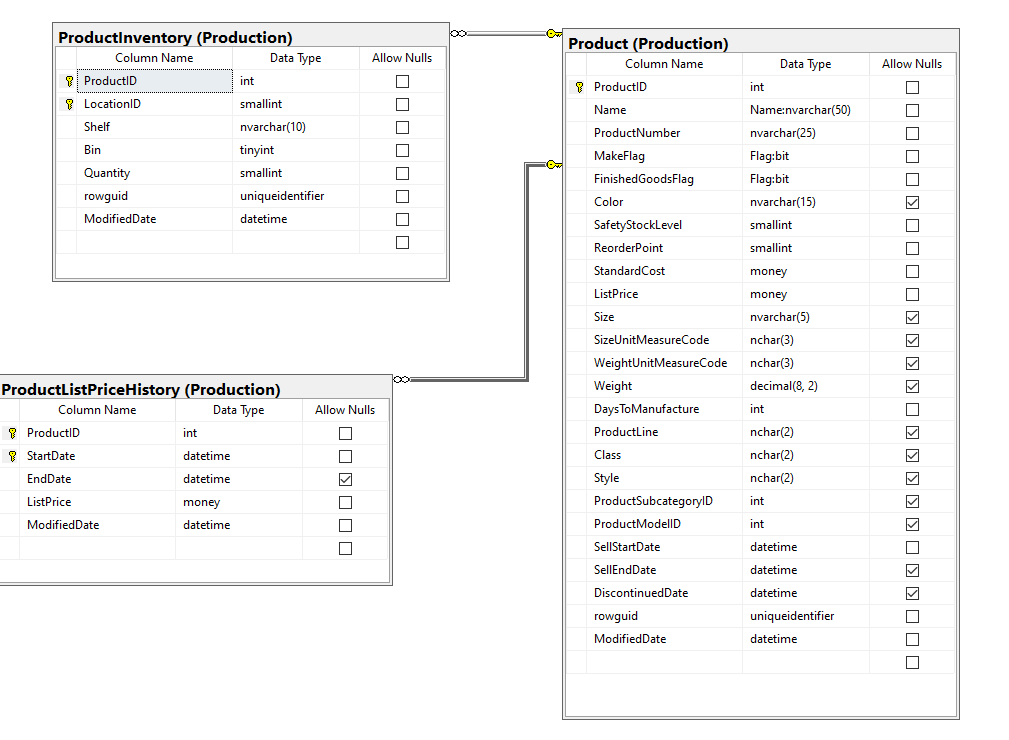
Proposition 1: What is the number of products with their listed price being in descending order in AdventureWorks2017 (354 rows affected)

## Detailed explanation of the problem

Include the the number of productIds, make sure the locationId is less than 50 with their listed price being in descending order greatest to least

## 

## Diagram(s) of tables



## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| ProductInventory | LocationID  ProductID |
| ProductListPriceHistory | ListPrice |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| ProductListPriceHistory | ListPrice | DESC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE AdventureWorks2017

GO

SELECT c.ProductID

,C.LocationID

,d.ListPrice

FROM [Production].[ProductInventory] AS c

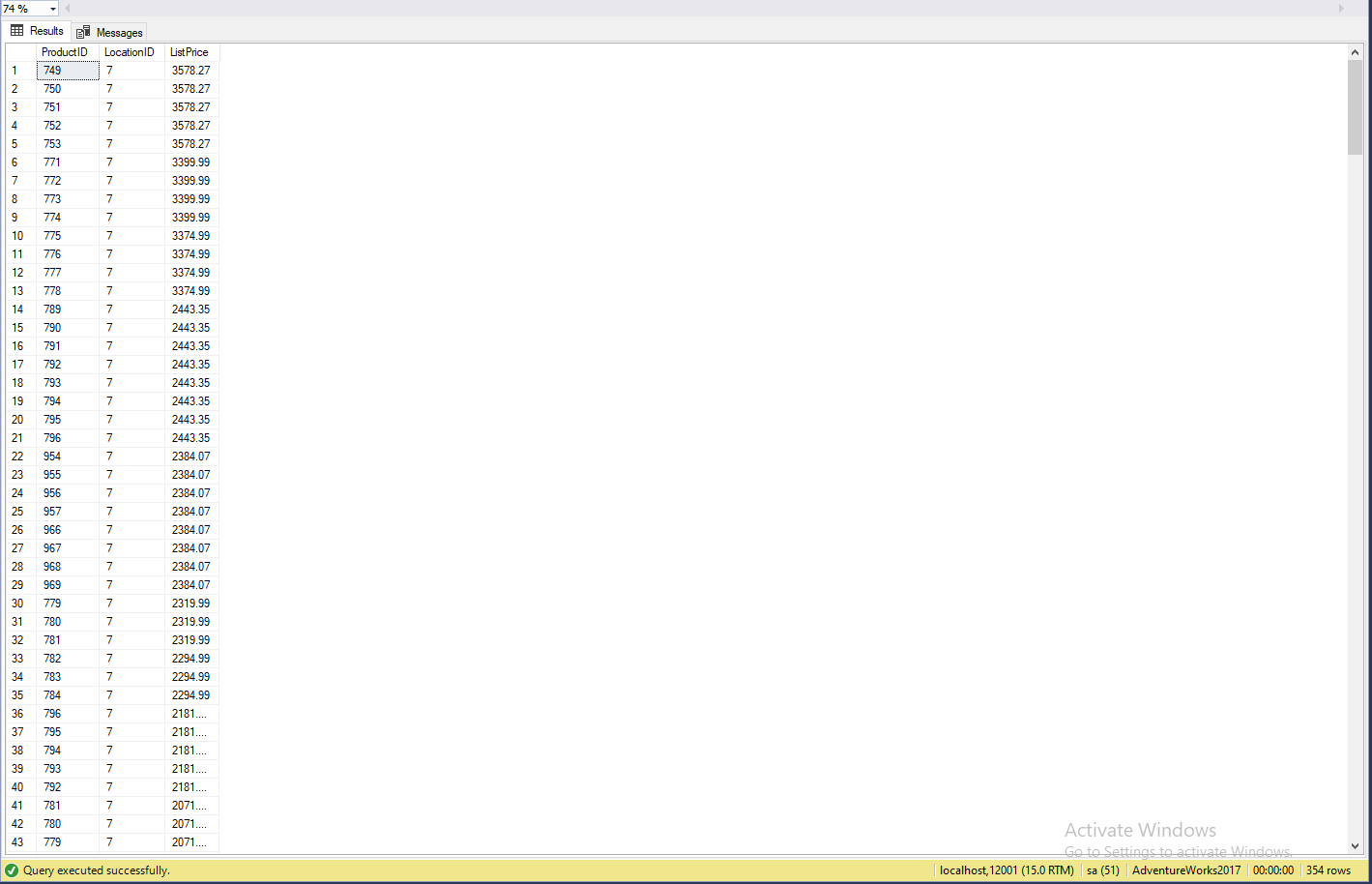
INNER JOIN [Production].[ProductListPriceHistory] AS d ON c.ProductID = d.ProductID

WHERE c.LocationID < 50

ORDER BY d.ListPrice DESC;

## 

## Problem Relational Output with total number of rows returned (354)



## 

## 

## 

## JSON Output with total number of rows returned (354)

USE AdventureWorks2017

GO

SELECT c.ProductID

,C.LocationID

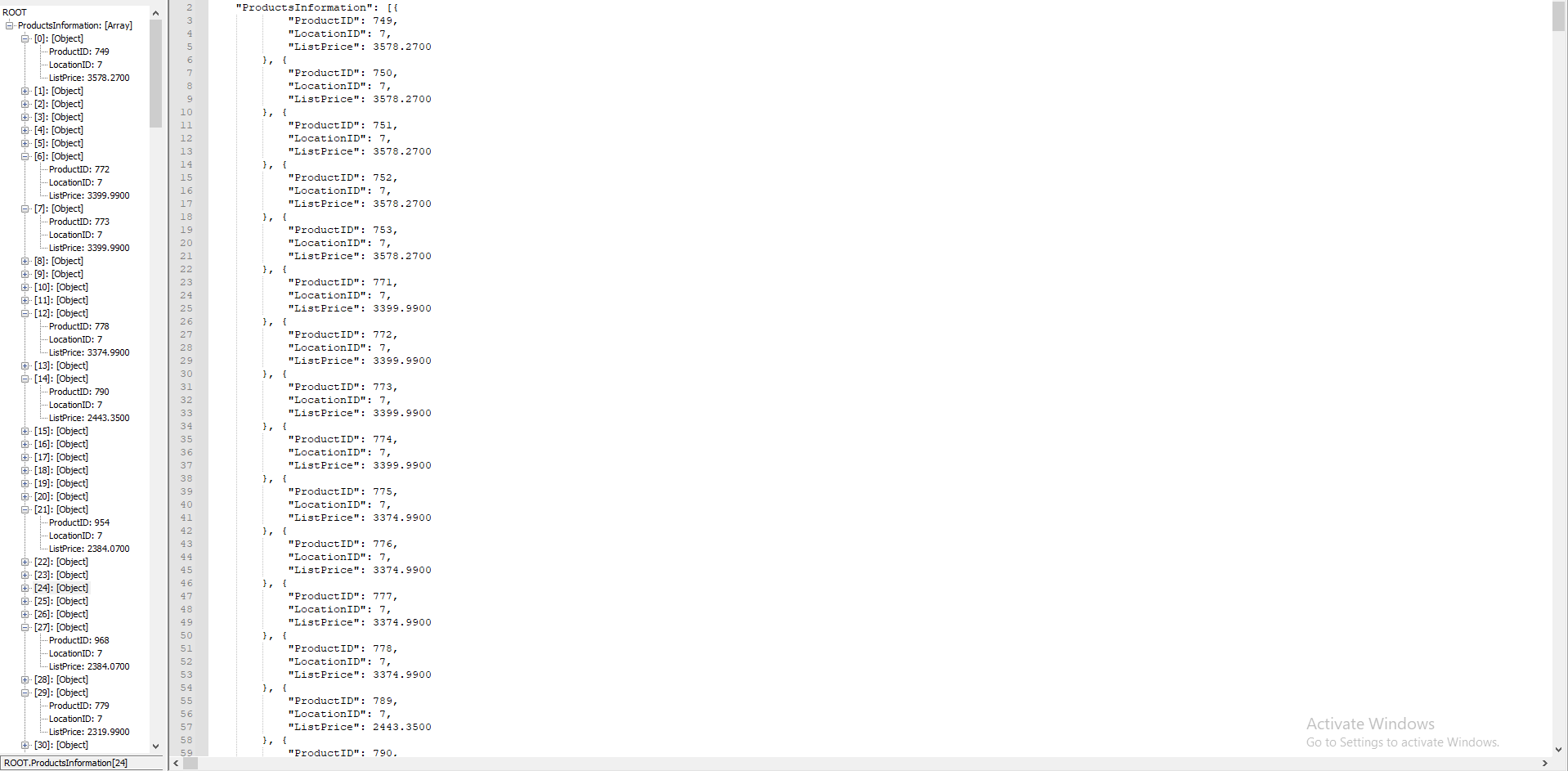
,d.ListPrice

FROM [Production].[ProductInventory] AS c

INNER JOIN [Production].[ProductListPriceHistory] AS d ON c.ProductID = d.ProductID

WHERE c.LocationID < 50

ORDER BY d.ListPrice DESC;

FOR JSON PATH, root('ProductsInformation'), include\_null\_values;

-----------------------------------------------------------------------------------------

Proposition 2: What number of employees in human resources are still working from their hire date

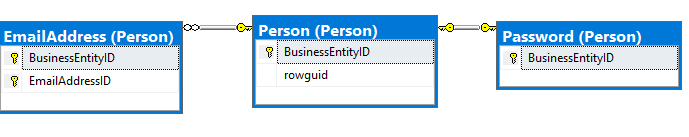
of 1/14/2009 in AdventureWorks2017 --(870 rows affected)

## Detailed explanation of the problem

Include the hiredate information and the End date information, where the hire date is 1/14/2009 and the end date is not available and join the tables of HumanResources.Employee and HumanResources.EmployeeDepartmentHistory

## 

## Diagram(s) of tables



## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Employee | HireDate |
| EmployeeDepartmentHistory | EndDate |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE AdventureWorks2017

SELECT e.HireDate

,d.EndDate

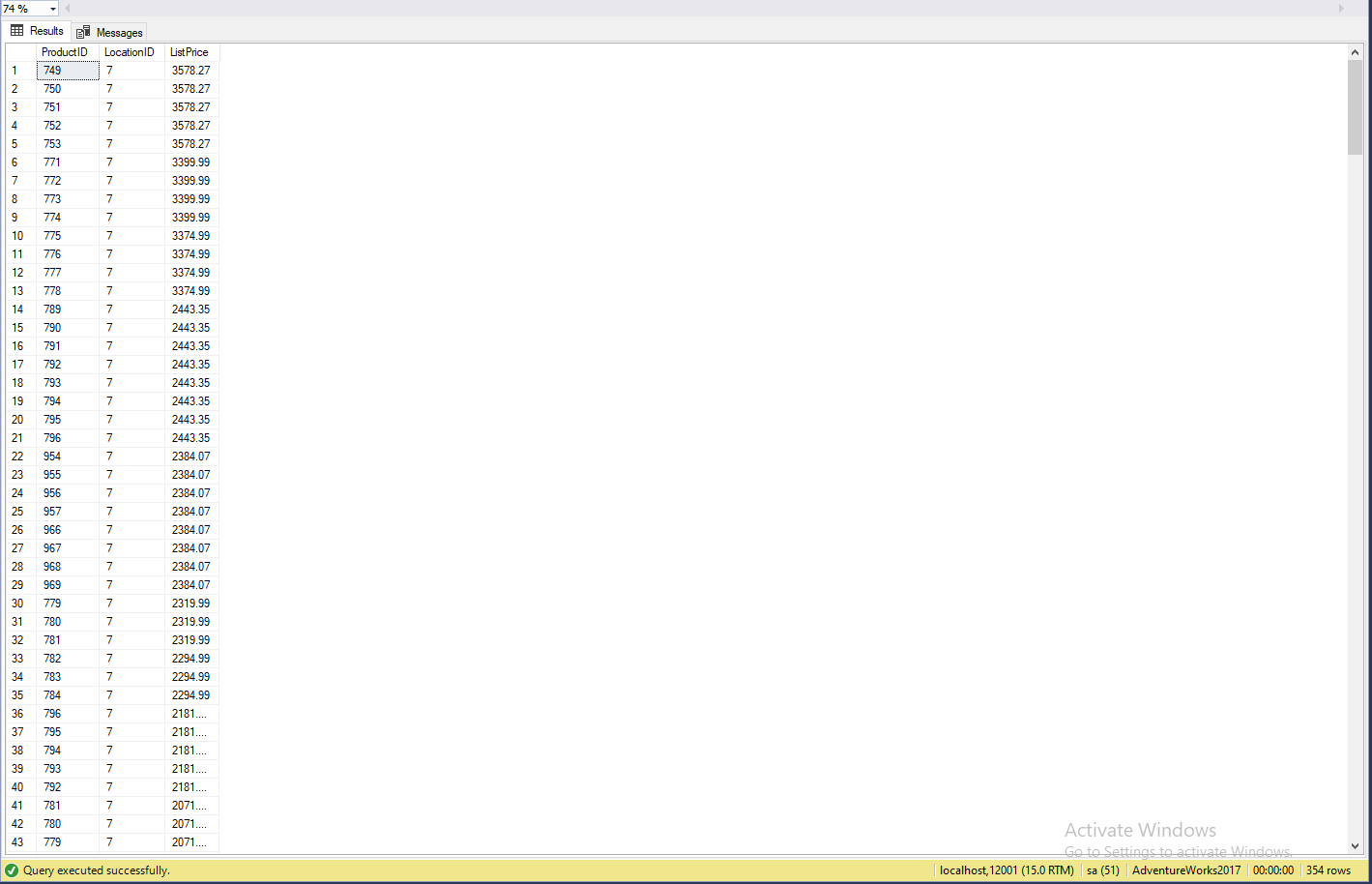
FROM HumanResources.EmployeeDepartmentHistory AS d

INNER JOIN [HumanResources].[Employee] AS e ON d.StartDate = d.StartDate

WHERE e.HireDate = '20090114'

AND d.EndDate IS NULL;

