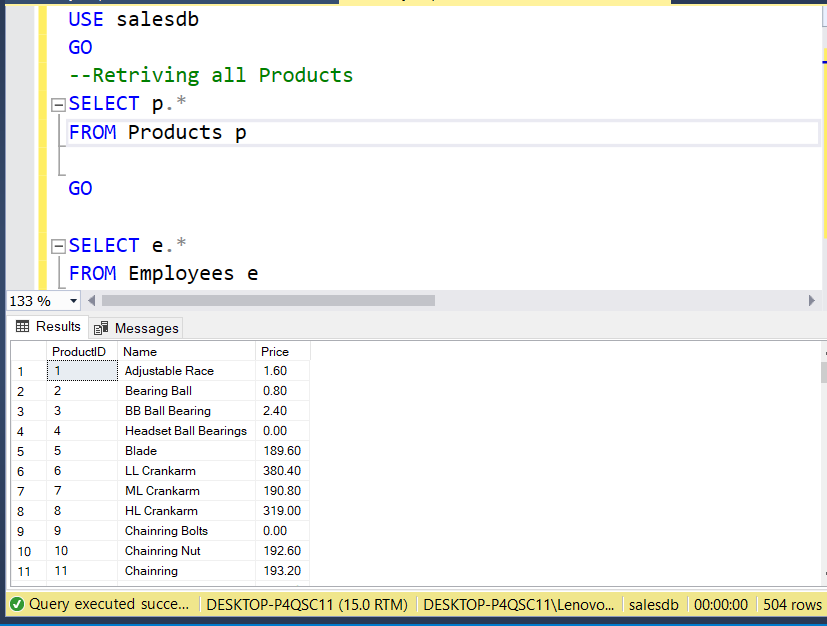
Given below are comparison between SQL queries and CQL queries along with the respective output screenshots:-

