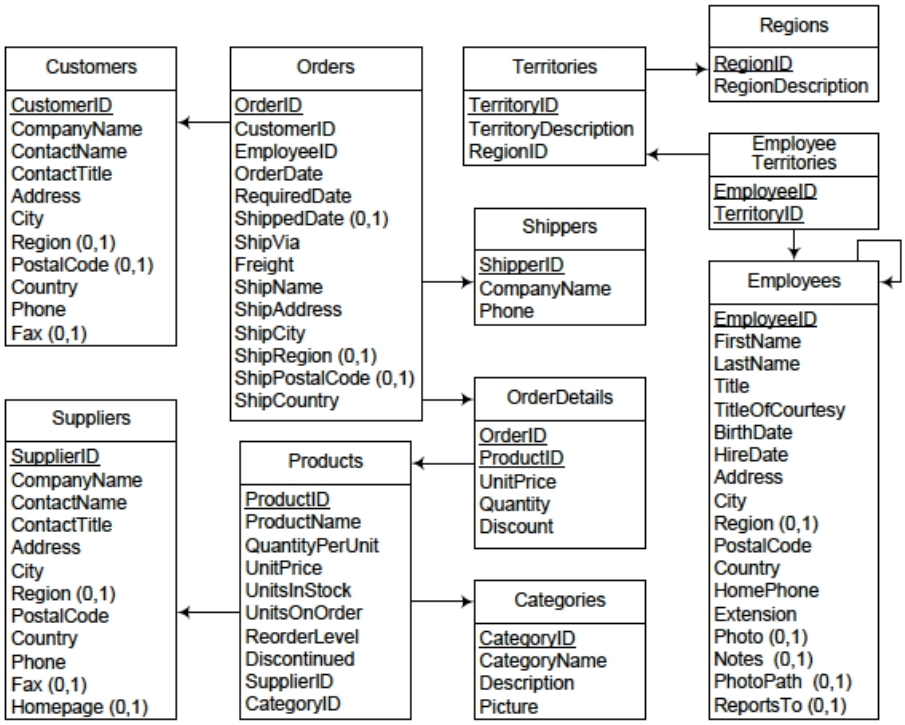
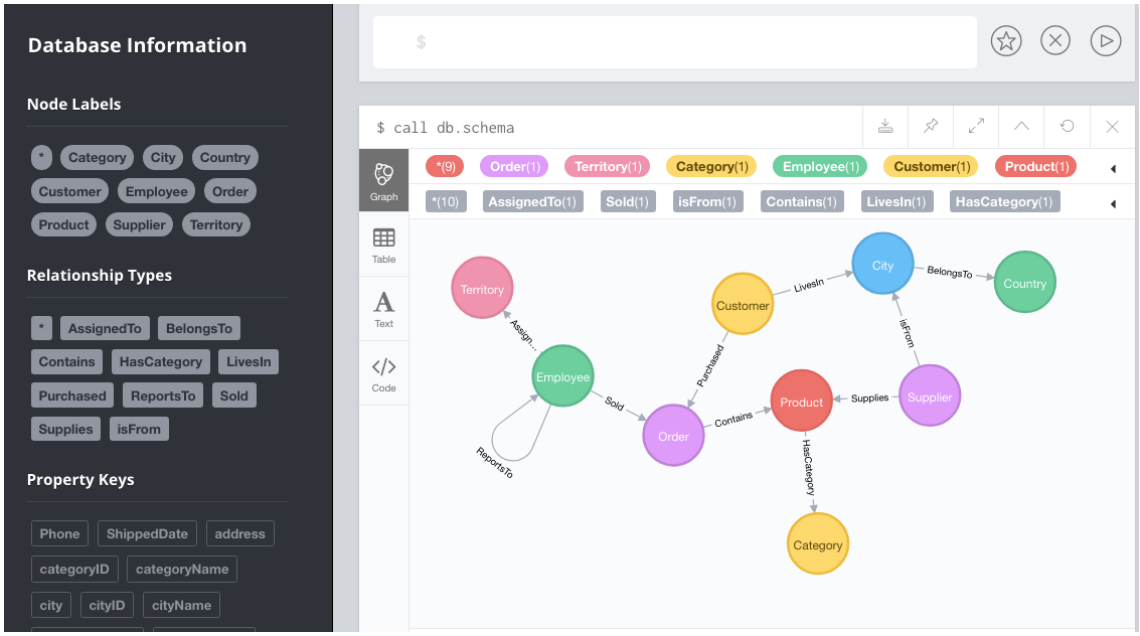
Graph Databases