## Problem Relational Output with total number of rows returned (870)



## Sample JSON Output with total number of rows returned (870)

USE AdventureWorks2017

SELECT e.HireDate

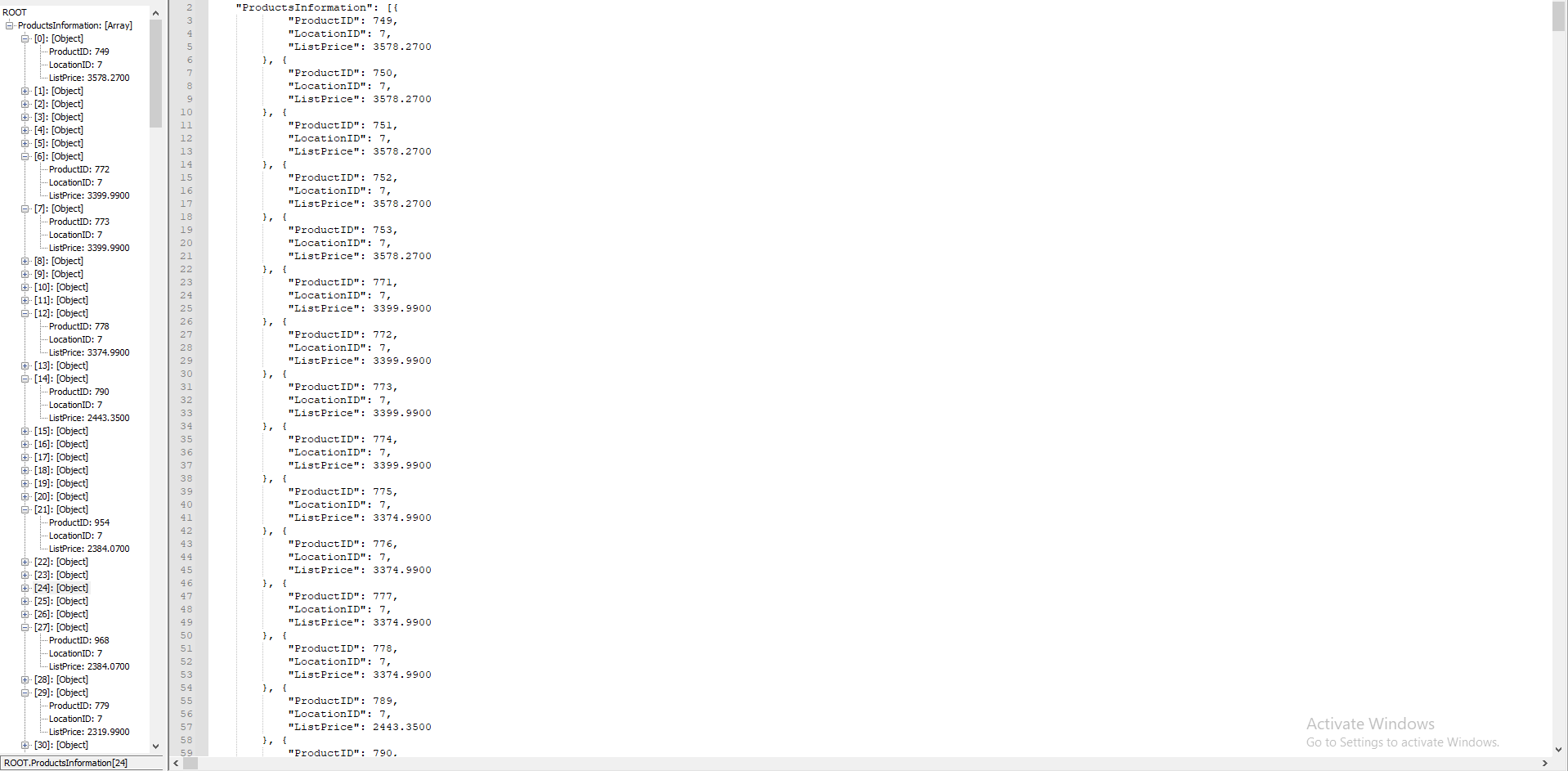
,d.EndDate

FROM HumanResources.EmployeeDepartmentHistory AS d

INNER JOIN [HumanResources].[Employee] AS e ON d.StartDate = d.StartDate

WHERE e.HireDate = '20090114'

AND d.EndDate IS NULL

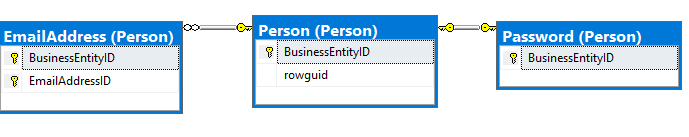
FOR JSON PATH, root('CurrentHires'), include\_null\_values;

-----------------------------------------------------------------------------------------

## Proposition 03: --The amount of people who edited their email address OR their password on the date of 10/16/2013 using AdventureWorks2017

## Detailed explanation of the problem

Find the last time a person has modified their Email Address or their password on the date of 10/16/2013



## 

## Columns from Standard view

## Select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| EmailAddress | ModifiedDate |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE AdventureWorks2017

go

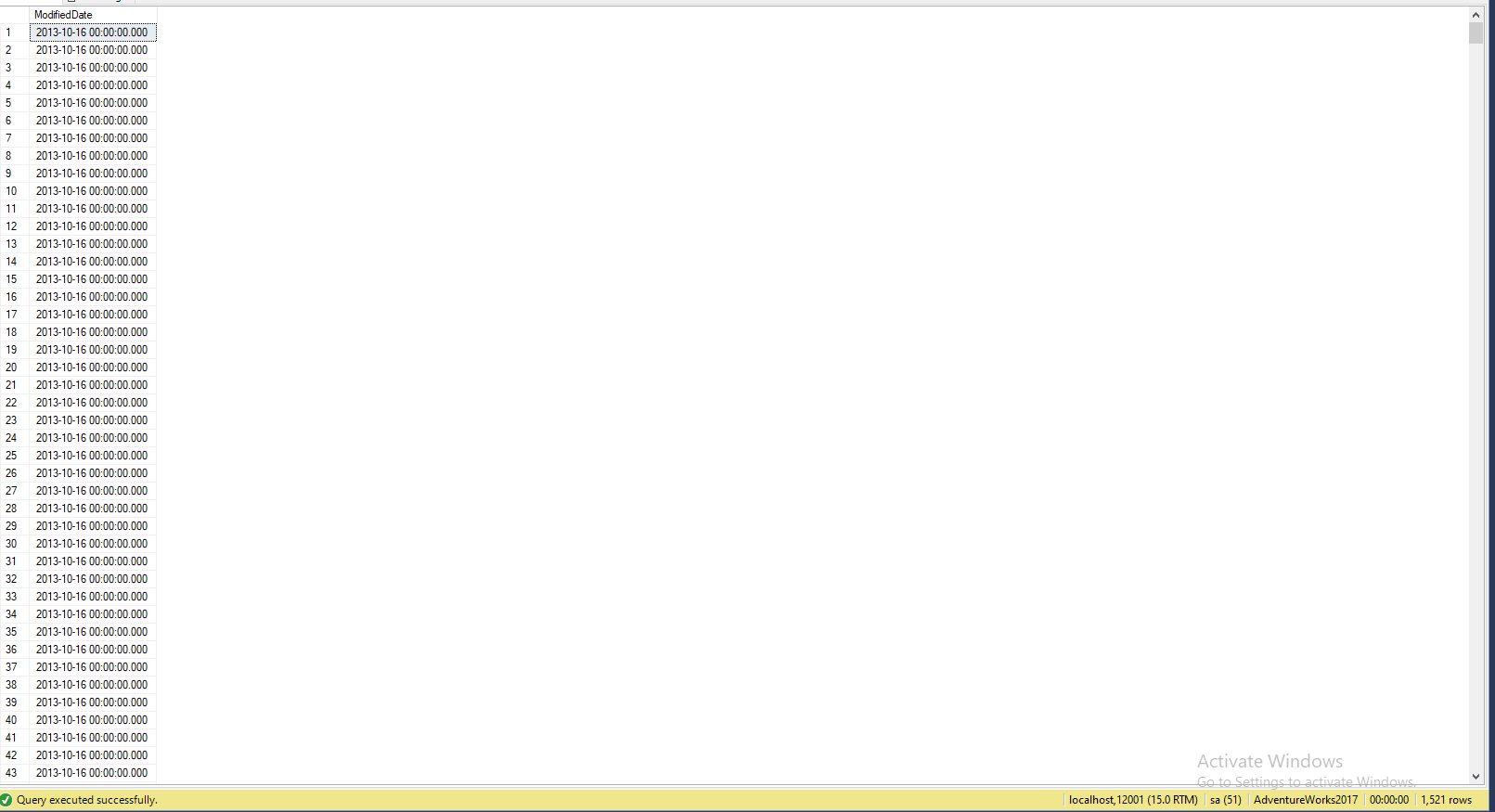
SELECT a.ModifiedDate

FROM person.EmailAddress AS a

INNER JOIN person.Password AS p ON a.ModifiedDate = p.ModifiedDate

WHERE a.ModifiedDate = '20131016'

## Output with total number of rows returned (1521)



JSON Output with total number of rows returned (1521)

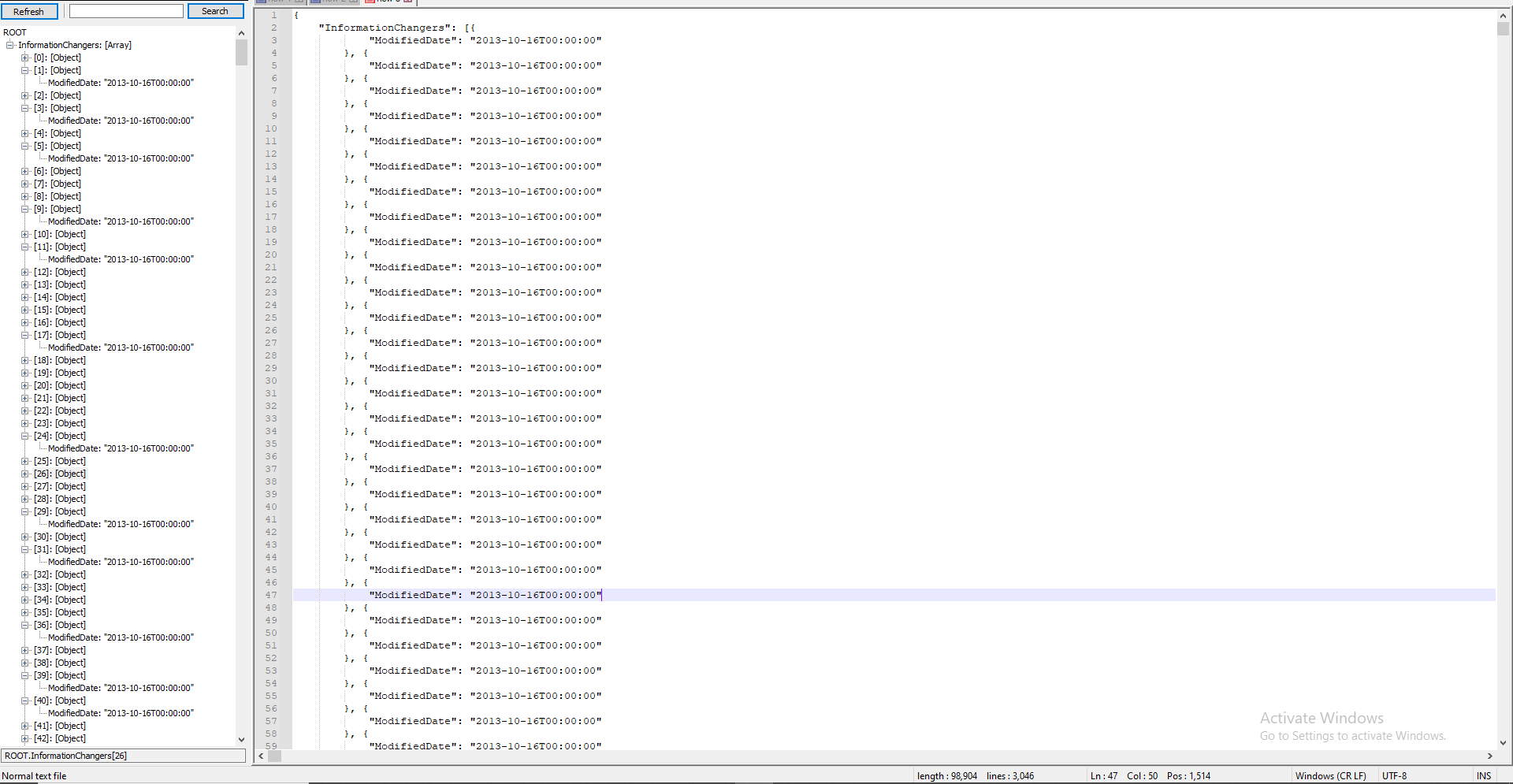
USE AdventureWorks2017

SELECT a.ModifiedDate

FROM person.EmailAddress AS a

INNER JOIN person.Password AS p ON a.ModifiedDate = p.ModifiedDate

WHERE a.ModifiedDate = '20131016'

FOR JSON PATH, root('InformationChangers'), include\_null\_values;

-----------------------------------------------------------------------------------------

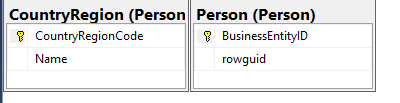
## Proposition 04: What is the amount of people in the Database of people who do not have a middle name AND the date it was modified not being null

## using AdventureWorks2017?

## Detailed explanation of the problem

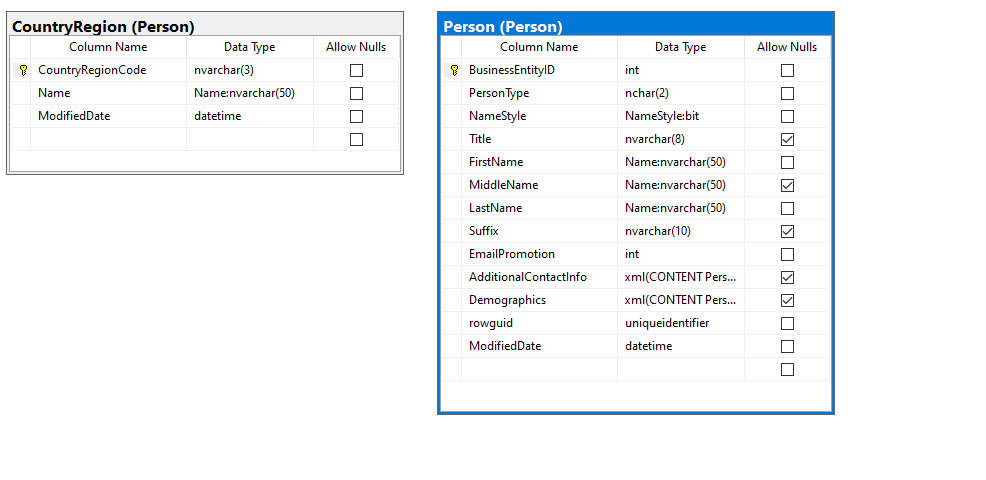
Find the First, Middle and Last name of the people in the database that do not have a middle name and was modified once

## Diagram(s) of tables



## 

## Standard view



## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Person | FirstName  MiddleName  LastName  ModifiedDate |
| Orders | OrderId  OrderDate |
|  |  |
|  |  |
|  |  |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
|  |  |  |
|  |  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