Retrieving the DATA

**SQL1.**

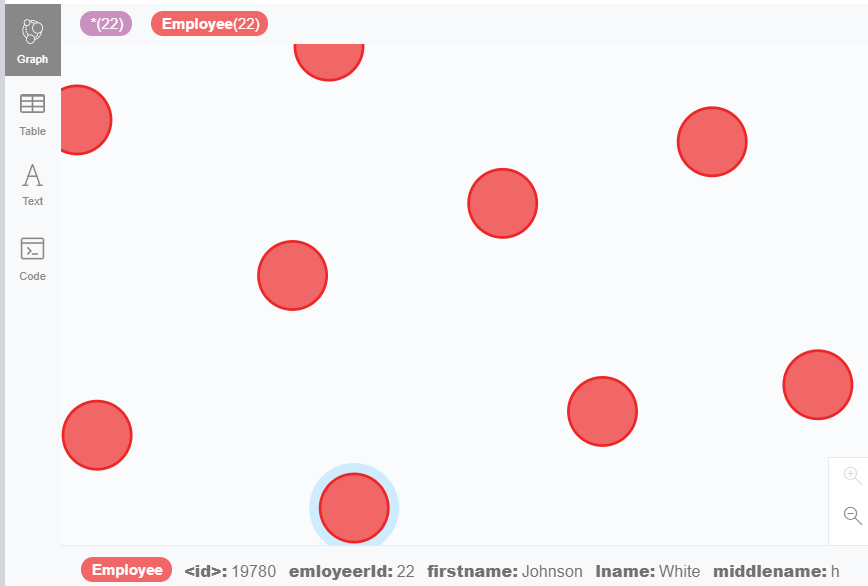
**SELECT e.\***

**FROM Employees e**

****

**CQL1.**

**MATCH (n:Employee) RETURN n**

****

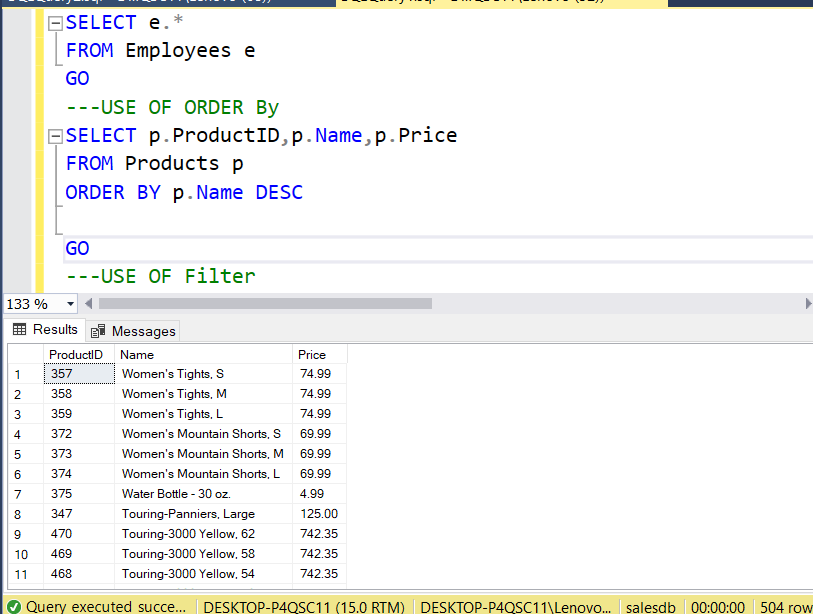
**SQL2.**

**---USE OF ORDER By**

**SELECT p.ProductID, p.Name, p.Price**

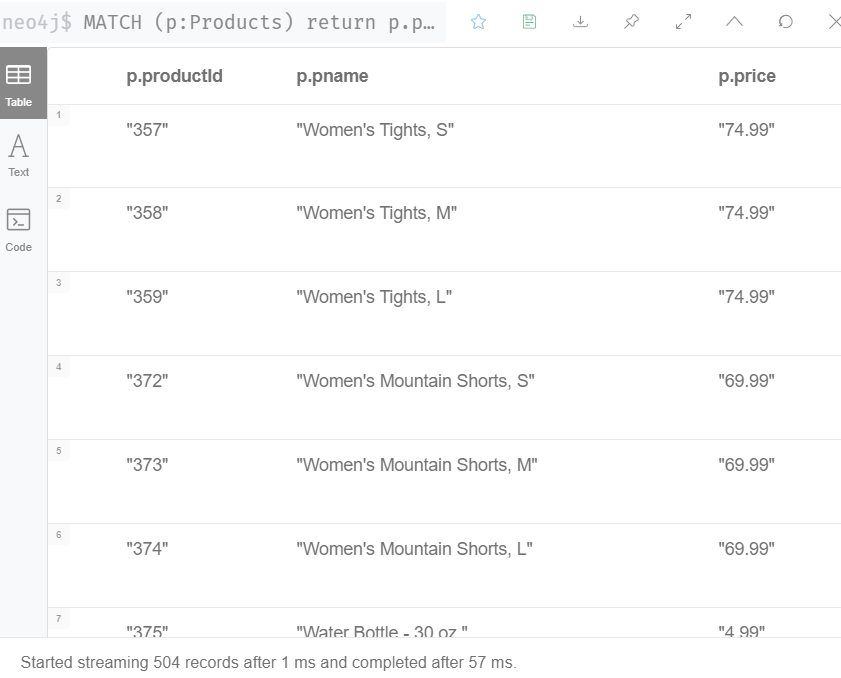
**FROM Products p**

**ORDER BY p.Name DESC**

****

**CQL 2.**

**MATCH (p:Products) return p.productId, p.pname, p.price ORDER BY p.pname DESC**

****

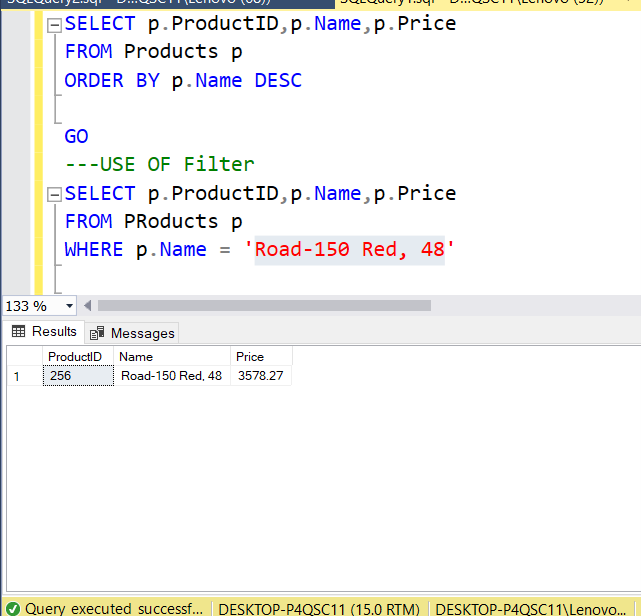