Activity 3 – Cypher - Solutions

Exercise 1.





Write in Cypher the following queries over the northwindhg.db database:

Query 1 - List products and their unit price.

```
MATCH (p:Product)
RETURN p.productName, p.unitPrice
ORDER BY p.unitPrice DESC
```

Query 2 - List information about products 'Chocolade' & 'Pavlova'.

```
MATCH (p:Product)
WHERE p.productName IN ['Chocolade','Pavlova']
RETURN p
```

Query 3 - List information about products with names starting with a "C", whose unit price is greater than 50.

```
MATCH (p:Product)
WHERE p.productName STARTS WITH "C" AND tofloat(p.unitPrice) > 50
RETURN p.productName, p.unitPrice;
```

Query 4 - Same as 3, but considering the sales price, not the product's price.

```
MATCH (p:Product) <- [c:Contains] - (o:Order)
WHERE p.productName STARTS WITH "C" AND tofloat(c.unitPrice) > 50
RETURN distinct p.productName, p.unitPrice,c.unitPrice;
```

Query 5 - Total amount purchased by customer and product.

```
MATCH (c:Customer)
OPTIONAL MATCH (p:Product)<-[pu:Contains]-(:Order)-[:Purchased]->(c)
RETURN c.customerName,p.productName, tofloat(sum(tofloat(pu.unitPrice) *
toInteger (pu.quantity))) AS volume
ORDER BY volume DESC;
```

Query 6 - Top ten employees, considering the number of orders sold.

```
MATCH (:Order)<-[:Sold]-(e:Employee)
RETURN e.firstName,e.lastName, count(*) AS Ordenes
ORDER BY Ordenes DESC LIMIT 10
```

Query 7 - For each employee, list the assigned territories.

```
MATCH (t:Territory)<-[:AssignedTo]-(e:Employee)
RETURN e.lastName, COLLECT(t.name);
```

Query 8 - For each city, list the companies settled in that city.

```
MATCH (c:City)<-[:locatedIn]-(c1:Customer)
RETURN c.cityname, COLLECT(c1.customerName);
```

Query 9 - How many persons an employee reports to, either directly or transitively?

```
MATCH (report:Employee)
OPTIONAL MATCH (e)<-[rel:ReportsTo*]-(report)
RETURN report.lastName AS e1, COUNT(rel) AS reports
```

Query 10 - To whom do persons called "Robert" report to?

```
MATCH (e:Employee)<-[:ReportsTo*]-(sub:Employee)
WHERE sub.firstName = 'Robert'
RETURN e.firstName,e.lastName,sub.lastName
```

Query 11 - Who does not report to anybody?

```
MATCH (e:Employee)
WHERE NOT (e)-[:ReportsTo]->()
RETURN e.firstName  as TopBossFirst, e.lastName as TopBossLast
```

Query 12 - Suppliers, number of categories they supply, and a list of such categories

```
MATCH (s:Supplier)-->(:Product)-->(c:Category)
WITH s.supplierName as Supplier, collect(distinct c.categoryName) as Categories
WITH Supplier, Categories, size(Categories) AS Cantidad ORDER BY Cantidad DESC
RETURN Supplier, Cantidad, Categories;
```