SELECT DISTINCT p.FirstName

,p.MiddleName

,p.LastName

,p.ModifiedDate

FROM [Person].[Person] AS p

FULL OUTER JOIN person.CountryRegion AS t ON t.ModifiedDate = p.ModifiedDate

WHERE p.MiddleName IS NULL

AND p.ModifiedDate IS NOT NULL

## 

## Output with total number of rows returned (8498)

## JSON Output with total number of rows returned (8498)

SELECT DISTINCT p.FirstName

,p.MiddleName

,p.LastName

,p.ModifiedDate

FROM [Person].[Person] AS p

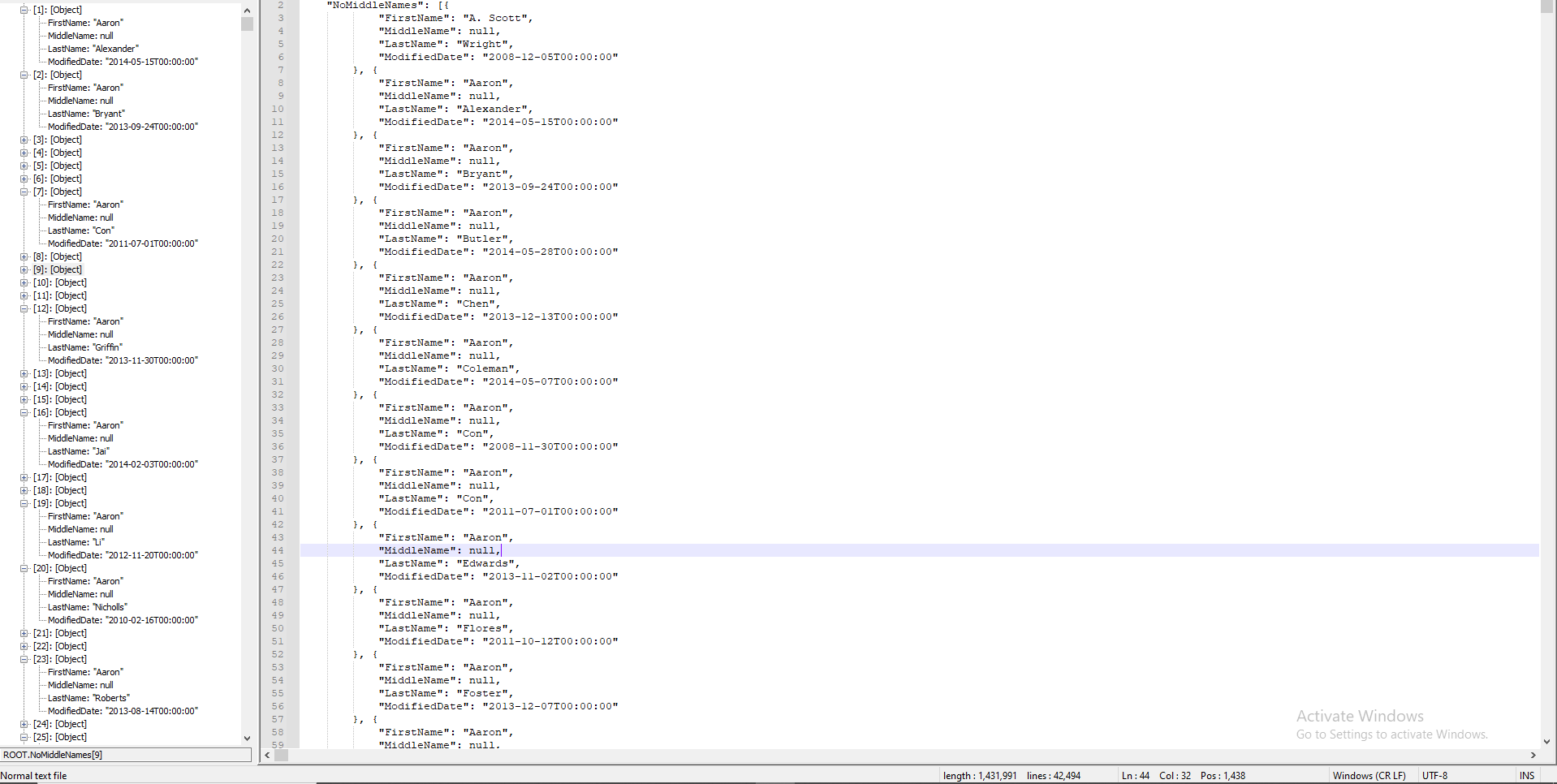
FULL OUTER JOIN person.CountryRegion AS t

ON t.ModifiedDate = p.ModifiedDate

WHERE p.MiddleName IS NULL

AND p.ModifiedDate IS NOT NULL

FOR JSON PATH, root('NoMiddleNames'), include\_null\_values;



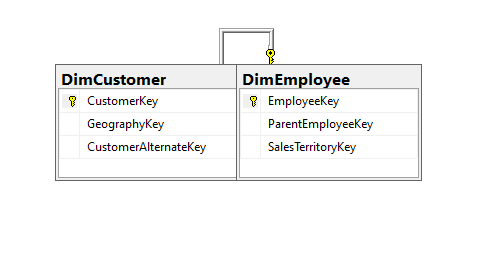
-----------------------------------------------------------------------------------------

## Proposition 05: What are the birthdates of customers and employees that have overlapped before the year 1950 using AdventureWorksDW2017?

## Detailed explanation of the problem

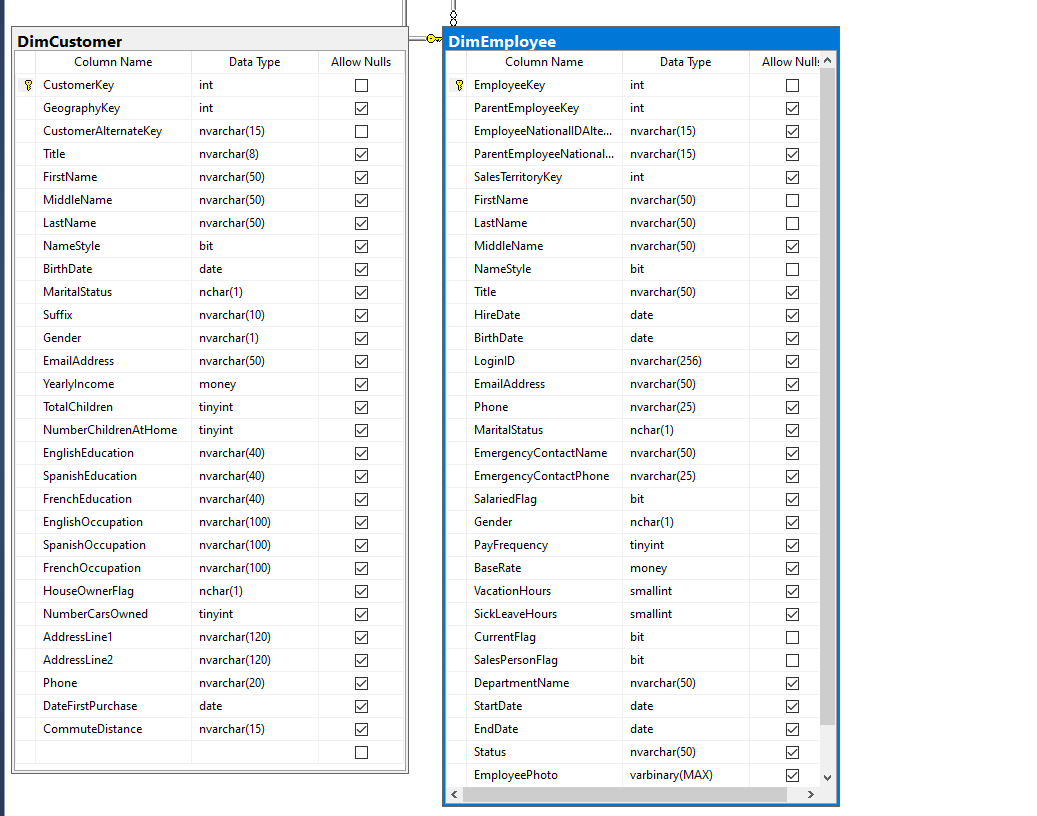
Find the Birthdates and join DimCustomer with DimEmployee and then make sure there is a constriction on the data to show only before 1950

## Diagram(s) of tables



## 

## Standard view



## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| DimCustomer | BirthDate |

## Problem solving Query

USE AdventureWorksDW2017

SELECT D.BirthDate

FROM dbo.DimCustomer AS D

INNER JOIN dbo.DimEmployee AS E ON E.BirthDate = D.BirthDate

WHERE YEAR(E.BirthDate) < 1950

## 

## Output with total number of rows returned (4)

## JSON Output with total number of rows returned (4)

SELECT DISTINCT p.FirstName

,p.MiddleName

,p.LastName

,p.ModifiedDate

FROM [Person].[Person] AS p

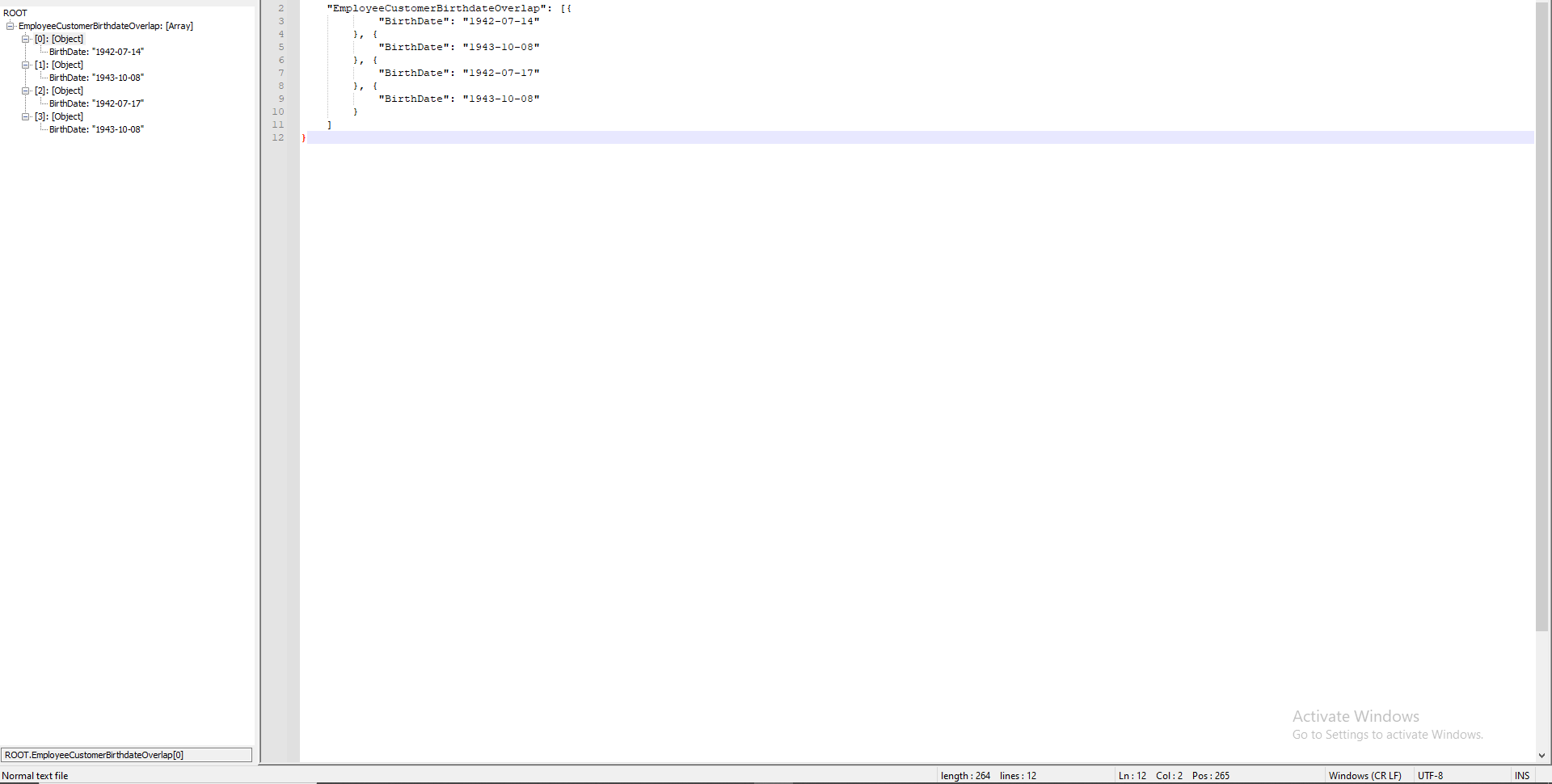
FULL OUTER JOIN person.CountryRegion AS t

ON t.ModifiedDate = p.ModifiedDate

WHERE p.MiddleName IS NULL

AND p.ModifiedDate IS NOT NULL

FOR JSON PATH, root('EmployeeCustomerBirthdateOverlap'), include\_null\_values;



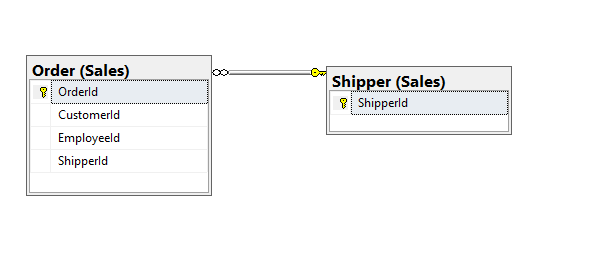
-----------------------------------------------------------------------------------------

## Proposition 06: What are the number of orders fulfilled by the Shipping company GVSUA,where the dates that they were required to be shipped out was before 2015? Using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the customers id and the orderid along with the shipper company name and the date that they expected the shipment to leave, add a where clause that finds out the shipper is GVSUA and that its shipdate is before 2015

## Diagram(s) of tables



## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | CustomerId  OrderId  ShipToDate |
| Shipper | ShipperCompanyName |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

SELECT st.CustomerId

,st.OrderId

,sh.ShipperCompanyName AS ShippingCompany

,st.ShipToDate AS ShippedDate

FROM [Sales].[Order] AS st

INNER JOIN [Sales].[Shipper] AS sh ON sh.ShipperId = st.ShipperId

WHERE sh.ShipperCompanyName = 'Shipper GVSUA'

AND YEAR(st.ShipToDate) < 2015

GROUP BY st.OrderId

,st.CustomerId

,st.ShipToDate

,sh.ShipperCompanyName;

## 

## Output with total number of rows returned (36)

## JSON Output with total number of rows returned (36)

USE Northwinds2020TSQLV6

SELECT st.CustomerId

,st.OrderId

,sh.ShipperCompanyName AS ShippingCompany

,st.ShipToDate AS ShippedDate

FROM [Sales].[Order] AS st

INNER JOIN [Sales].[Shipper] AS sh ON sh.ShipperId = st.ShipperId

WHERE sh.ShipperCompanyName = 'Shipper GVSUA'

AND YEAR(st.ShipToDate) < 2015

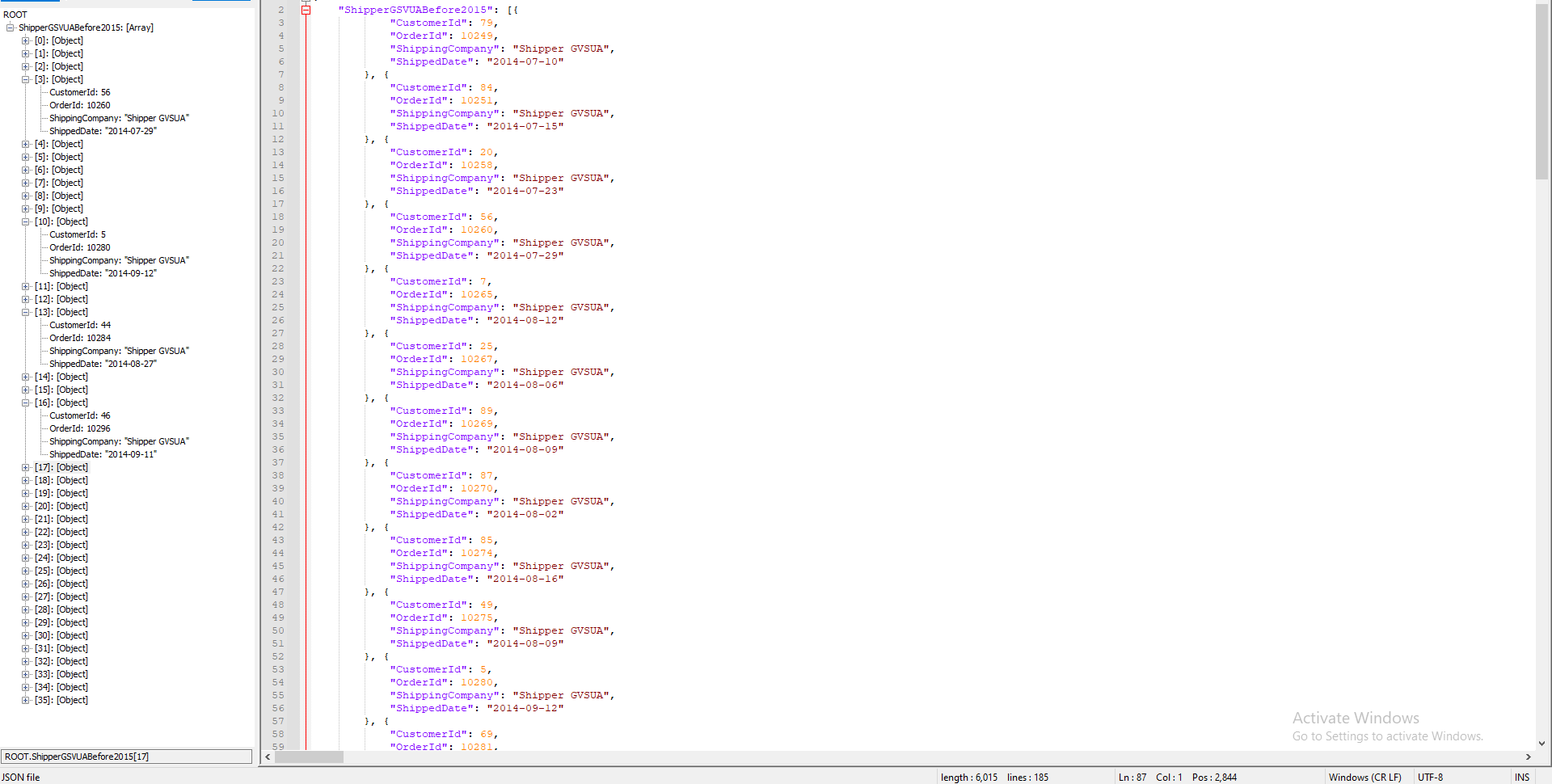
GROUP BY st.OrderId

,st.CustomerId

,st.ShipToDate

,sh.ShipperCompanyName

FOR JSON PATH, root('ShipperGSVUABefore2015'), include\_null\_values;



