**CSCI 331**

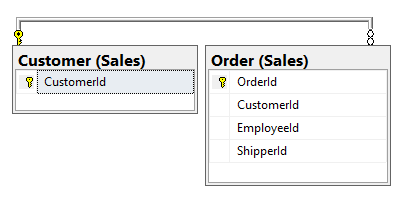
**PROJECT 1**

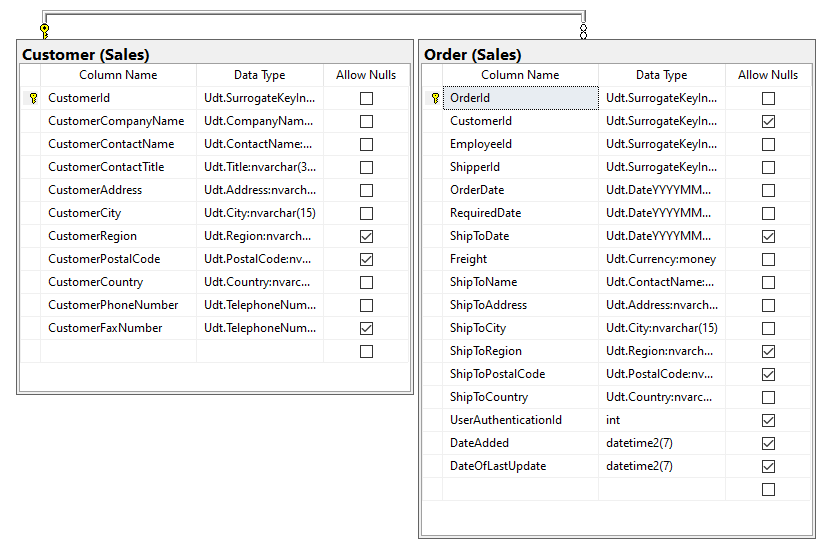
**Jonathan Eng**

Fall 2020

**DATABASES - PROFESSOR HELLER - 10:45-12:00 TUES/THURS**

Proposition #1 (Simple)

Show the Orders that were made in 2016 but have no ship date   




**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerId |
| Order | OrderId; Orderdate; ShipToDate |

**Order By**

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

**Solution Query**

USE Northwinds2020TSQLV6;

SELECT C.CustomerId, O.orderid, O.orderdate, O.ShipToDate

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

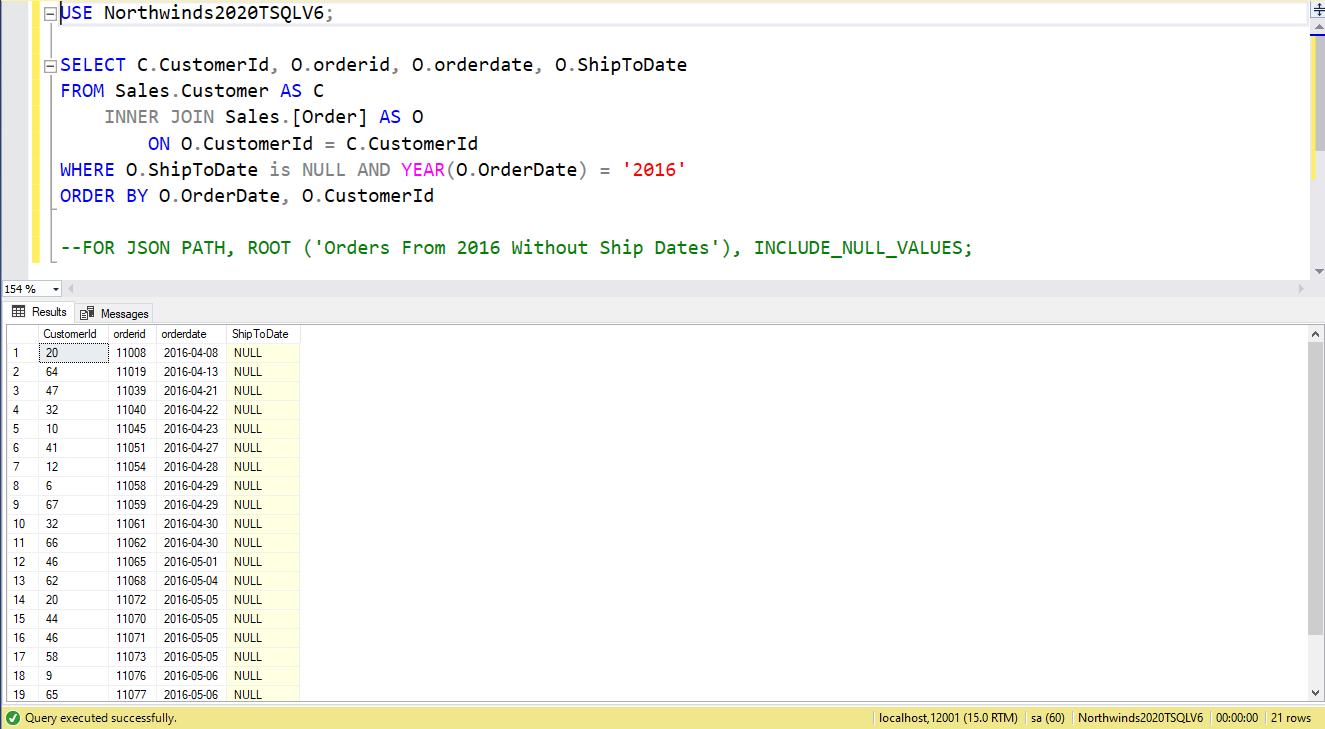
WHERE O.ShipToDate is NULL AND YEAR(O.OrderDate) = '2016'