Query 13 - Suppliers who supply beverages

```
MATCH (c:Category {categoryName:"Beverages"})<--(:Product)<--(s:Supplier)
RETURN DISTINCT s.supplierName as ProduceSuppliers;
```

Query 14 - Customer who purchases the largest amount of beverages

```
MATCH (cust:Customer)<-[:Purchased]-(:Order)-[o:Contains]->(p:Product),
(p)-[:hasCategory]->(c:Category {categoryName:"Beverages"})
RETURN DISTINCT cust.customerName as CustomerName, SUM(toInteger(o.quantity))
as cant ORDER by  cant DESC LIMIT 1
```

Query 15 - List the five most popular products (considering number of orders)

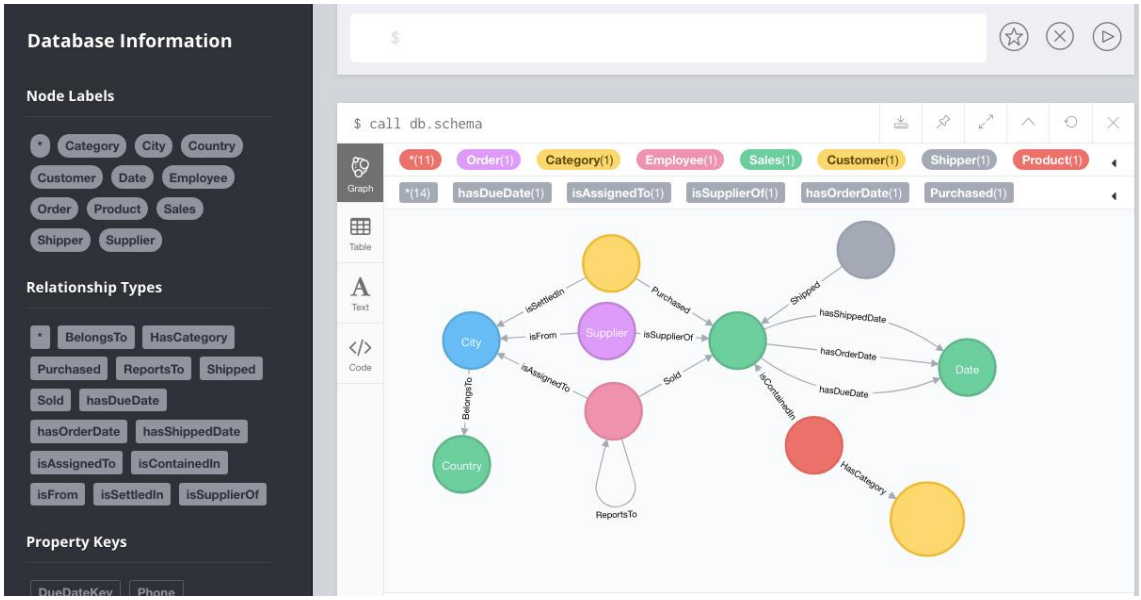
```
MATCH (c:Customer)<-[:Purchased]-(o:Order)-[o1:Contains]->(p:Product)
return c.customerName, p.productName, count(o1) as orders
order by orders desc  LIMIT 5
```

Query 16 - Products ordered by customers from the same country than their suppliers

```
MATCH (c:Customer)-[r:locatedIn]->(cy:City) -[:belongsTo]-(:Region)-[:isIn]-
>(co:Country)
WITH co,c MATCH (s:Supplier) WHERE co.countryname=s.country
WITH s,co,c MATCH (s)-[su:Supplies]->(p:Product) <-[:Contains]-(o:Order)-
[:Purchased]->(c)
RETURN c.customerName,s.supplierName,co.countryname,p.productName
```

Exercise 2.