-----------------------------------------------------------------------------------------

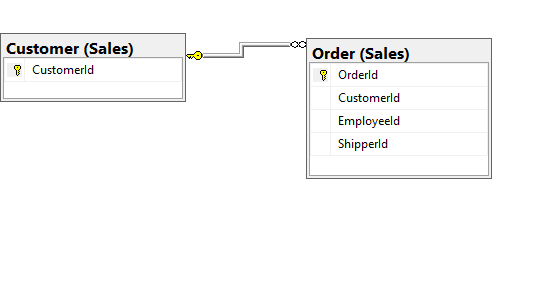
## Proposition 07: What are the Customer's names and their title, when they ordered

## and those who have ordered a very low overall amount of freight less than 10 using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the customer contact name and their title and find the orderdate of the shipments and find the amount of freight the customer ordered that was below 10

## Diagram(s) of tables



## 

## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerContactName  CystomerContactTitle |
| Order | OrderDate  Freight |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

SELECT c.CustomerContactName AS CustomerName

,c.CustomerContactTitle AS CustomerTitle

,o.OrderDate

,SUM(o.Freight) AS TotalFreightPerCustomer

FROM Sales.Customer AS c

INNER JOIN Sales.[order] AS o ON c.CustomerId = o.CustomerId

WHERE o.Freight < 10

GROUP BY o.Freight

,c.CustomerContactName

,c.CustomerContactTitle

,o.OrderDate

,o.ShipToPostalCode

## 

## Output with total number of rows returned (176)

## JSON Output with total number of rows returned (176)

SELECT c.CustomerContactName AS CustomerName

,c.CustomerContactTitle AS CustomerTitle

,o.OrderDate

,SUM(o.Freight) AS TotalFreightPerCustomer

FROM Sales.Customer AS c

INNER JOIN Sales.[order] AS o ON c.CustomerId = o.CustomerId

WHERE o.Freight < 10

GROUP BY o.Freight

,c.CustomerContactName

,c.CustomerContactTitle

,o.OrderDate

,o.ShipToPostalCode

FOR JSON PATH, root('OrderedLessThan10Freight'), include\_null\_values;



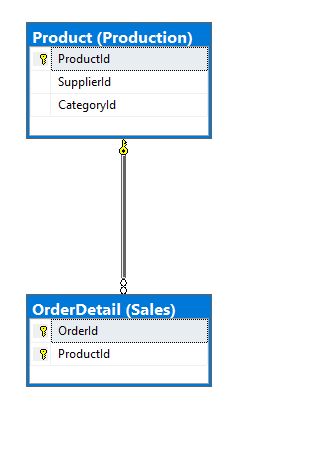
-----------------------------------------------------------------------------------------

## Proposition 08: What is the information on our largest amount of products being ordered, greater than 100Quantity and what is the final price they receive and the amount of their discounts using NorthWinds2020TSQLV6

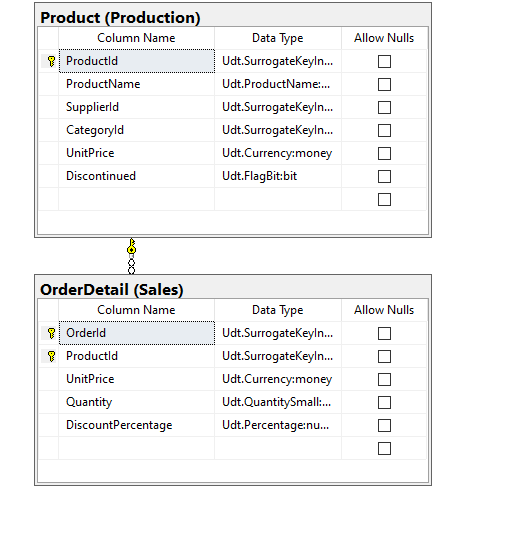
## Detailed explanation of the problem

Find the buyers who order the largest quantity of products, the amount being over 100 and then show the final price they received and how much of a discount they got for their order

## Diagram(s) of tables



## Standard view



## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Product | ProductName |
| OrderDetail | ProductId  UnitPrice  Quantity  DiscountPercentage |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

SELECT DISTINCT (o.OrderId)

,o.ProductId

,p.ProductName

,o.UnitPrice

,o.Quantity

,o.DiscountPercentage

,(o.UnitPrice \* o.Quantity) - ((o.UnitPrice \* o.Quantity) \* o.DiscountPercentage) AS FinalPrice

FROM Production.Product AS p

INNER JOIN Sales.OrderDetail AS o ON p.ProductId = o.ProductId

WHERE o.Quantity > 100

GROUP BY (o.OrderId)

,o.ProductId

,p.ProductName

,o.UnitPrice

,o.Quantity

,o.DiscountPercentage

## 

## Output with total number of rows returned (13)

## JSON Output with total number of rows returned (13)

USE Northwinds2020TSQLV6

SELECT DISTINCT (o.OrderId)

,o.ProductId

,p.ProductName

,o.UnitPrice

,o.Quantity

,o.DiscountPercentage

,(o.UnitPrice \* o.Quantity) - ((o.UnitPrice \* o.Quantity) \* o.DiscountPercentage) AS FinalPrice

FROM Production.Product AS p

INNER JOIN Sales.OrderDetail AS o ON p.ProductId = o.ProductId

WHERE o.Quantity > 100

GROUP BY (o.OrderId)

,o.ProductId

,p.ProductName

,o.UnitPrice

,o.Quantity

,o.DiscountPercentage

FOR JSON PATH, root('BiggestBuyersQuantity'), include\_null\_values;



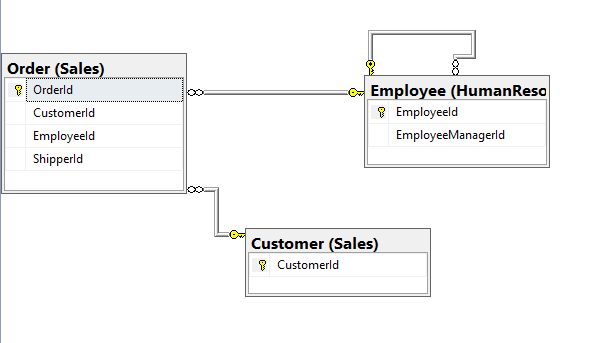
-----------------------------------------------------------------------------------------

## Proposition 09: Which customers and which employees live in the same general area, within the same country, in the same region,in the same city using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the customers and employees which have the same country, same region and same city within given the HumanResources.Employee and the Sales.Customer Northwinds2020TSQLV6

## Diagram(s) of tables



## 

## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Employee | EmployeeID  EmployeeRegion  EmployeeCountry  EmployeeCity |
| Customer | CustomerRegion  CustomerCountry  CustomerCity |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

SELECT c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,e.EmployeeCountry

,c.CustomerCountry

,c.CustomerCity

,e.EmployeeCity

FROM [HumanResources].[Employee] AS e

INNER JOIN Sales.Customer AS c ON e.EmployeeCity = c.CustomerCity

INNER JOIN Sales.[Order] AS o ON o.ShipToCity = c.CustomerCity

WHERE c.CustomerRegion IS NOT NULL

AND e.EmployeeRegion IS NOT NULL

GROUP BY c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,e.EmployeeCountry

,c.CustomerCountry

,c.CustomerCity

,e.EmployeeCity

## 

## Output with total number of rows returned (3)

## JSON Output with total number of rows returned (3)

SELECT c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,e.EmployeeCountry

,c.CustomerCountry

,c.CustomerCity

,e.EmployeeCity

FROM [HumanResources].[Employee] AS e

INNER JOIN Sales.Customer AS c ON e.EmployeeCity = c.CustomerCity

INNER JOIN Sales.[Order] AS o ON o.ShipToCity = c.CustomerCity

WHERE c.CustomerRegion IS NOT NULL

AND e.EmployeeRegion IS NOT NULL

GROUP BY c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,e.EmployeeCountry

,c.CustomerCountry

,c.CustomerCity

,e.EmployeeCity

FOR JSON PATH, root('CustomerEmployeeSameArea'), include\_null\_values;



-----------------------------------------------------------------------------------------

## Proposition 10: What was the earliest order dates in the table and who was the supplier and where were they located for shipping the order using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the OrderId, orderdate, customerid, employeeid as well as the suppliercompanyname and shiptocountry and the suppliercountry within the sales.order and production.supplier

## Diagram(s) of tables

## 

## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId  OrderDate  CustomerId  EmployeeId  ShipToCountry |
| Supplier | SupplierCountry  SupplierCompanyName |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE NorthWinds2020TSQLV6

SELECT o.Orderid

,o.Orderdate

,o.CustomerId

,o.EmployeeId

,s.SupplierCompanyName

,o.ShipToCountry

,s.SupplierCountry

FROM Sales.[order] AS o

INNER JOIN Production.Supplier AS s ON o.ShiptoCountry = s.SupplierCountry

WHERE o.Orderdate = (

SELECT MIN(O.orderdate)

FROM Sales.[Order] AS O

)

GROUP BY o.Orderid

,o.Orderdate

,o.CustomerId

,o.EmployeeId

,s.SupplierCompanyName

,o.ShipToCountry

,s.SupplierCountry

## 

## Output with total number of rows returned (3)

## JSON Output with total number of rows returned (3)

USE NorthWinds2020TSQLV6

SELECT o.Orderid

,o.Orderdate

,o.CustomerId

,o.EmployeeId

,s.SupplierCompanyName

,o.ShipToCountry

,s.SupplierCountry

FROM Sales.[order] AS o

INNER JOIN Production.Supplier AS s ON o.ShiptoCountry = s.SupplierCountry

WHERE o.Orderdate = (

SELECT MIN(O.orderdate)

FROM Sales.[Order] AS O

)

GROUP BY o.Orderid

,o.Orderdate

,o.CustomerId

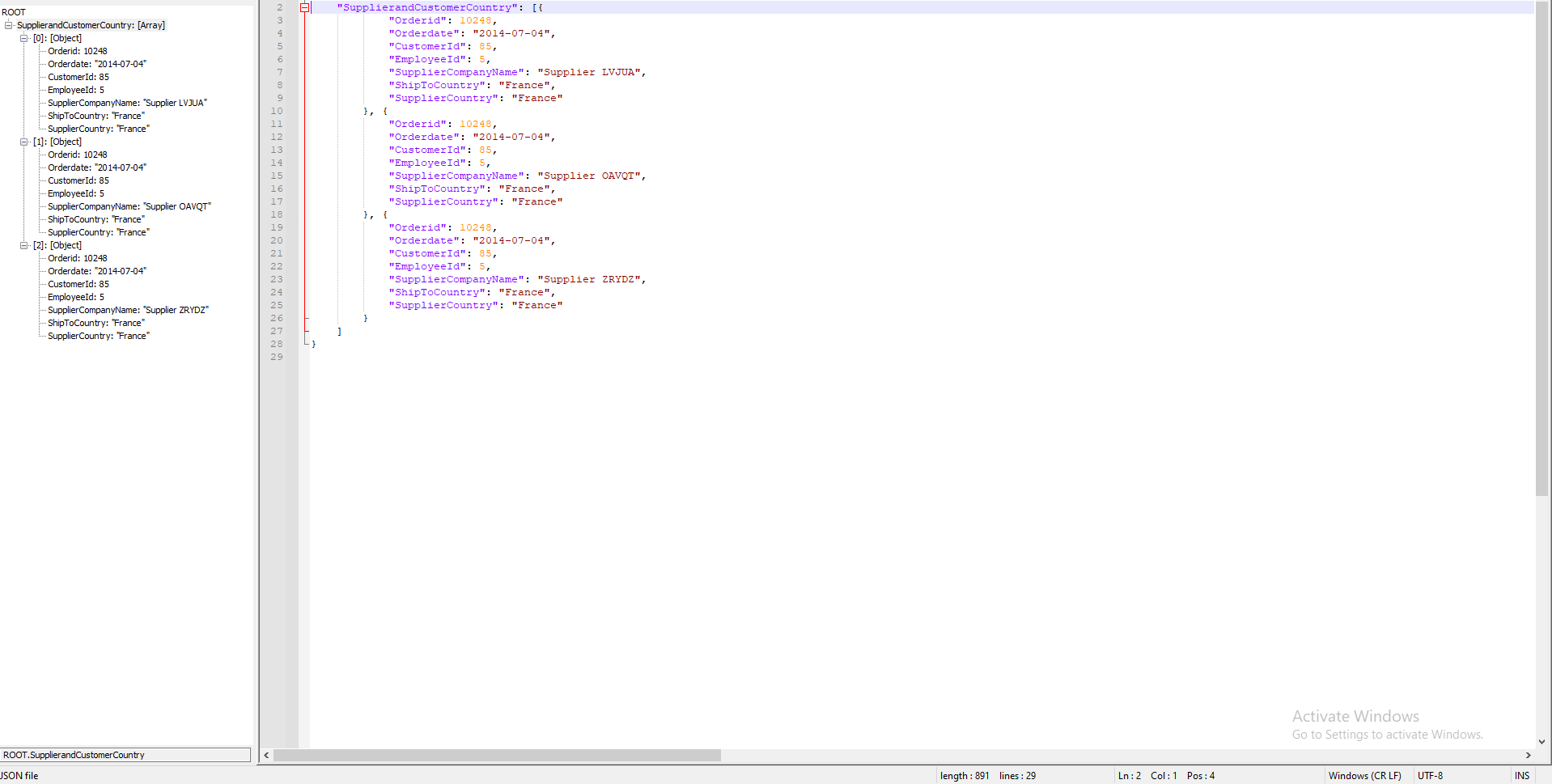
,o.EmployeeId

,s.SupplierCompanyName

,o.ShipToCountry

,s.SupplierCountry

FOR JSON PATH, root('SupplierandCustomerCountry'), include\_null\_values;



-----------------------------------------------------------------------------------------

## Proposition 11: What is the contact information of the customer only in a country where employees exist, in alphabetical order of the country using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the contact information of the customer through fax and phone number as well as the country they are currently located in and find the country in which employees exist in.

## Diagram(s) of tables

## 

## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerCountry  CustomerPhoneNumber  CustomerFaxNumber |
| Employee | EmployeeCountry |
|  |  |
|  |  |
|  |  |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Customer | CustomerCountry | ASC |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

SELECT c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

FROM Sales.Customer AS c

OUTER APPLY HumanResources.Employee AS hr

WHERE c.CustomerCountry IN(

SELECT E.[EmployeeCountry]

FROM [HumanResources].[Employee] AS E

)

GROUP BY c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

ORDER BY c.CustomerCountry

## 

## Output with total number of rows returned (20)

## JSON Output with total number of rows returned (20)

USE Northwinds2020TSQLV6

SELECT c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

FROM Sales.Customer AS c

OUTER APPLY HumanResources.Employee AS hr

WHERE c.CustomerCountry IN(

SELECT E.[EmployeeCountry]

FROM [HumanResources].[Employee] AS E

)

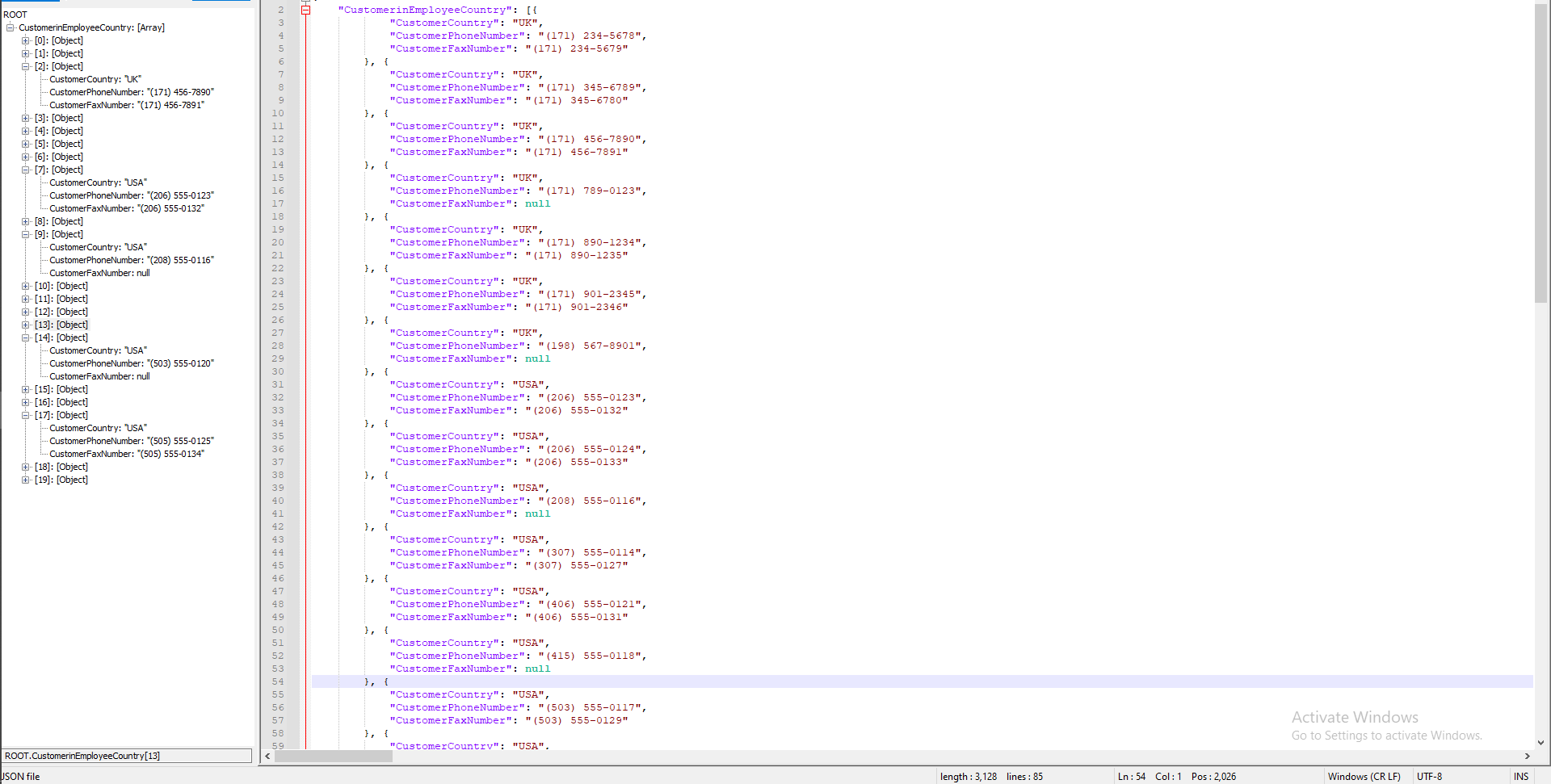
GROUP BY c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