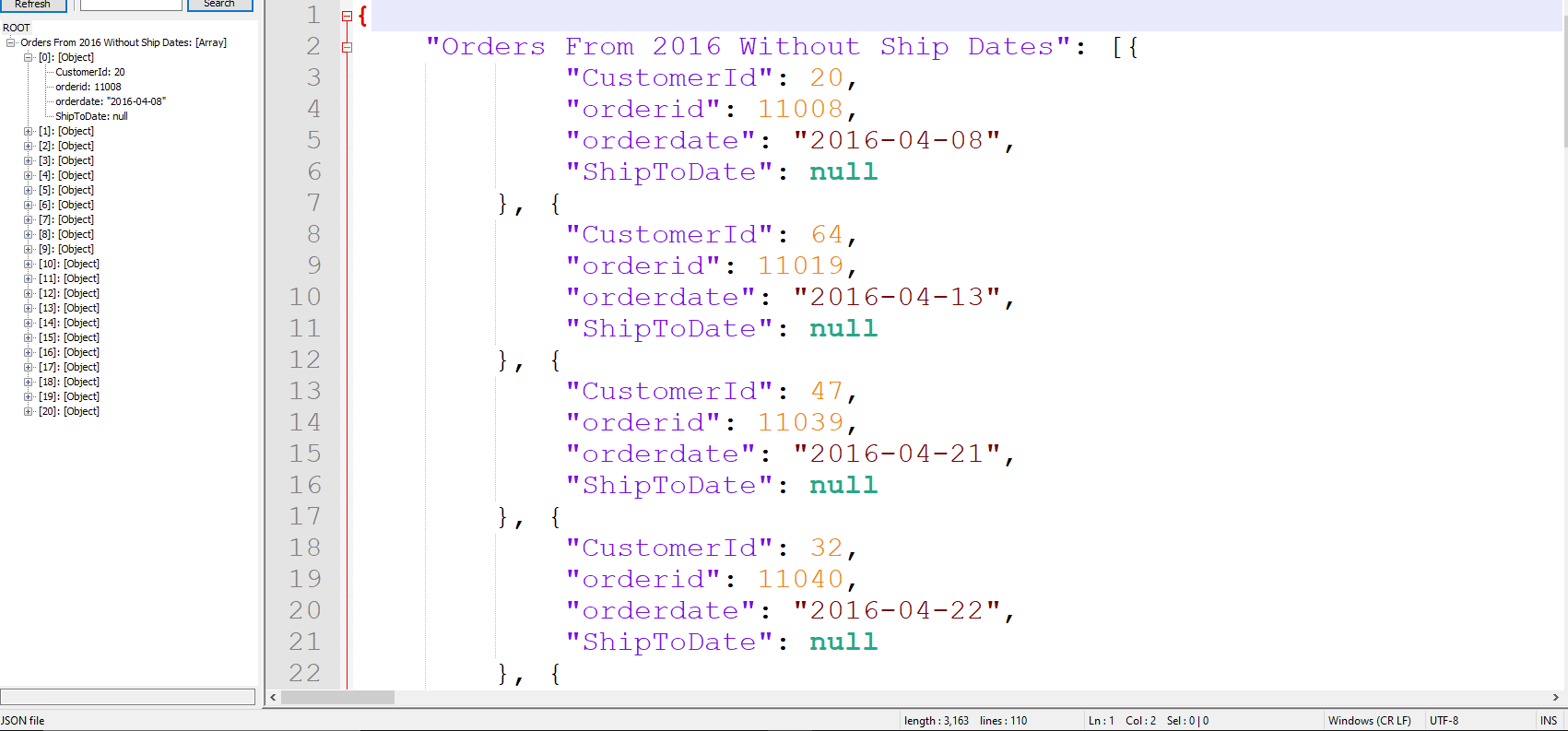
ORDER BY O.OrderDate, O.CustomerId

--Uncomment Below to get JSON output

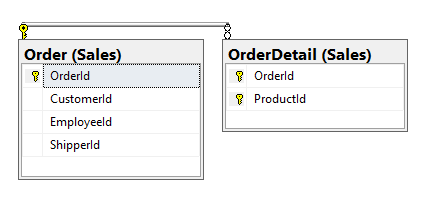
--FOR JSON PATH, ROOT ('Orders From 2016 Without Ship Dates'), INCLUDE\_NULL\_VALUES;