Switch to the northwinddw database, doing the **same steps as in Assignment 2.** Now, the database is **northwinddw**. The schema is:



Write in Cypher the following queries over the northwindDW.db database:

Query 1. Total sales amount per customer, year, and product category

```
MATCH (c:Category)<-[:HasCategory]-(:Product)<-[:Contains]-(s:Sales)-[:PurchasedBy]->(u:Customer)
MATCH (s)-[:HasOrderDate]->(d:Date)
RETURN u.CompanyName AS Customer, c.CategoryName AS Category, d.Year AS Year, sum(s.SalesAmount) AS SalesAmount
ORDER BY Customer, Year, Category
```

Query 2. Yearly sales amount for each pair of customer and supplier countries

```
MATCH (cc:Country)<-[:OfCountry]-(:State)<-[:OfState]-(:City)<-[:IsLocatedIn]-(:Customer)<-[:PurchasedBy]-(s:Sales)
MATCH (d:Date)<-[:HasOrderDate]-(s:Sales)-[:SuppliedBy]->(:Supplier)-[:IsFrom]->(:City)-[:OfState]->(:State)-[:OfCountry]->(sc:Country)
RETURN cc.CountryName AS CustomerCountry, sc.CountryName AS SupplierCountry, d.Year AS Year, sum(s.SalesAmount) AS SalesAmount
ORDER BY CustomerCountry, SupplierCountry, Year
```

Query 3 - Monthly sales by customer country compared to those of the previous year

```
MATCH (d:Date)
MATCH (:Customer)-[:IsLocatedIn]->(:City)-[:OfState]->(t:State)
WITH DISTINCT t.StateName AS State, d.Year AS Year, d.MonthNumber AS Month
// Sales amount by month and state including months without sales
```

```
OPTIONAL MATCH (d:Date)<-[:HasOrderDate]-(s:Sales)-[:PurchasedBy]->
(:Customer)-[:IsLocatedIn]->(:City)-[:OfState]->(t:State)
WHERE t.StateName = State AND d.Year = Year AND d.MonthNumber = Month
WITH State, Year, Month, sum(s.SalesAmount) AS SalesAmount
ORDER BY State, Year, Month
WITH State, collect({y:Year, m:Month, s:SalesAmount}) AS rows
UNWIND range(0, size(rows) - 1) AS i
// Same month previous year using row number
WITH State, rows[i].y AS Year, rows[i].m AS Month, rows[i].s AS SalesAmount,
CASE WHEN i-12 < 0 THEN 0 ELSE rows[i-12].s END AS SalesAmountPY
WHERE SalesAmount <> 0 OR SalesAmountPY <> 0
RETURN State, Year, Month, SalesAmount, SalesAmountPY
ORDER BY State, Year, Month
```

Query 4 Monthly sales growth per product, that is, total sales per product compared to those of the previous month

```
MATCH (d:Date)
MATCH (p:Product)
WITH DISTINCT p.ProductName AS Product, d.Year AS Year, d.MonthNumber AS
Month
// Sales amount by month and product including months without sales
OPTIONAL MATCH (p:Product)<-[:Contains]-(s:Sales)-[:HasOrderDate]->(d:Date)
WHERE p.ProductName = Product AND d.Year = Year AND d.MonthNumber = Month
WITH Product, Year, Month, sum(s.SalesAmount) AS SalesAmount
ORDER BY Product, Year, Month
// Add row number
WITH Product, collect({y:Year, m:Month, s:SalesAmount}) AS rows
UNWIND range(0, size(rows) - 1) AS i
// Previous month using row number
WITH Product, rows[i].y AS Year, rows[i].m AS Month, rows[i].s AS SalesAmount,
CASE WHEN i-1 < 0 THEN 0 ELSE rows[i-1].s END AS SalesAmountPM
RETURN Product, Year, Month, SalesAmount, SalesAmountPM,
SalesAmount - SalesAmountPM AS SalesGrowth
ORDER BY Product, Year, Month
```