ORDER BY c.CustomerCountry

FOR JSON PATH, root('CustomerinEmployeeCountry'), include\_null\_values;



-----------------------------------------------------------------------------------------

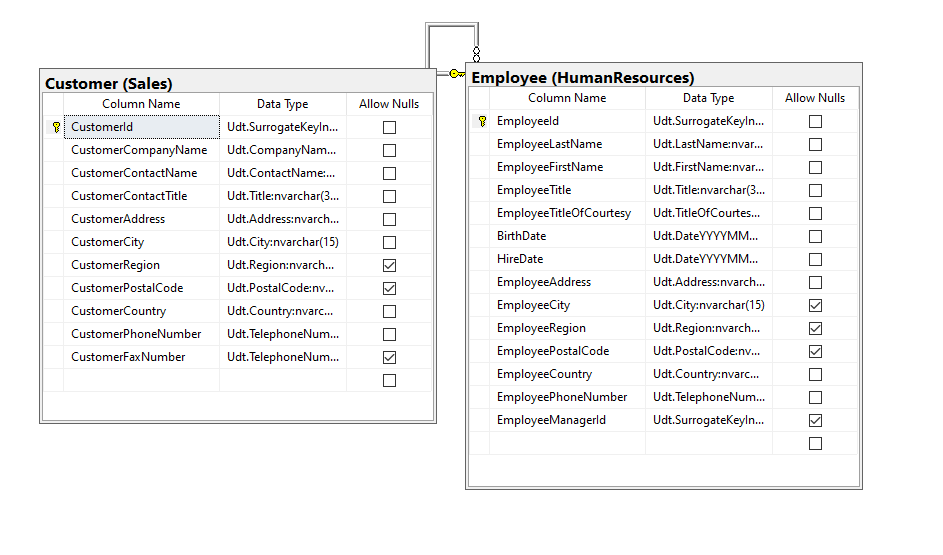
## Proposition 12: What is the contact information of the customer in a country where there are no employees, in alphabetical order of country using NorthWinds2020TSQLV6

## Detailed explanation of the problem

Find the OrderId, orderdate, customerid, employeeid as well as the suppliercompanyname and shiptocountry and the suppliercountry within the sales.order and production.supplier

## Diagram(s) of tables

## Standard view

Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | OrderId  OrderDate  CustomerId  EmployeeId  ShipToCountry |
| Employee | EmployeeCountry |
|  |  |
|  |  |
|  |  |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Customer | CustomerCountry | ASC |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE NorthWinds2020TSQLV6

SELECT c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

FROM Sales.Customer AS c

OUTER APPLY HumanResources.Employee

WHERE c.CustomerCountry NOT IN (

SELECT E.[EmployeeCountry]

FROM [HumanResources].[Employee] AS E

)

GROUP BY c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

ORDER BY c.CustomerCountry

## 

## Output with total number of rows returned (71)

## JSON Output with total number of rows returned (71)

USE NorthWinds2020TSQLV6

SELECT c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

FROM Sales.Customer AS c

OUTER APPLY HumanResources.Employee

WHERE c.CustomerCountry NOT IN (

SELECT E.[EmployeeCountry]

FROM [HumanResources].[Employee] AS E

)

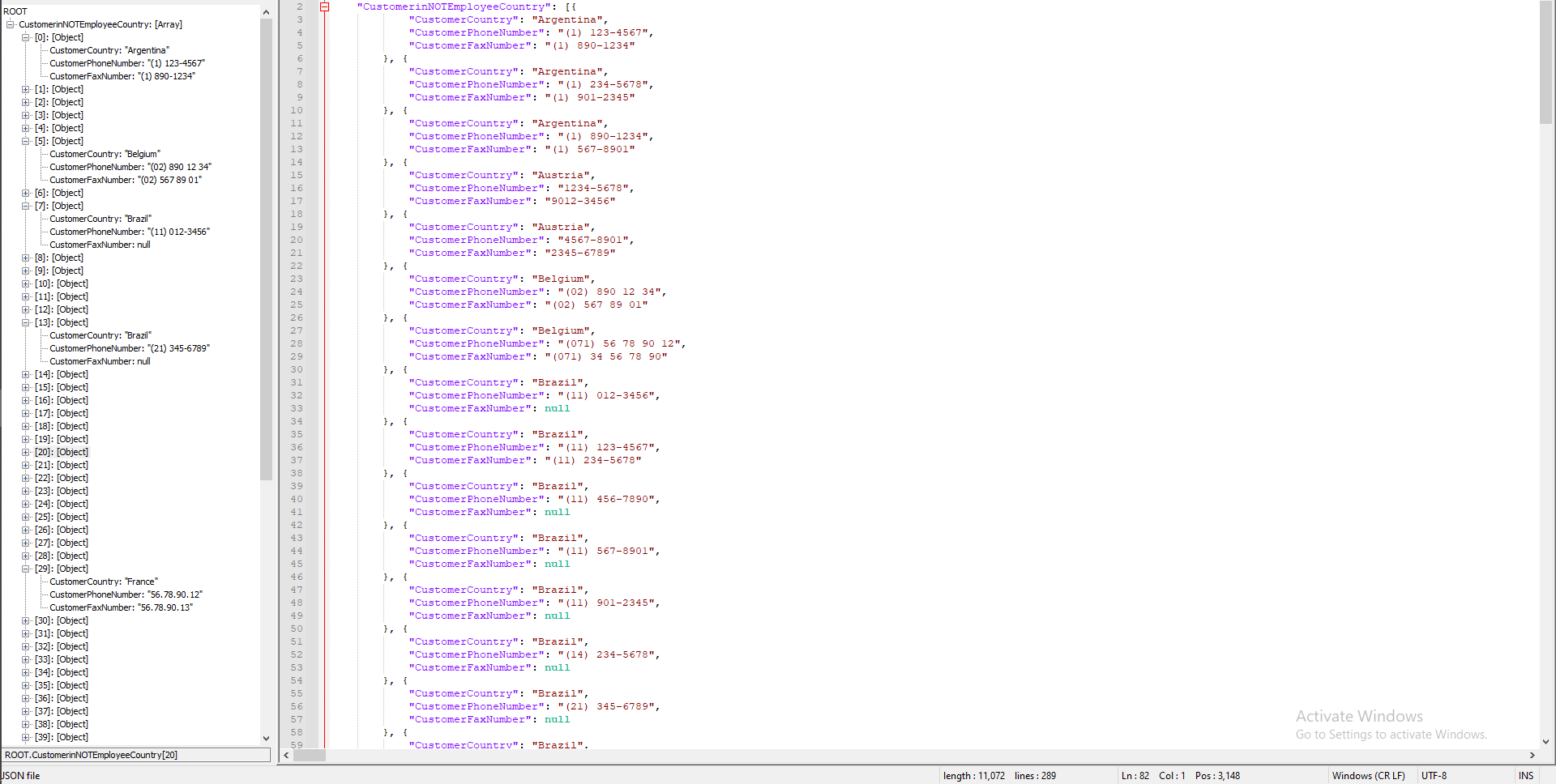
GROUP BY c.CustomerCountry

,c.CustomerPhoneNumber

,c.CustomerFaxNumber

ORDER BY c.CustomerCountry

FOR JSON PATH, root('CustomerinNOTEmployeeCountry'), include\_null\_values;



-----------------------------------------------------------------------------------------

## Proposition 13: How many orders did each customer make across 2 quarters in Q2 and Q3 in 2014 using NorthWinds2020TSQLV6?

## Detailed explanation of the problem

Find the OrderId, orderdate, customerid, employeeid as well as the suppliercompanyname and shiptocountry and the suppliercountry within the sales.order and production.supplier

## Diagram(s) of tables

## Standard view

## Select Clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | CustomerId |
| Customer | OrderId |
|  |  |
|  |  |
|  |  |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE NorthWinds2020TSQLV6

SELECT c.CustomerId

,COUNT(o.OrderId) AS OrderAmount

FROM Sales.Customer AS c

INNER JOIN Sales.[Order] AS o ON c.CustomerId = o.CustomerId

WHERE o.OrderDate BETWEEN '20140401'

AND '20140930'

GROUP BY c.CustomerId

## 

## Output with total number of rows returned (43)

## JSON Output with total number of rows returned (43)

USE NorthWinds2020TSQLV6

SELECT c.CustomerId

,COUNT(o.OrderId) AS OrderAmount

FROM Sales.Customer AS c

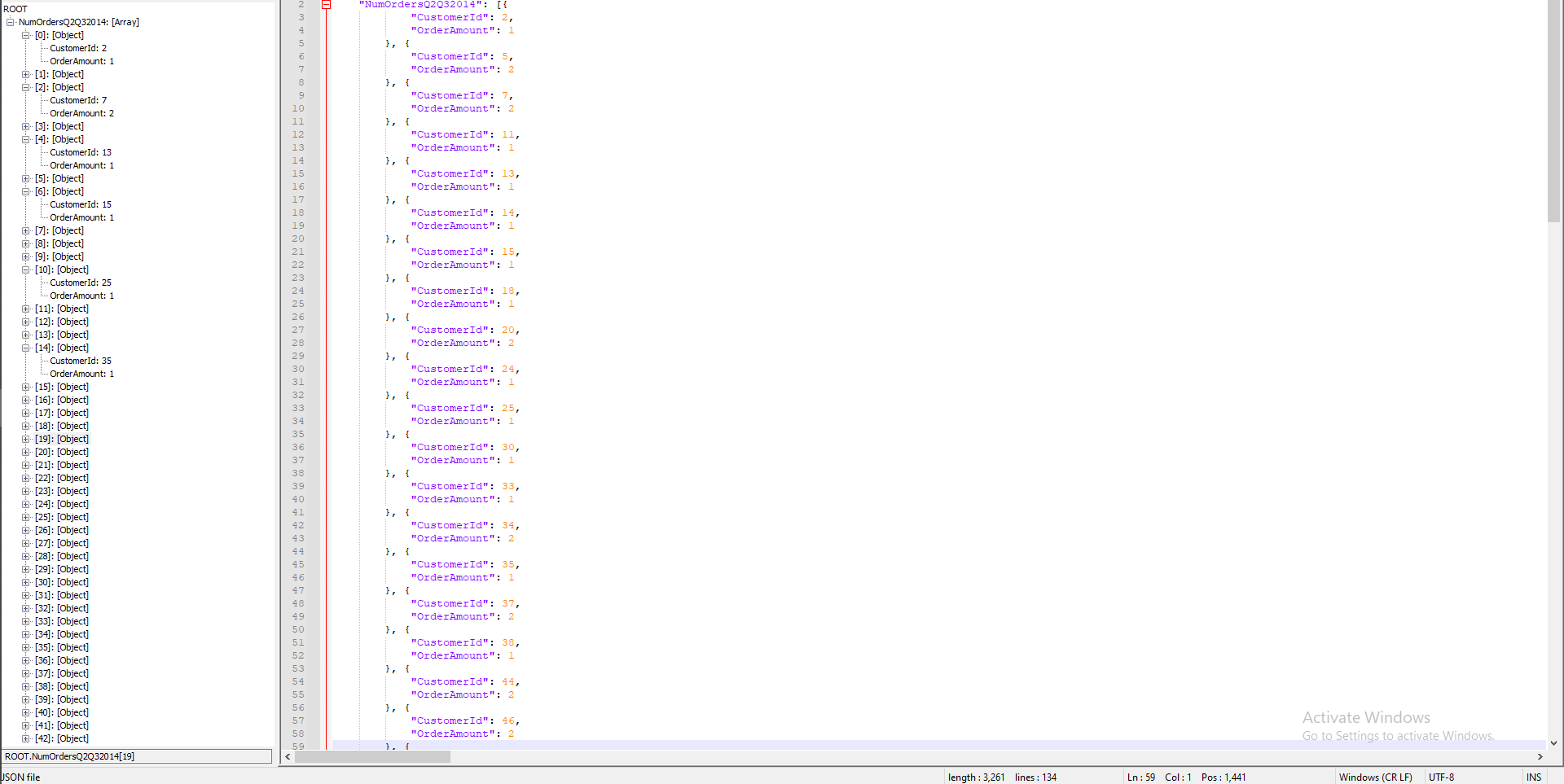
INNER JOIN Sales.[Order] AS o ON c.CustomerId = o.CustomerId

WHERE o.OrderDate BETWEEN '20140401'

AND '20140930'

GROUP BY c.CustomerId

FOR JSON PATH, root('NumOrdersQ2Q32014'), include\_null\_values;



-----------------------------------------------------------------------------------------

Proposition 14: What are the Regions in which there exists both a customer and a Employee, allows NULLS but must specify if the NULL is one or both Regions, Otherwise return if they are equal or not

## Detailed explanation of the problem

Find out the ids of the customer and the employee, as well as the regions that they are in, make a custom scalar function that outputs whether one of the regions are NULL as one is unknown or if both are null then output both are unknown, if neither are null then output if they are in the same region or else they are not in the same region

## 

## Diagram(s) of tables

## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerId  CustomerRegion |
| Employee | EmployeeId |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Customer | CustomerId | ASC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6;

DROP FUNCTION

IF EXISTS dbo.CustomerEmployeeRegion

GO

CREATE FUNCTION dbo.CustomerEmployeeRegion (

@customerRegion NVARCHAR(15)

,@employeeRegion NVARCHAR(15)

)

RETURNS NVARCHAR(80)

AS

BEGIN

IF (

@customerRegion IS NULL

AND @employeeRegion IS NULL

)

RETURN 'Both are unknown';

ELSE IF (

@customerRegion IS NULL

OR @employeeRegion IS NULL

)

RETURN 'One is unknown'

IF (@customerRegion = @employeeRegion)

RETURN 'Employee and Customer are in the same Region';

RETURN 'Employee and Customer are not in the same Region';

END;

GO

SELECT c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,dbo.CustomerEmployeeRegion(c.CustomerRegion, e.EmployeeRegion) AS [Location]

FROM Sales.Customer AS c

LEFT OUTER JOIN Sales.[Order] AS o ON c.CustomerId = o.CustomerId

LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId

LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId

GROUP BY c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,dbo.CustomerEmployeeRegion(c.CustomerRegion, e.EmployeeRegion)

ORDER BY c.CustomerId

## 

## Problem Relational Output with total number of rows returned (466)

## JSON Output with total number of rows returned (466)

use NorthWinds2020TSQLV6

SELECT c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,dbo.CustomerEmployeeRegion(c.CustomerRegion, e.EmployeeRegion) AS [Location]

FROM Sales.Customer AS c

LEFT OUTER JOIN Sales.[Order] AS o ON c.CustomerId = o.CustomerId

LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId

LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId

GROUP BY c.CustomerId

,e.EmployeeId

,c.CustomerRegion

,e.EmployeeRegion

,dbo.CustomerEmployeeRegion(c.CustomerRegion, e.EmployeeRegion)

ORDER BY c.CustomerId

FOR JSON PATH, root('EmployeeCustomerRegionEquivalences'), include\_null\_values;

-----------------------------------------------------------------------------------------

Proposition 15: There is a sugar shortage in your town and you must find out how to get sugar, identify which products are confections and which aren't and then which suppliers deliver those products and which do not ship confections, it is life or death (77 rows affected)