**SQL 3.**

**---USE OF Filter**

**SELECT p.ProductID,p.Name,p.Price**

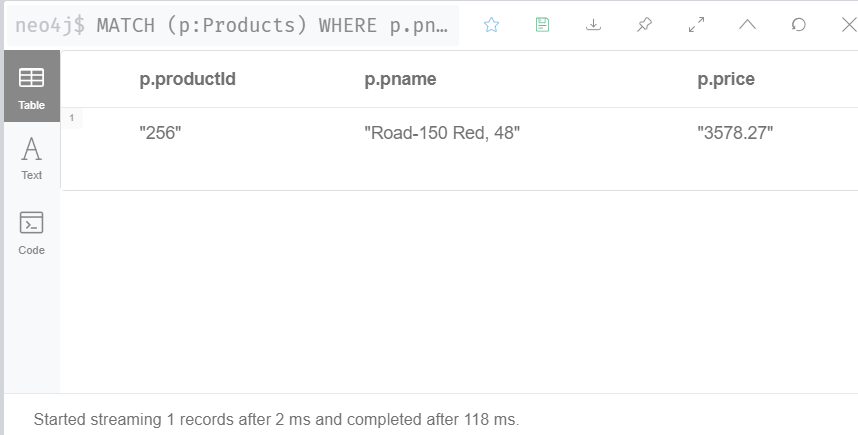
**FROM PRoducts p**

**WHERE p.Name = 'Road-150 Red, 48'**

****

**CQL 3.**

**MATCH (p:Products) WHERE p.pname ='Road-150 Red, 48'return p.productId, p.pname,p.price**



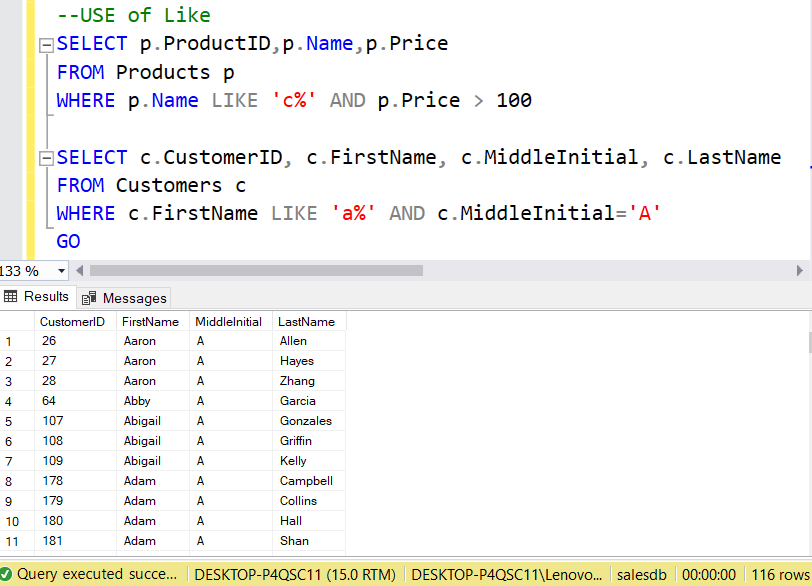
**SQL 4.**

**--USE of Like**

**SELECT c.CustomerID, c.FirstName, c.MiddleInitial, c.LastName**

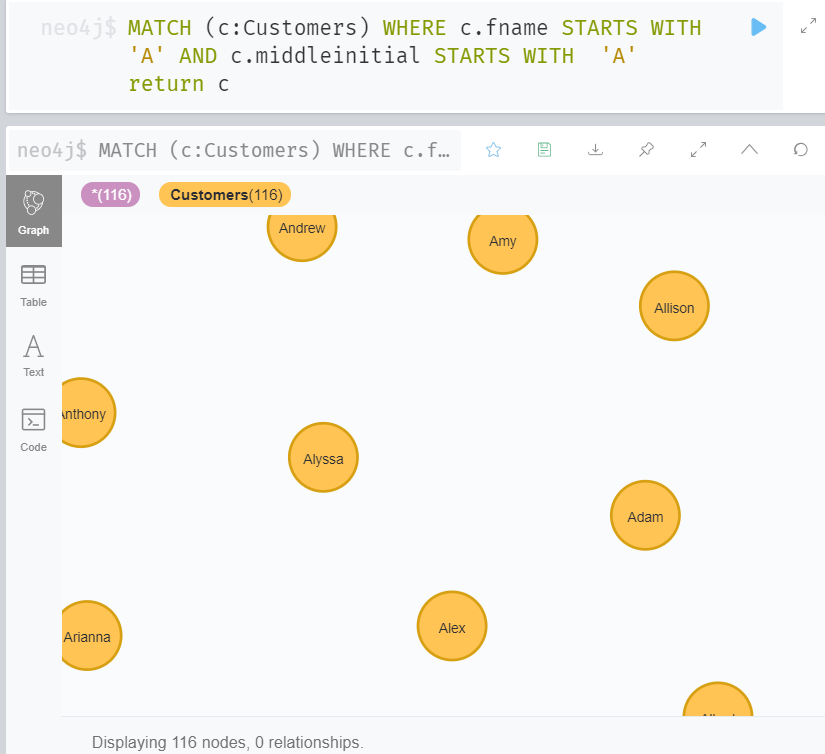