Query 5. Three best-selling employees

```
MATCH (e:Employee)<-[:HandledBy]-(s:Sales)
RETURN e.FirstName + ' ' + e.LastName, SUM(s.SalesAmount) AS SalesAmount
ORDER BY SalesAmount DESC LIMIT 3
```

Query 6. Best-selling employee per product and year

```
MATCH (e:Employee)<-[:HandledBy]-(s:Sales)-[:Contains]->(p:Product)
MATCH (d:Date)<-[:HasOrderDate]-(s)
WITH p.ProductName AS Product, d.Year AS Year, e.FirstName + ' ' + e.LastName AS
Employee, SUM(s.SalesAmount) AS SalesAmount
ORDER BY SalesAmount
WITH Product,Year,apoc.agg.maxItems(Employee,SalesAmount) as maxdata
WITH Product,Year,maxdata.items as BestEmp, maxdata.value as BestSales
UNWIND BestEmp as BestEmployee
RETURN Product, Year, BestEmployee,BestSales
ORDER BY Product,Year
```

Query 7. Countries that account for top 50\% of the sales amount

```
MATCH (c:Country)<-[:OfCountry]-(:State)<-[:OfState]-(:City)<-[:IsLocatedIn]-
(:Customer)<-[:PurchasedBy]-(s:Sales)
WITH c.CountryName AS CountryName, sum(s.SalesAmount) AS SalesAmount
ORDER BY SalesAmount DESC
WITH collect({CountryName:CountryName, SalesAmount:SalesAmount}) AS Countries,
sum(SalesAmount) * 0.5 AS Sales50Perc

UNWIND Countries AS c

WITH c.CountryName AS CountryName, c.SalesAmount AS SalesAmount,
apoc.coll.sum([c1 IN Countries where c1.SalesAmount >= c.SalesAmount |
c1.SalesAmount ]) AS CumulSales, Sales50Perc
WITH collect({CountryName:CountryName, SalesAmount:SalesAmount,
CumulSales:CumulSales}) AS CumulCountries, Sales50Perc
UNWIND [c1 IN CumulCountries where c1.CumulSales <= Sales50Perc] +
[c1 IN CumulCountries where c1.CumulSales > Sales50Perc][0] AS r
RETURN r.CountryName AS CountryName, r.SalesAmount AS SalesAmount,
    r.CumulSales As CummulativeSales
ORDER BY r.SalesAmount DESC
```

Query 8 Total sales and average monthly sales by employee and year

```
MATCH (e:Employee)<-[:HandledBy]-(s:Sales)-[:HasOrderDate]->(d:Date)
WITH e.FirstName + ' ' + e.LastName AS Employee, d.Year AS Year,
d.MonthNumber AS Month, sum(s.SalesAmount) AS MonthSales
WITH Employee, Year, sum(MonthSales) AS YearSales, COUNT(*) AS count
RETURN Employee, Year, YearSales, YearSales / count AS avgMonthlySales
```

```
ORDER BY Employee, Year
```

Query 9. Total sales amount and total discount amount per product and month

```
MATCH (d:Date)<-[:HasOrderDate]-(s:Sales)-[:Contains]->(p:Product)
WITH p.ProductName AS Product, d.Year AS Year, d.MonthNumber AS Month,
sum(s.SalesAmount) AS SalesAmount,
sum(s.UnitPrice * s.Quantity * s.Discount) AS TotalDiscount
RETURN Product, Year, Month, SalesAmount, TotalDiscount
ORDER BY Product, Year, Month
```

Query 10. Monthly year-to-date sales for each product category

```
MATCH (c:Category)<-[:HasCategory]-(:Product)<-[:Contains]-(s:Sales)-
[:HasOrderDate]->(d:Date)
WITH c.CategoryName AS Category, d.Year AS Year, d.MonthNumber AS Month,
sum(s.SalesAmount) AS SalesAmount
MATCH (c1:Category)<-[:HasCategory]-(:Product)<-[:Contains]-(s1:Sales)-
[:HasOrderDate]->(d1:Date)
WHERE c1.CategoryName = Category AND
d1.MonthNumber <= Month AND d1.Year = Year
RETURN Category, Year, Month, SalesAmount,
SUM(s1.SalesAmount) AS YTDSalesAmount
ORDER BY Category, Year, Month
```