## Detailed explanation of the problem

Find out the suppliers that ship confections, and create a custom scalar function that checks whether or not the shipper is a confection shipper and/or if they are currently shipping

## 

## Diagram(s) of tables

## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| ProductInventory | LocationID  ProductID |
| ProductListPriceHistory | ListPrice |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| ProductListPriceHistory | ListPrice | DESC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

GO

DROP FUNCTION

IF EXISTS Production.Confection

GO

CREATE FUNCTION Production.Confection (

@SupplierName NVARCHAR(15)

,@Categoryid NVARCHAR(15)

)

RETURNS NVARCHAR(60)

AS

BEGIN

IF (

@Categoryid = 3

AND @SupplierName = 'Supplier BWGYE'

)

RETURN 'This is a confection product and confection supplier #1'

ELSE IF (

@Categoryid = 3

AND @SupplierName = 'Supplier ELCRN'

)

RETURN 'This is a confection product and confection supplier #3'

ELSE IF (

@Categoryid = 3

AND @SupplierName = 'Supplier FNUXM'

)

RETURN 'This is a confection product and confection supplier #4'

ELSE IF (

@Categoryid = 3

AND @SupplierName = 'Supplier GQRCV'

)

RETURN 'This is a confection product and confection supplier #5'

ELSE IF (

@Categoryid = 3

AND @SupplierName = 'Supplier OGLRK'

)

RETURN 'This is a confection product and confection supplier #6'

ELSE IF (

@Categoryid = 3

AND @SupplierName = 'Supplier ZPYVS'

)

RETURN 'This is a confection product and confection supplier #2'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier BWGYE'

)

RETURN 'This is NOT a confection product but is confection supplier #1'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier ELCRN'

)

RETURN 'This is NOT a confection product but is confection supplier #3'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier FNUXM'

)

RETURN 'This is NOT confection product but is confection supplier #4'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier GQRCV'

)

RETURN 'This is NOT confection product but is confection supplier #5'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier OGLRK'

)

RETURN 'This is NOT confection product but is confection supplier #6'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier ZPYVS'

)

RETURN 'This is NOT confection product but is confection supplier #2'

ELSE IF (

@Categoryid <> 3

AND @SupplierName <> 'Supplier BWGYE'

)

RETURN 'This is NOT a confection product nor a confection supplier'

ELSE IF (

@Categoryid <> 3

AND @SupplierName <> 'Supplier ELCRN'

)

RETURN 'This is NOT a confection product nor a confection supplier'

ELSE IF (

@Categoryid <> 3

AND @SupplierName <> 'Supplier FNUXM'

)

RETURN 'This is NOT a confection product nor a confection supplier'

ELSE IF (

@Categoryid <> 3

AND @SupplierName <> 'Supplier GQRCV'

)

RETURN 'This is NOT a confection product nor a confection supplier'

ELSE IF (

@Categoryid <> 3

AND @SupplierName <> 'Supplier OGLRK'

)

RETURN 'This is NOT a confection product nor a confection supplier'

ELSE IF (

@Categoryid <> 3

AND @SupplierName = 'Supplier ZPYVS'

)

RETURN 'This is NOT a confection product nor a confection supplier'

RETURN 'Done'

END;

GO

SELECT pc.CategoryId

,pc.CategoryName

,pp.ProductId

,ps.SupplierCompanyName

,Production.Confection(ps.SupplierCompanyName, pc.CategoryId) AS ConfectionSupplierInformation

FROM Production.Category AS pc

LEFT OUTER JOIN Production.Product AS pp ON pp.CategoryId = pc.CategoryId

LEFT OUTER JOIN Production.Supplier AS ps ON ps.SupplierId = pp.SupplierId

GROUP BY pc.CategoryId

,pc.CategoryName

,pp.ProductId

,ps.SupplierCompanyName

,Production.Confection(ps.SupplierCompanyName, pc.CategoryId)

ORDER BY ps.SupplierCompanyName

## Relational Output with total number of rows returned (77)

## 

## 

## 

## JSON Output with total number of rows returned (77)

USE Northwinds2020TSQLV6

SELECT pc.CategoryId

,pc.CategoryName

,pp.ProductId

,ps.SupplierCompanyName

,Production.Confection(ps.SupplierCompanyName, pc.CategoryId) AS ConfectionSupplierInformation

FROM Production.Category AS pc

LEFT OUTER JOIN Production.Product AS pp ON pp.CategoryId = pc.CategoryId

LEFT OUTER JOIN Production.Supplier AS ps ON ps.SupplierId = pp.SupplierId

GROUP BY pc.CategoryId

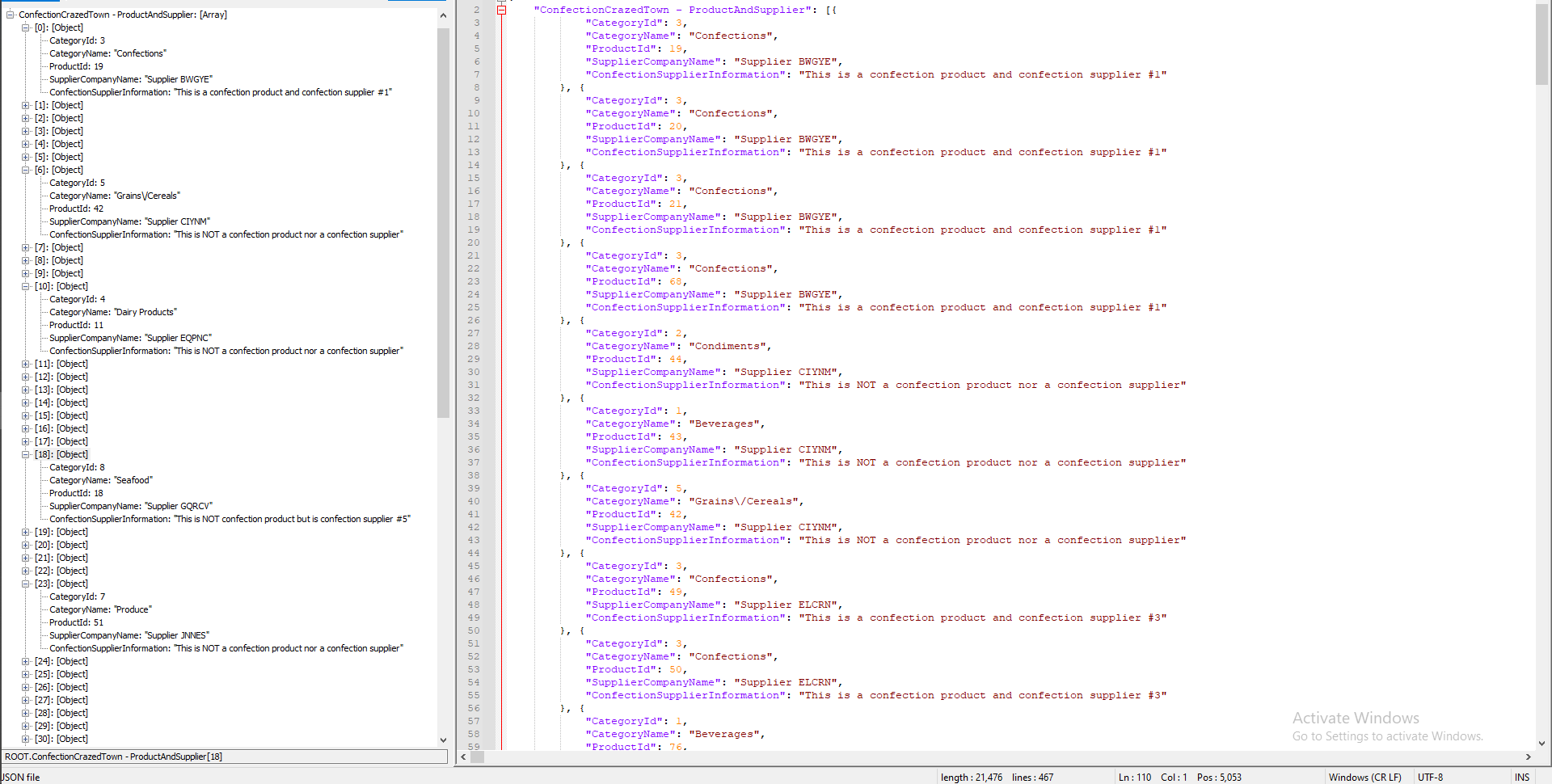
,pc.CategoryName

,pp.ProductId

,ps.SupplierCompanyName

,Production.Confection(ps.SupplierCompanyName, pc.CategoryId)

ORDER BY ps.SupplierCompanyName

FOR JSON PATH, root('ProductsInformation'), include\_null\_values;

-----------------------------------------------------------------------------------------

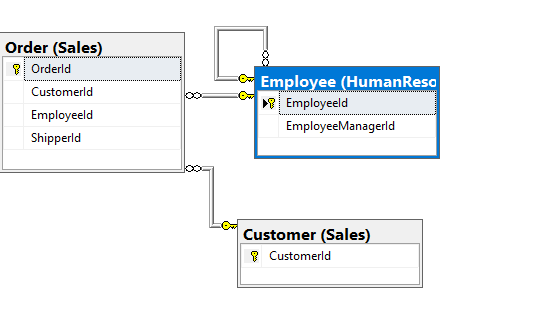
Proposition 16:Find all of the orders ordered by CustomerId and show the orderid included with the information of the employee as well as if they are male or female or not sure

## Detailed explanation of the problem

Find all the orderId’s and customerid’s along with employees first name and employees full name and their titleofcourtesy and then create a custom scalar function that compares the title of courtesy to common known male/female pronouns and for non-gender specific Titles of courtesy.

## 

## Diagram(s) of tables



## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId |
| Customer | CustomerId |
| Employee | EmployeeFirstName  EmployeeLastName  EmployeeTitleOfCourtesy |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Customer | CustomerId | ASC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

**USE Northwinds2020TSQLV6**

**DROP FUNCTION**

**IF EXISTS dbo.CheckGender**

**GO**

**CREATE FUNCTION dbo.CheckGender (@TitleofCourtesy NVARCHAR(5))**

**RETURNS NVARCHAR(15)**

**AS**

**BEGIN**

**IF (@TitleofCourtesy = 'Mr.')**

**RETURN 'Male'**

**IF (**

**@TitleofCourtesy = 'Mrs.'**

**OR @TitleofCourtesy = 'Ms.'**

**)**

**RETURN 'Female'**

**IF (@TitleofCourtesy = 'Dr.')**

**RETURN 'Not sure'**

**RETURN 'Something is wrong I can feel it'**

**END**

**GO**

**SELECT hr.EmployeeId**

**,sc.CustomerId**

**,so.OrderId**

**,hr.EmployeeFirstName**

**,hr.EmployeeLastName**

**,hr.EmployeeTitleOfCourtesy**

**,dbo.CheckGender(hr.EmployeeTitleOfCourtesy) AS Gender**

**FROM Sales.[Order] AS so**

**INNER JOIN [HumanResources].[Employee] AS hr ON so.EmployeeId = so.EmployeeId**

**INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId**

**GROUP BY hr.EmployeeId**

**,sc.CustomerId**

**,so.OrderId**

**,hr.EmployeeFirstName**

**,hr.EmployeeLastName**

**,hr.EmployeeTitleOfCourtesy**

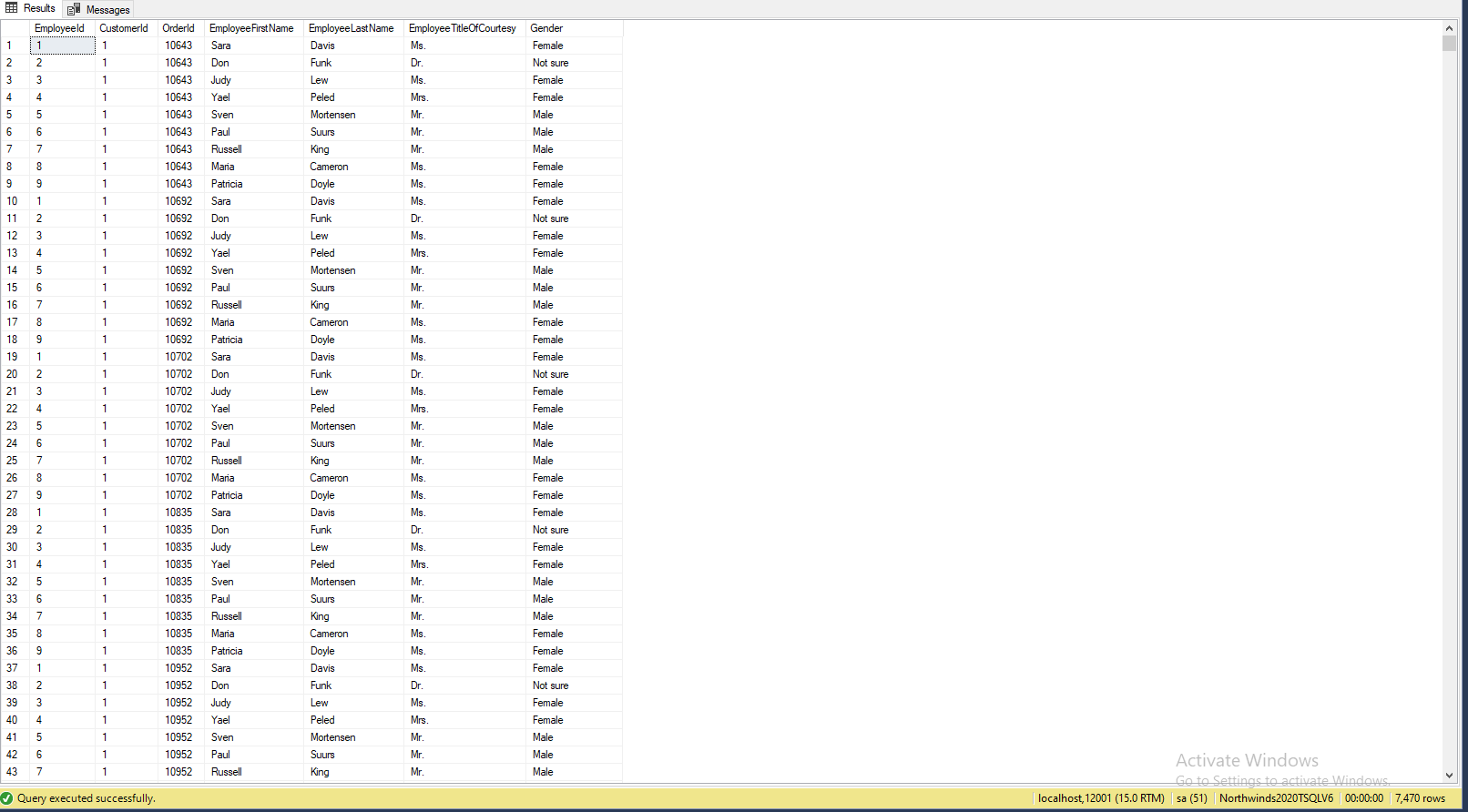
**,dbo.CheckGender(hr.EmployeeTitleOfCourtesy)**