**FROM Customers c**

**WHERE c.FirstName LIKE 'a%' AND c.MiddleInitial='A'**

****

**CQL 4.**

**MATCH (c:Customers) WHERE c.fname STARTS WITH 'A' AND c.middleinitial STARTS WITH 'A' return c**

****

**SQL 5.**

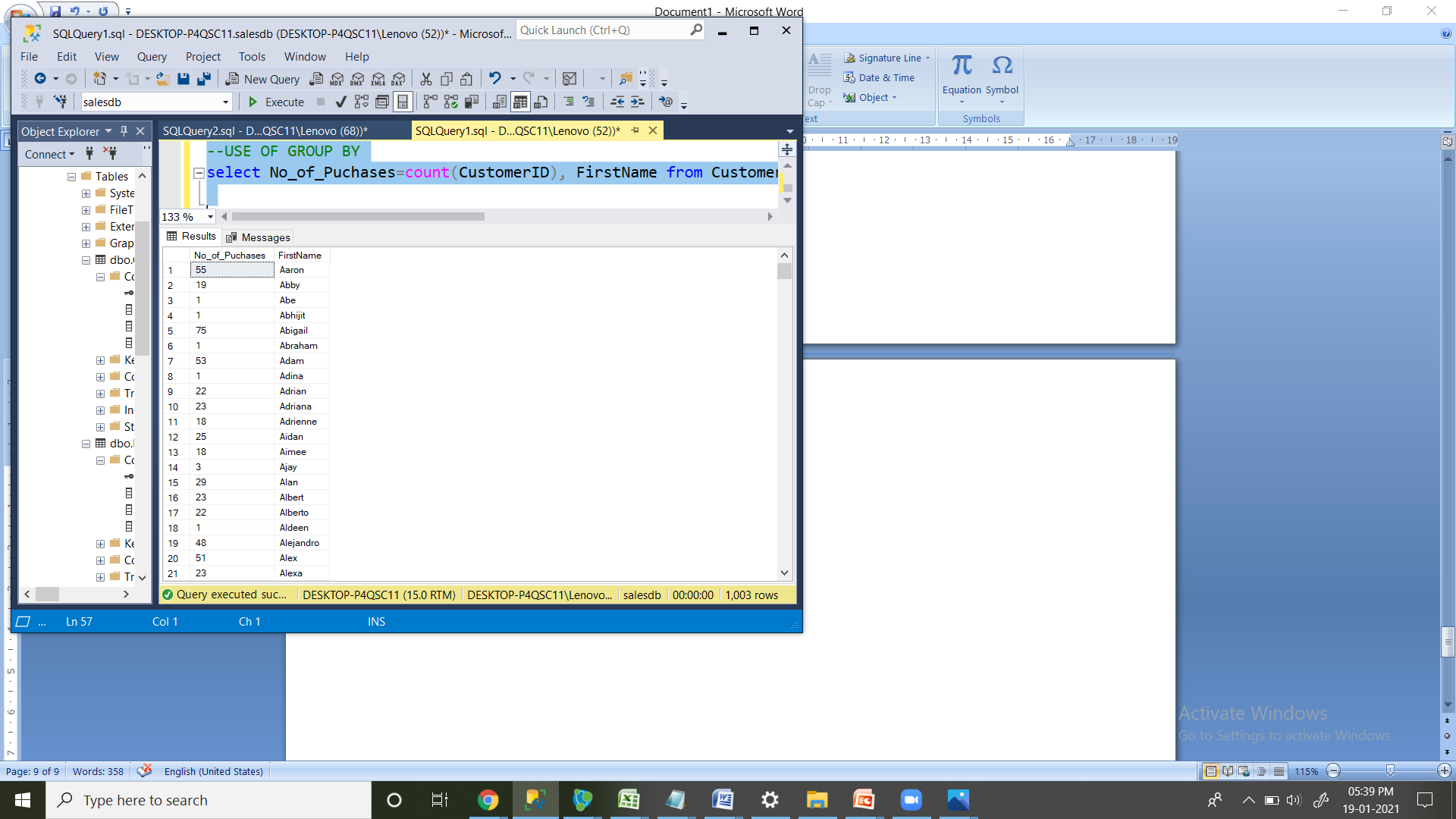
**--USE OF GROUP BY**

**select No\_of\_Puchases=count(CustomerID), FirstName**

**from Customers**

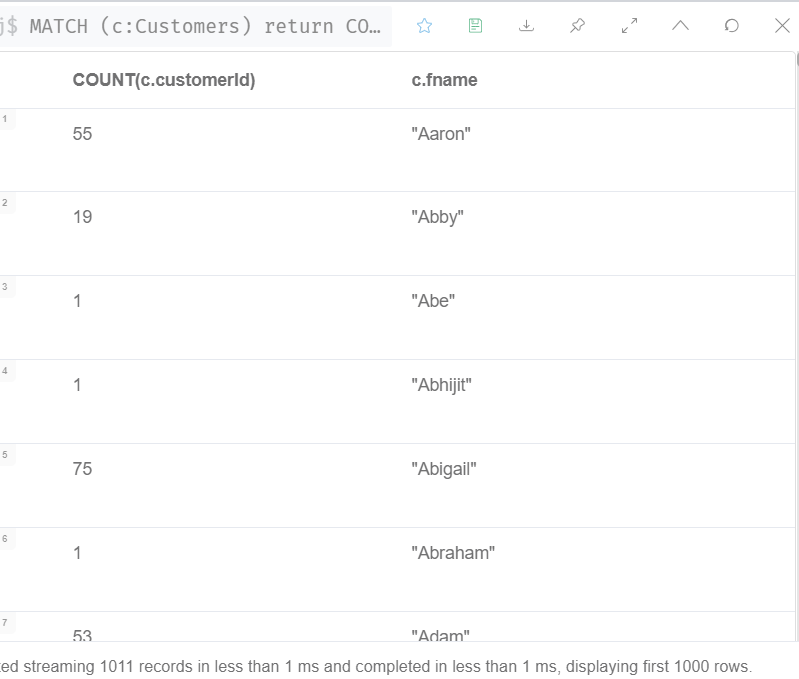