Query 11. Moving average over the last 3 months of the sales amount by product category

```
MATCH (d:Date)
MATCH (c:Category)
WITH DISTINCT c.CategoryName AS Category, d.Year AS Year, d.MonthNumber AS
Month
OPTIONAL MATCH (d:Date)<-[:HasOrderDate]-(s:Sales)-[:Contains]->(p:Product)-
[:HasCategory]->(c:Category)
WHERE c.CategoryName = Category AND d.Year = Year AND d.MonthNumber = Month
WITH Category, Year, Month, sum(s.SalesAmount) AS SalesAmount
ORDER BY Category, Year, Month
// Add row number
WITH Category, collect({y:Year, m:Month, s:SalesAmount}) AS rows
UNWIND range(0, size(rows) - 1) AS i
// Moving average using row number
WITH Category, rows[i].y AS Year, rows[i].m AS Month, rows[i].s AS SalesAmount,
rows[CASE WHEN i-2 < 0 THEN 0 ELSE i-2 END..i+1] AS Values
UNWIND Values AS Value
RETURN Category, Year, Month, SalesAmount, collect(Value.s) AS Values,
```

```
avg(Value.s) AS MovAvg3M
ORDER BY Category, Year, Month
```

Query 12. Personal sales amount made by an employee compared with the total sales amount made by herself and her subordinates during 1997

```
MATCH (e:Employee)<-[:HandledBy]-(s:Sales)-[:HasOrderDate]->(d:Date)
WHERE d.Year = 2017
WITH e, e.FirstName + ' ' + e.LastName AS Employee, sum(s.SalesAmount) AS PersSales
OPTIONAL MATCH (e2:Employee)-[:ReportsTo*]->(e:Employee)
OPTIONAL MATCH (e2:Employee)<-[:HandledBy]-(s:Sales)-[:HasOrderDate]->(d:Date)
WHERE d.Year = 2017
RETURN Employee, PersSales, sum(s.SalesAmount) AS SubordSales,
    (PersSales+sum(s.SalesAmount)) AS PersonalAndSubordSales
ORDER BY Employee
```

Query 13. Total sales amount, number of products, and sum of the quantities sold for each order

```
MATCH (p:Product)<-[:Contains]-(s:Sales)-[:HasOrderDate]->(d:Date)
WITH s.OrderNo AS Order, sum(s.UnitPrice * s.Quantity) AS OrderAmount,
sum(s.Quantity) AS TotalQty, count(*) AS NbrOfProducts
RETURN Order, OrderAmount, TotalQty, NbrOfProducts
ORDER BY Order
```

Query 14. For each month, total number of orders, total sales amount, and average sales amount by order

```
MATCH (s:Sales)-[:HasOrderDate]->(d:Date)
WITH d.Year AS Year, d.MonthNumber AS Month, s.OrderNo AS Order,
sum(s.SalesAmount) AS OrderAmount
RETURN Year, Month, count(*) AS NoOrders, sum(OrderAmount) AS SalesAmount,
avg(OrderAmount) AS AvgOrderAmount
ORDER BY Year, Month
```

Query 15. For each employee, total sales amount, number of cities, and number of states to which she is assigned

```
MATCH (e:Employee)<-[:HandledBy]-(s:Sales)
MATCH (e:Employee)-[:IsAssignedTo]->(c:City)-[:OfState]->(t:State)
RETURN e.FirstName + ' ' + e.LastName AS Employee,
sum(s.SalesAmount) AS SalesAmount, count(DISTINCT c) AS NoCities,
```

```
count(DISTINCT t) AS NoStates
ORDER BY Employee
```