**ORDER BY sc.CustomerId**

**FOR JSON PATH,root('GenderChecker'),include\_null\_values;**

## 

## Relational Output with total number of rows returned (7470)



## JSON Output with total number of rows returned (7470)

**SELECT hr.EmployeeId**

**,sc.CustomerId**

**,so.OrderId**

**,hr.EmployeeFirstName**

**,hr.EmployeeLastName**

**,hr.EmployeeTitleOfCourtesy**

**,dbo.CheckGender(hr.EmployeeTitleOfCourtesy) AS Gender**

**FROM Sales.[Order] AS so**

**INNER JOIN [HumanResources].[Employee] AS hr ON so.EmployeeId = so.EmployeeId**

**INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId**

**GROUP BY hr.EmployeeId**

**,sc.CustomerId**

**,so.OrderId**

**,hr.EmployeeFirstName**

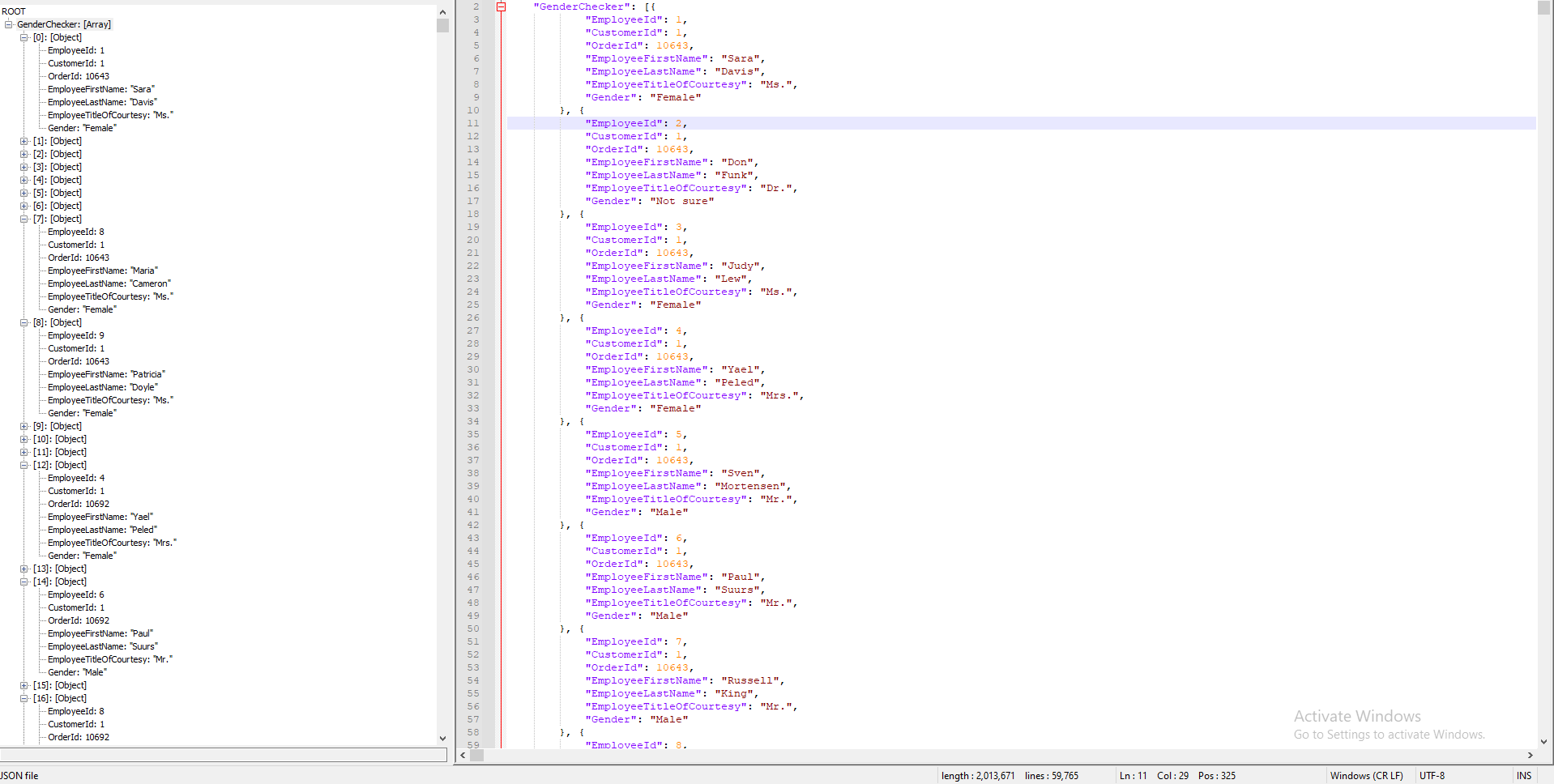
**,hr.EmployeeLastName**

**,hr.EmployeeTitleOfCourtesy**

**,dbo.CheckGender(hr.EmployeeTitleOfCourtesy)**

**ORDER BY sc.CustomerId**

**FOR JSON PATH,root('GenderChecker'),include\_null\_values;**



-----------------------------------------------------------------------------------------

Proposition 17: Where is the country located in which we have to shipto and the order date and orderid of the company that ordered within the entire year of 2016

## Detailed explanation of the problem

Find the orderid, customercompany name and the orderdate and which country it will be shipped to and then create a custom scalar function that identifies the part of the continent in which the country being shipped to is located in

## 

## Diagram(s) of tables

## Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderDate  ShipToCountry |
| Customer | CustomerCompanyName |
|  |  |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Order | OrderDate | ASC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS dbo.CountryLocationChecker

GO

CREATE FUNCTION dbo.CountryLocationChecker (@ShipCountry NVARCHAR(20))

RETURNS NVARCHAR(50)

AS

BEGIN

IF (

@ShipCountry = 'Canada'

OR @ShipCountry = 'USA'

OR @ShipCountry = 'Mexico'

)

RETURN 'Country is in North America'

ELSE IF (

@ShipCountry = 'Argentina'

OR @ShipCountry = 'Brazil'

OR @ShipCountry = 'Venezuela'

)

RETURN 'Country is in South America'

ELSE IF (

@ShipCountry = 'Austria'

OR @ShipCountry = 'Belgium'

OR @ShipCountry = 'Denmark'

OR @ShipCountry = 'Finland'

OR @ShipCountry = 'France'

OR @ShipCountry = 'Germany'

OR @ShipCountry = 'Ireland'

OR @ShipCountry = 'Italy'

OR @ShipCountry = 'Norway'

OR @ShipCountry = 'Portugal'

OR @ShipCountry = 'Spain'

OR @ShipCountry = 'Sweden'

OR @ShipCountry = 'UK'

OR @ShipCountry = 'Switzerland'

OR @ShipCountry = 'Poland'

)

RETURN 'Country is in Europe'

RETURN 'Country does not exist'

END

GO

SELECT so.OrderId

,sc.CustomerCompanyName

,so.OrderDate

,so.ShipToCountry

,dbo.CountryLocationChecker(so.ShipToCountry) AS CountryLocated

FROM Sales.[Order] AS so

INNER JOIN [HumanResources].[Employee] AS hr ON so.EmployeeId = so.EmployeeId

INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId

WHERE YEAR(so.OrderDate) = '2016'

GROUP BY so.OrderId

,sc.CustomerCompanyName

,so.OrderDate

,so.ShipToCountry

,dbo.CountryLocationChecker(so.ShipToCountry)

ORDER BY so.OrderDate

## Relational Output with total number of rows returned (270)

## JSON Output with total number of rows returned (270)

SELECT so.OrderId

,sc.CustomerCompanyName

,so.OrderDate

,so.ShipToCountry

,dbo.CountryLocationChecker(so.ShipToCountry) AS CountryLocated

FROM Sales.[Order] AS so

INNER JOIN [HumanResources].[Employee] AS hr ON so.EmployeeId = so.EmployeeId

INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId

WHERE YEAR(so.OrderDate) = '2016'

GROUP BY so.OrderId

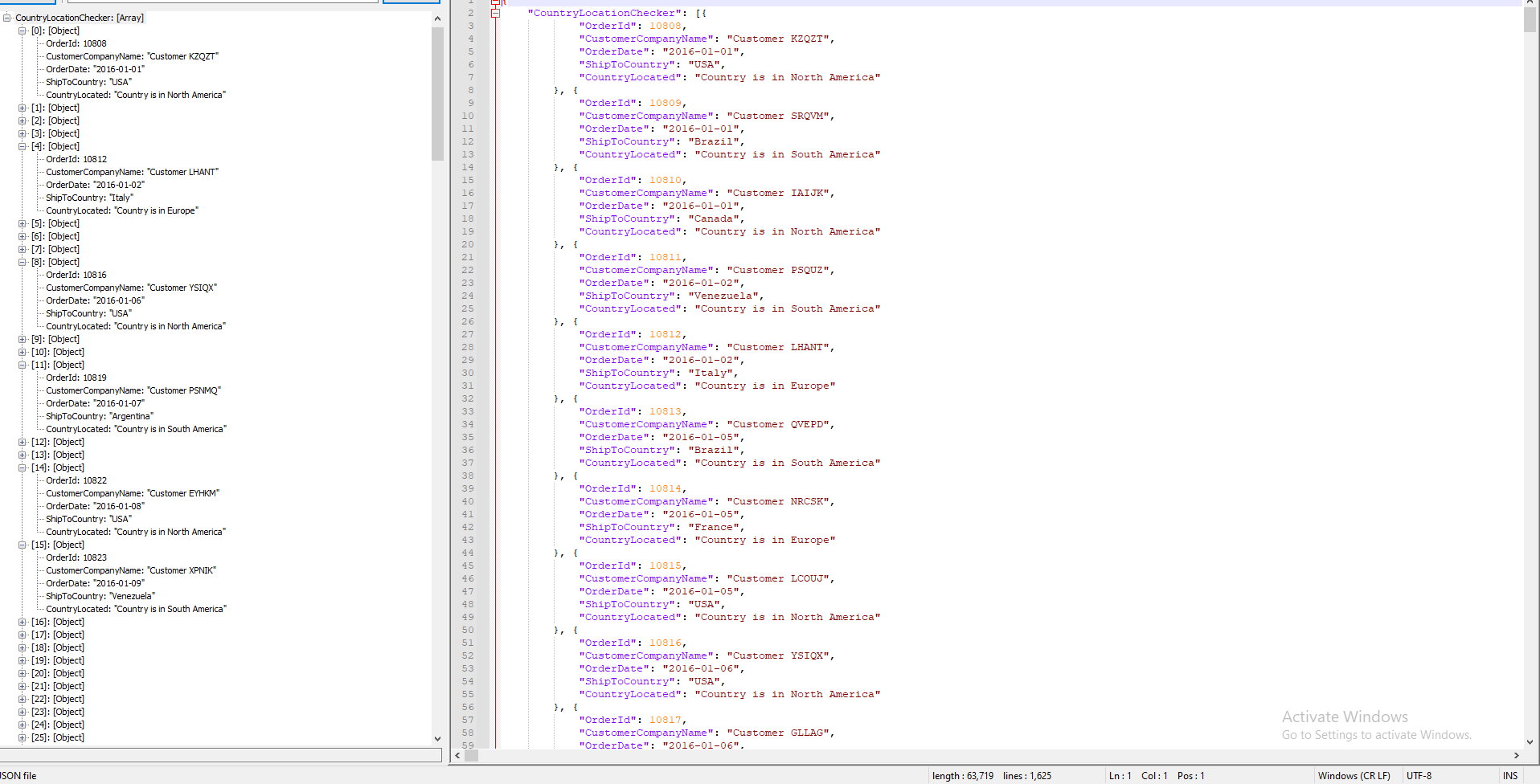
,sc.CustomerCompanyName

,so.OrderDate

,so.ShipToCountry

,dbo.CountryLocationChecker(so.ShipToCountry)

ORDER BY so.OrderDate

FOR JSON PATH, root('CountryLocationChecker'), include\_null\_values;

-----------------------------------------------------------------------------------------

Proposition 18: Find the number of orders made in Decembers by the HighestLevelImportance in order by the orderdate using NorthWinds2020TSQLV6(15 rows affected)

## Detailed explanation of the problem

Find the number of orders made in december and which of those orders were made by the HighestLevelImportance people when you create a scalar function of Lower level of importance, to medium and then to highest

## 

## Diagram(s) of tables

## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId  OrderDate  EmployeeId |
| Customer | CustomerContactTitle |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Order | OrderDate | ASC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS dbo.CustomerTitleImportance

GO

CREATE FUNCTION dbo.CustomerTitleImportance (

@CustomerContactTitle NVARCHAR(30))

RETURNS NVARCHAR(40)

AS

BEGIN

IF (

@CustomerContactTitle = 'Owner'

OR @CustomerContactTitle = 'Order Administrator'

)

RETURN 'Highest Level Importance'

IF (

@CustomerContactTitle = 'Accounting Manager'

OR @CustomerContactTitle = 'Marketing Manager'

OR @CustomerContactTitle = 'Sales Manager'

)

RETURN 'Intermediate Level Importance'

IF (

@CustomerContactTitle = 'Sales Associate'

OR @CustomerContactTitle = 'Assistant Sales Agent'

OR @CustomerContactTitle = 'Marketing Assistant'

OR @CustomerContactTitle = 'Assistant Sales Representative'

OR @CustomerContactTitle = 'Sales Representative'

OR @CustomerContactTitle = 'Sales Agent'

)

RETURN 'Lower Level Importance'

RETURN 'Mixed Importance'

END

GO

SELECT so.OrderId

,so.OrderDate

,so.EmployeeId

,sc.CustomerContactTitle

,dbo.CustomerTitleImportance(sc.CustomerContactTitle) AS 'Relative Importance Level'

FROM Sales.[Order] AS so

INNER JOIN [HumanResources].[Employee] AS hr ON hr.EmployeeId = so.EmployeeId

INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId

WHERE dbo.CustomerTitleImportance(sc.CustomerContactTitle) = 'Highest Level Importance'

AND MONTH(so.OrderDate) = 12

GROUP BY so.OrderId

,so.OrderDate

,so.EmployeeId

,sc.CustomerContactTitle

,dbo.CustomerTitleImportance(sc.CustomerContactTitle)

ORDER BY so.OrderDate

## 

## Output with total number of rows returned (15)

## JSON Output with total number of rows returned (15)

SELECT so.OrderId

,so.OrderDate

,so.EmployeeId

,sc.CustomerContactTitle

,dbo.CustomerTitleImportance(sc.CustomerContactTitle) AS 'Relative Importance Level'

FROM Sales.[Order] AS so

INNER JOIN [HumanResources].[Employee] AS hr ON hr.EmployeeId = so.EmployeeId

INNER JOIN Sales.Customer AS sc ON sc.CustomerId = so.CustomerId

WHERE dbo.CustomerTitleImportance(sc.CustomerContactTitle) = 'Highest Level Importance'

AND MONTH(so.OrderDate) = 12

GROUP BY so.OrderId

,so.OrderDate

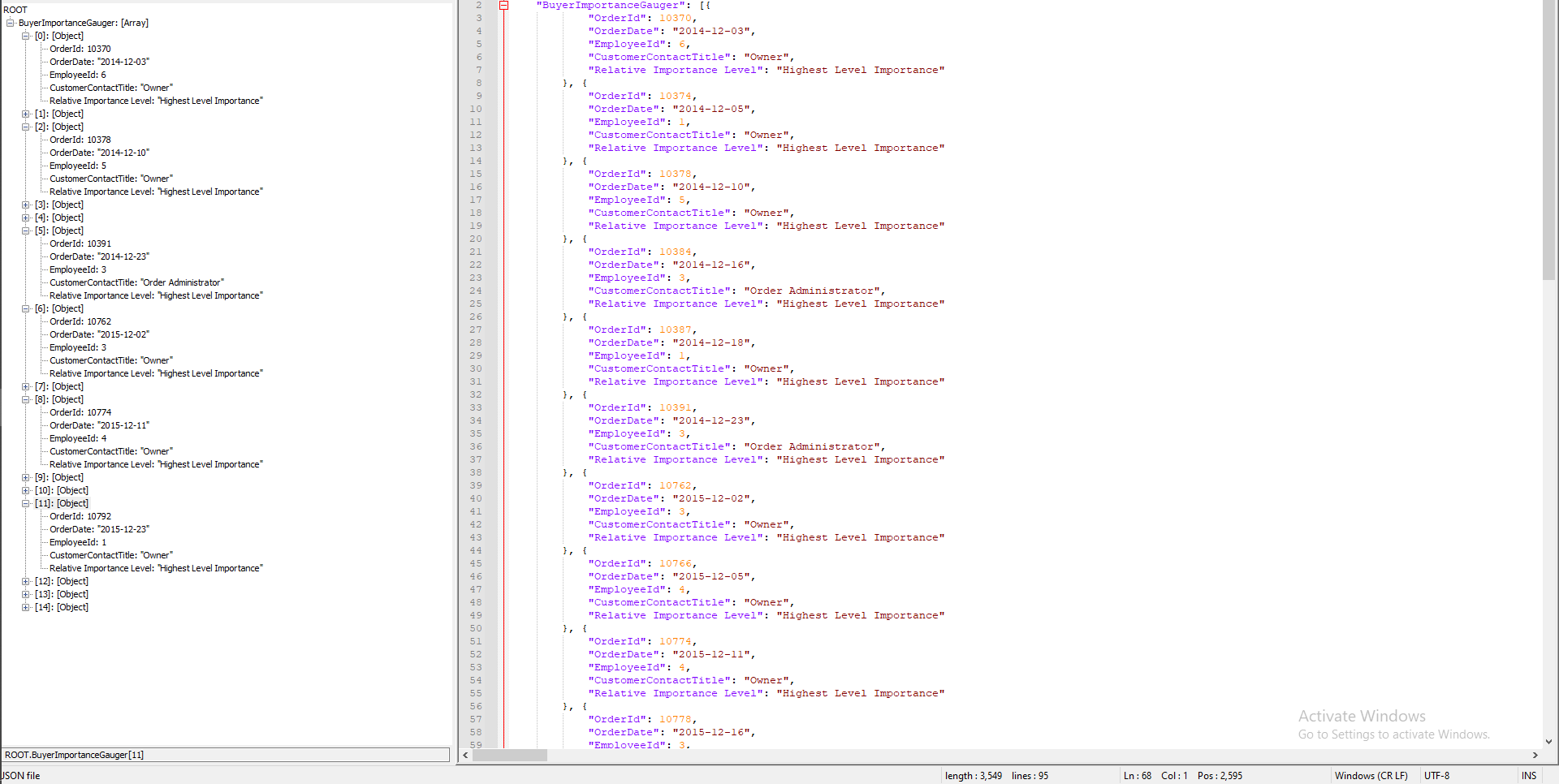
,so.EmployeeId

,sc.CustomerContactTitle

,dbo.CustomerTitleImportance(sc.CustomerContactTitle)