**group by FirstName**

**order by FirstName**

****

**CQL 5.**

**MATCH (c:Customers) return COUNT(c.customerId), c.fname ORDER BY c.fname**

****

**SQL 6.**

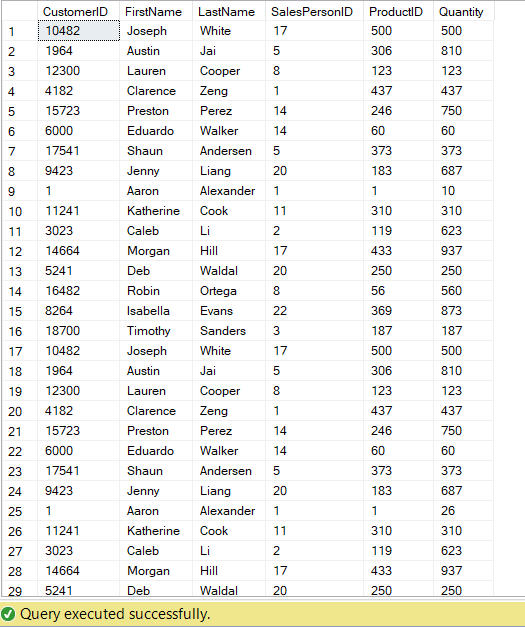
**--USE OF JOINS**

**SELECT TOP 1000 c.CustomerID,c.FirstName, c.LastName,s.SalesPersonID,s.ProductID,s.Quantity**

**FROM Customers c**

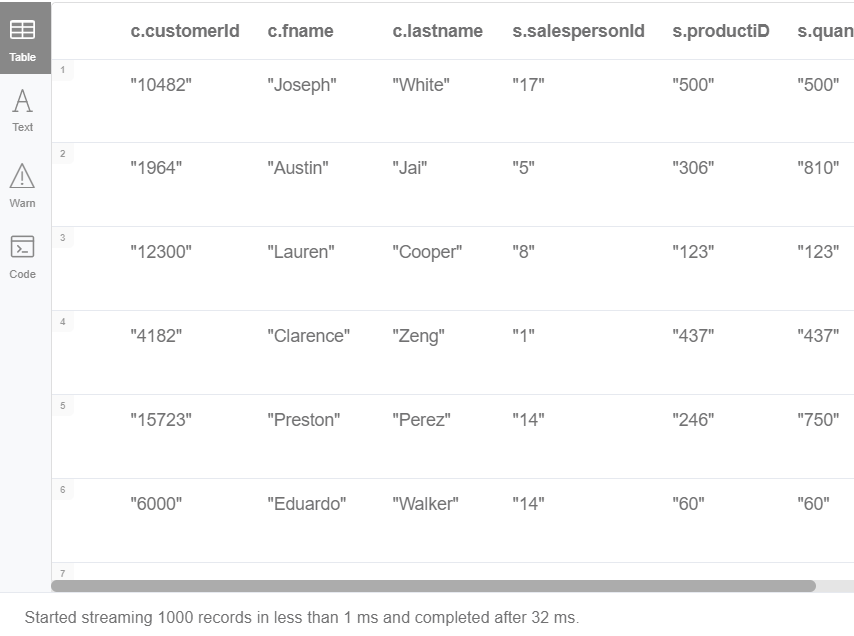
**INNER JOIN Sales s**

**ON c.CustomerID=s.CustomerID**

****

**CQL 6.**

**MATCH (c:Customers),(s:Sale) WHERE c.customerId=s.customeriD return