ORDER BY so.OrderDate

FOR JSON PATH, root('BuyerImportanceGauger'), include\_null\_values;



-----------------------------------------------------------------------------------------

Proposition 19: Return the set of letters that the first name and the last name fall into as well as the employeesID and their managersID

## Detailed explanation of the problem

Create a scalar function that processes the first letter of an inputted name and outputs it as an alphabetical range of the name, that has to work with both first and last name of the employee, then as a non scalar function their manager’s id and their own employeeid, the managerid is null if they do not have a manager above them

## 

## Diagram(s) of tables

## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Employees | EmployeeManagerId  EmployeeId |
| Order | OrderId |

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS dbo.AlphabetNameIdentifier

GO

CREATE FUNCTION dbo.AlphabetNameIdentifier (@Names NVARCHAR(20))

RETURNS NVARCHAR(40)

AS

BEGIN

IF (@Names LIKE '[ABCDEF]%')

RETURN 'A to E'

IF (@Names LIKE '[GHJIJKL]%')

RETURN 'F to L'

IF (@Names LIKE '[MNOPQR]%')

RETURN 'M to S'

IF (@Names LIKE '[STUVWXYZ]%')

RETURN 'G to Z'

RETURN 'Wrong'

END

GO

SELECT DISTINCT e.EmployeeManagerId

,e.EmployeeId

,dbo.AlphabetNameIdentifier(e.EmployeeFirstName) AS FirstName

,dbo.AlphabetNameIdentifier(e.EmployeeLastName) AS LastName

FROM Sales.[Order] AS o

LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId

LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId

GROUP BY e.EmployeeManagerId

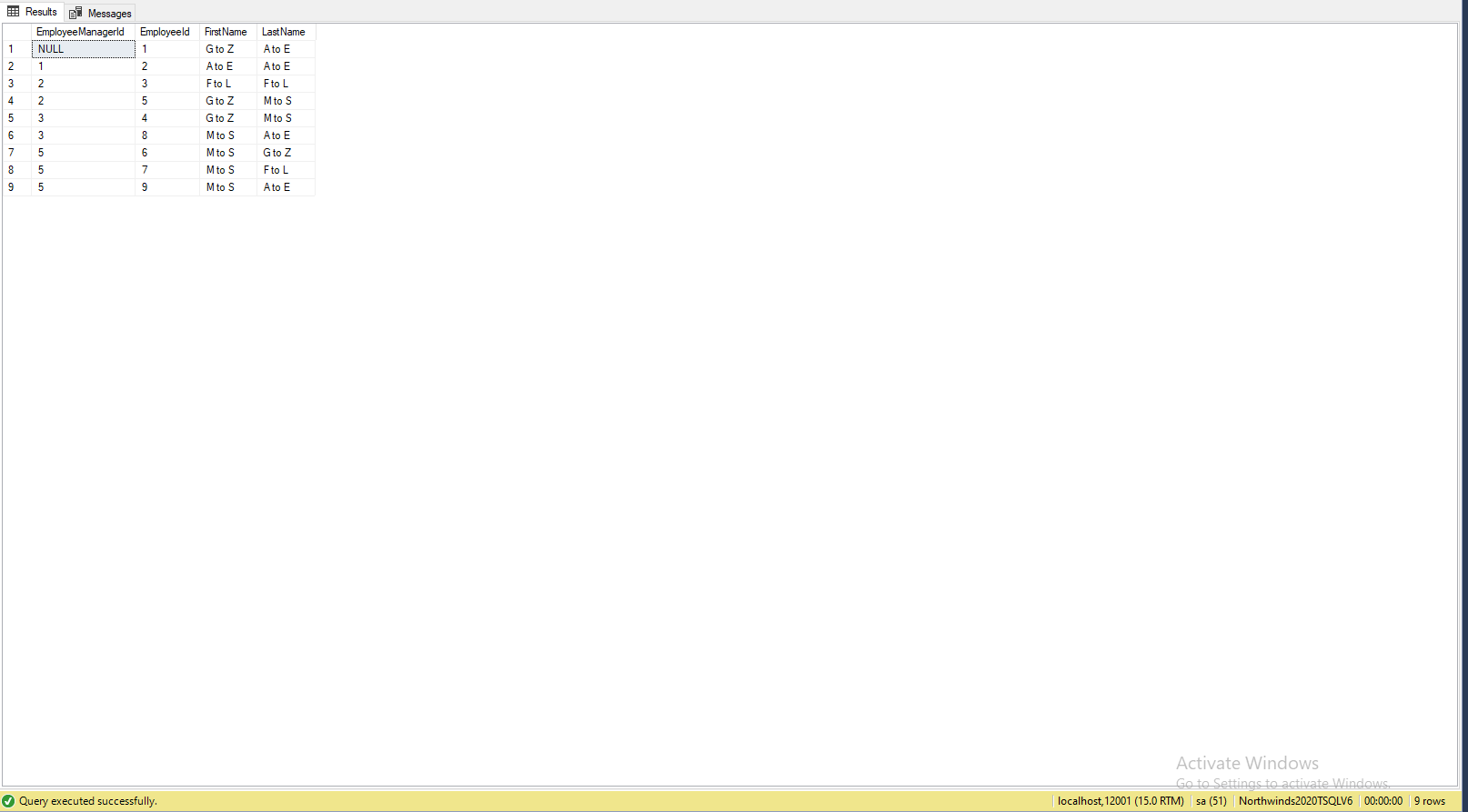
,e.EmployeeId

,o.OrderId

,dbo.AlphabetNameIdentifier(e.EmployeeFirstName)

,dbo.AlphabetNameIdentifier(e.EmployeeLastName)

## Output with total number of rows returned (830)



## JSON Output with total number of rows returned (830)

**SELECT DISTINCT e.EmployeeManagerId**

**,e.EmployeeId**

**,dbo.AlphabetNameIdentifier(e.EmployeeFirstName) AS FirstName**

**,dbo.AlphabetNameIdentifier(e.EmployeeLastName) AS LastName**

**FROM Sales.[Order] AS o**

**LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId**

**LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId**

**GROUP BY e.EmployeeManagerId**

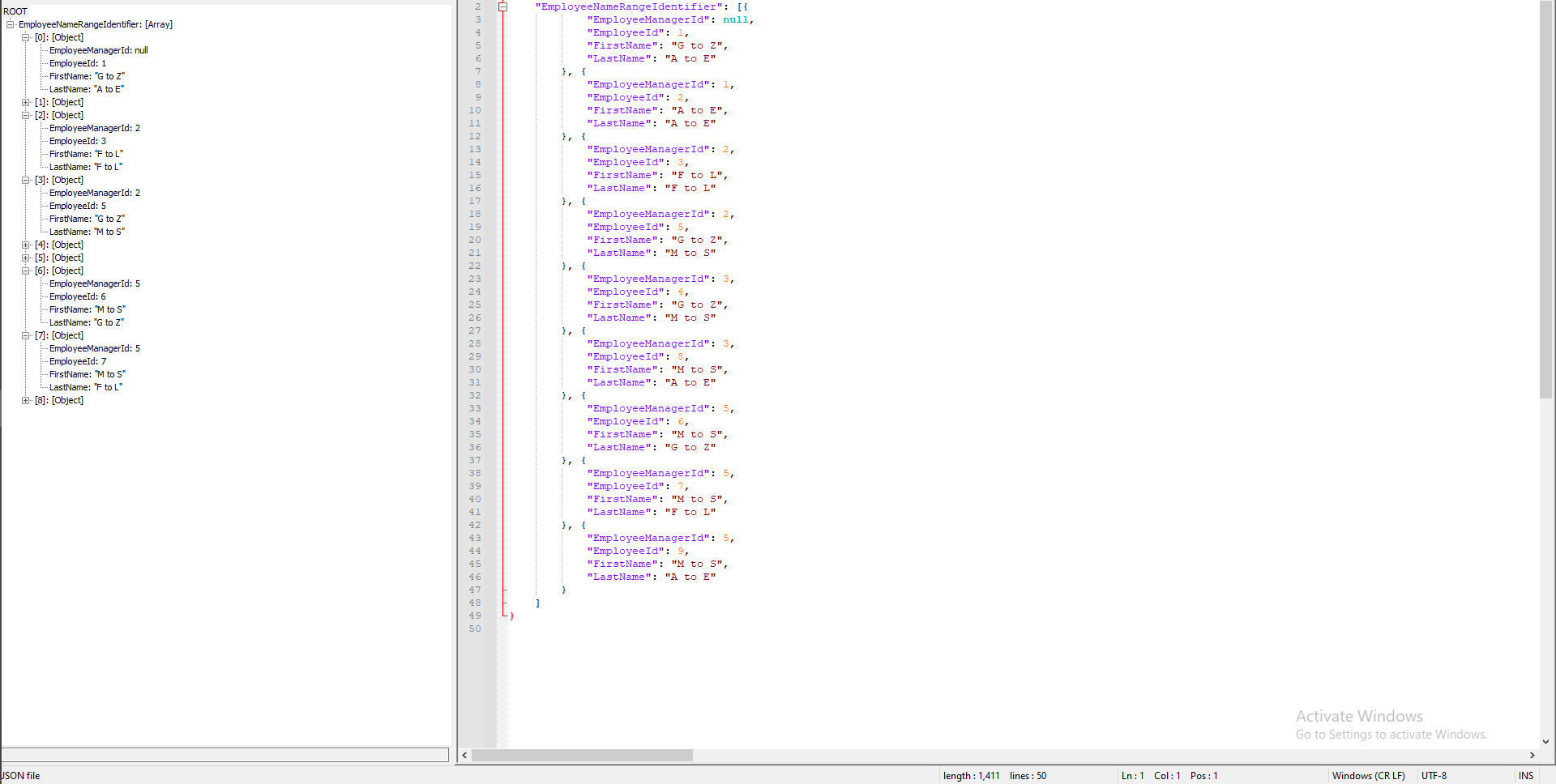
**,e.EmployeeId**

**,o.OrderId**

**,dbo.AlphabetNameIdentifier(e.EmployeeFirstName)**

**,dbo.AlphabetNameIdentifier(e.EmployeeLastName)**

**FOR JSON PATH, root('EmployeeNameRangeIdentifier'), include\_null\_values;**



-----------------------------------------------------------------------------------------

Proposition 20: Return whether or not the first character of each address is numerical or alphabetical and then break the numerical into 2 halfs 1 and 2 from 0 to 9 and then break Alphabetical into 4 quarters for A to Z respectively?

## Detailed explanation of the problem

Create a scalar function that checks the first letter or number of an address and then categorizes it a a numerical address and alphabetical address, numerical is split between 0 to 5 and 6 to 9, while the alphabetical address is split into 4, A-F is section1, G-L is section2, M-R sections3, and then S-Z is the 4th section. find the shipping to address the employee id and use the scalar function to create a column that specifies what kind of address the presented one is

## 

## Diagram(s) of tables

## Columns from Standard view

## Project following columns from their respective tables in the select clause

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | ShipToAddress |
| Employee | EmployeeId |

## Order by

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Order | ShipToAddress | ASC |

## 

## Problem solving Query

**All queries must use the ANSI 92 standard for queries with the type safe “on”.**

USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS dbo.AddressIdentifier

GO

CREATE FUNCTION dbo.AddressIdentifier (@ShipAddress NVARCHAR(20))

RETURNS NVARCHAR(40)

AS

BEGIN

IF (@ShipAddress LIKE '[0-5]%')

RETURN 'Numerical Address Section 1'

IF (@ShipAddress LIKE '[6-9]%')

RETURN 'Numerical Address Section 2'

IF (@ShipAddress LIKE '[A-F]%')

RETURN 'Alphabetical Address Section 1'

IF (@ShipAddress LIKE '[G-L]%')

RETURN 'Alphabetical Address Section 2'

IF (@ShipAddress LIKE '[M-R]%')

RETURN 'Alphabetical Address Section 3'

IF (@ShipAddress LIKE '[S-Z]%')

RETURN 'Alphabetical Address Section 4'

RETURN 'Out of Bounds'

END

GO

SELECT DISTINCT o.ShipToAddress

,e.EmployeeId

,dbo.AddressIdentifier(o.ShipToAddress) AS AddressSpecification

FROM Sales.[Order] AS o

LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId

LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId

GROUP BY o.ShipToAddress

,e.EmployeeId

,dbo.AddressIdentifier(o.ShipToAddress)

ORDER BY o.ShipToAddress

## 

## Relational Output with total number of rows returned (665)

## JSON Output with total number of rows returned (665)

**SELECT DISTINCT o.ShipToAddress**

**,e.EmployeeId**

**,dbo.AddressIdentifier(o.ShipToAddress) AS AddressSpecification**

**FROM Sales.[Order] AS o**

**LEFT OUTER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId**

**LEFT OUTER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId**

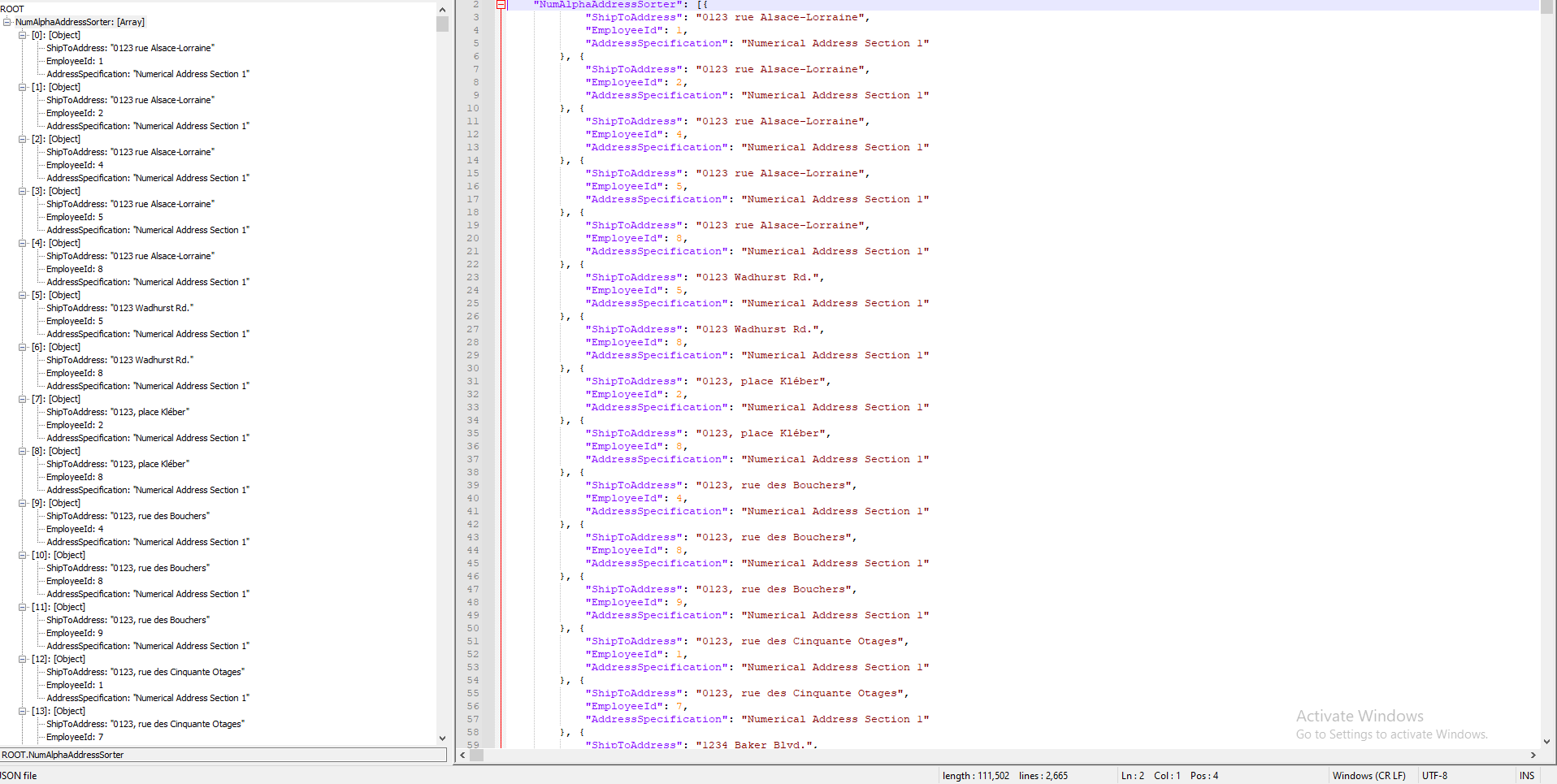
**GROUP BY o.ShipToAddress**

**,e.EmployeeId**

**,dbo.AddressIdentifier(o.ShipToAddress)**

**ORDER BY o.ShipToAddress**

**FOR JSON PATH,root('NumAlphaAddressSorter'),include\_null\_values;**

****