c.customerId,c.fname, c.lastname,s.salespersonId,s.productiD,s.quantity**

****

**SQL 7.**

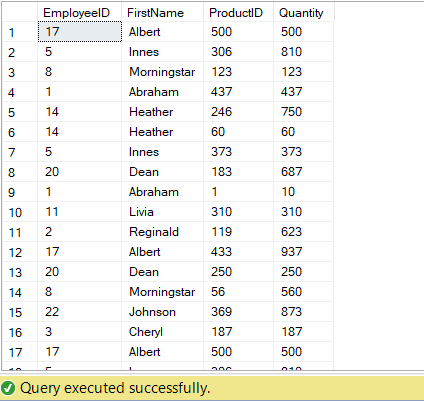
**--USE OF JOINS**

**SELECT TOP 1000 e.EmployeeID,e.FirstName,s.ProductID,s.Quantity**

**FROM Employees e**

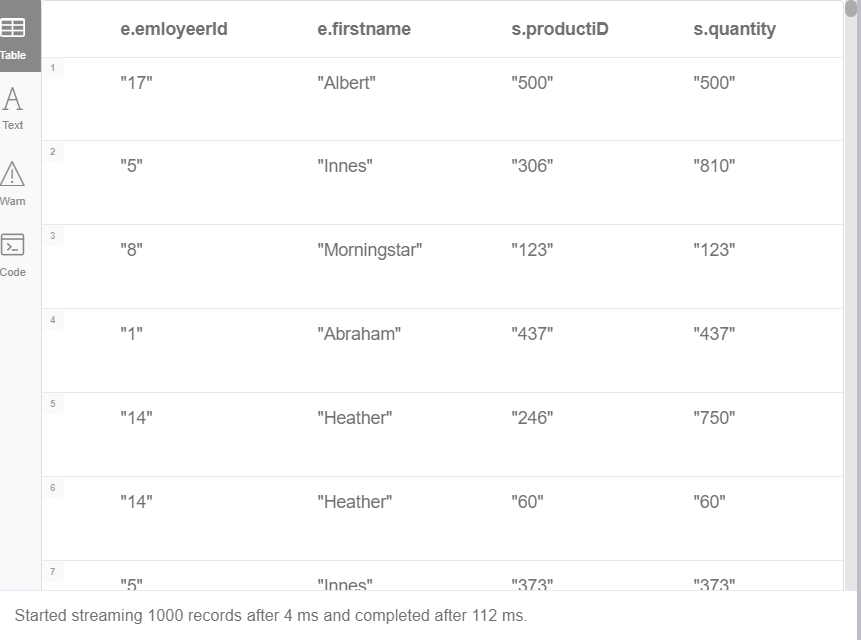
**INNER JOIN Sales s**

**ON e.EmployeeID=s.SalesPersonID**

****

**CQL 7.**

**MATCH (e:Employee),(s:Sale) WHERE e.emloyeerId=s.salespersonId return e.emloyeerId,e.firstname,s.productiD,s.quantity**

****