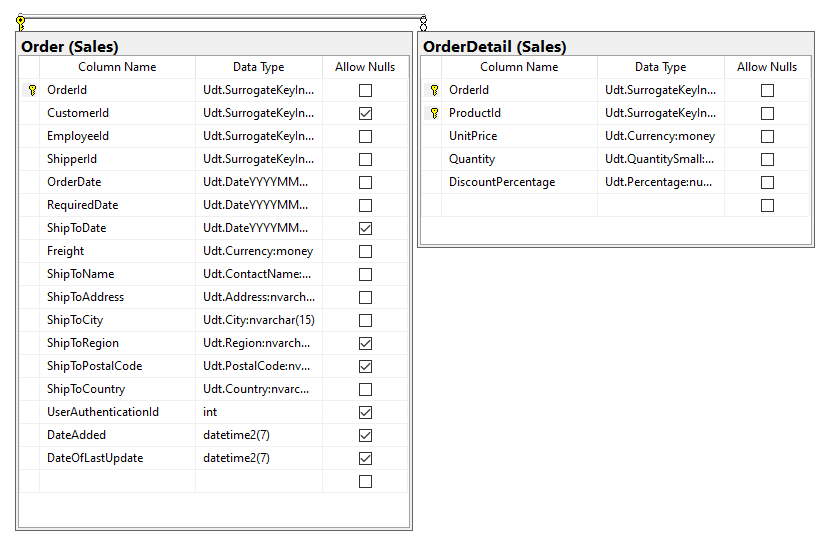
**Output**





Proposition #2 (Simple)

Show Customer's Orders along with the Discount given if it has one 



**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | CustomerId |
| OrderDetail | OrderId, DiscountPercentage |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| OrderDetail | DiscountPercentage | DESC |

USE Northwinds2020TSQLV6;

SELECT OD.OrderId, O.CustomerId, OD.DiscountPercentage

FROM Sales.[Order] AS O

LEFT OUTER JOIN Sales.[OrderDetail] AS OD

ON OD.OrderId = O.OrderId

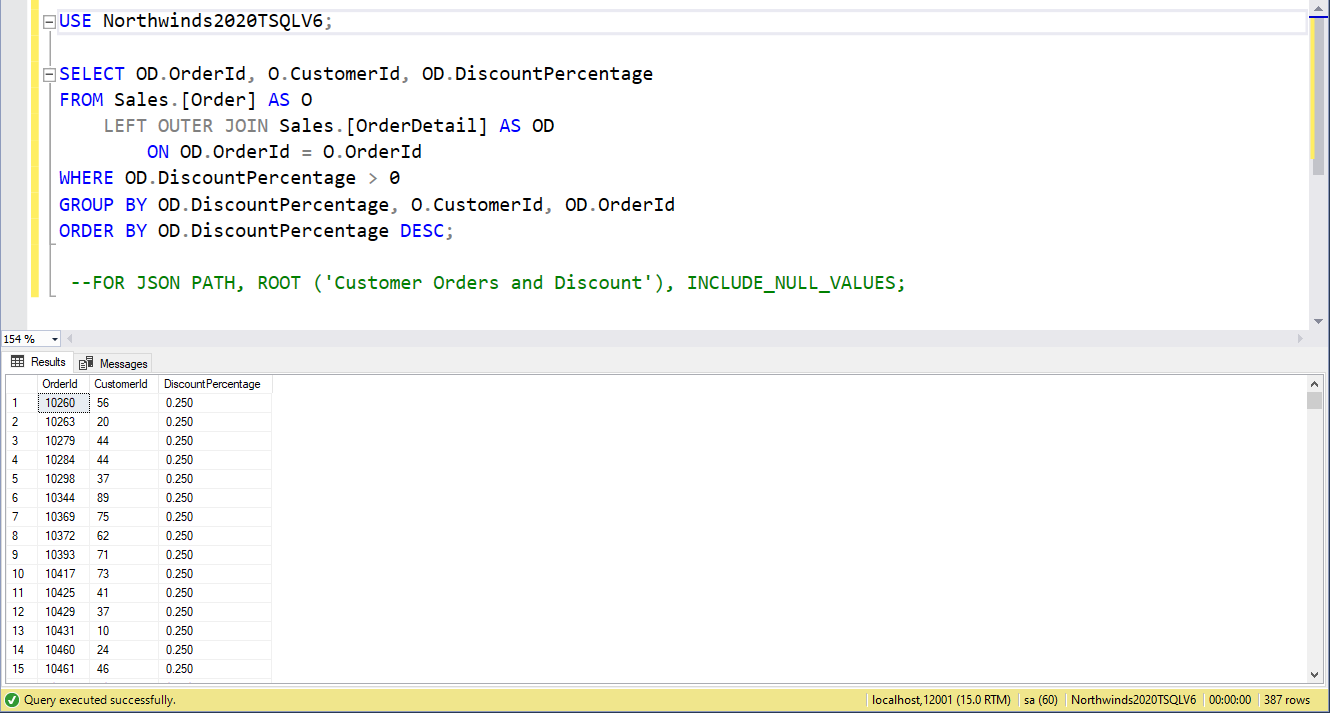
WHERE OD.DiscountPercentage > 0

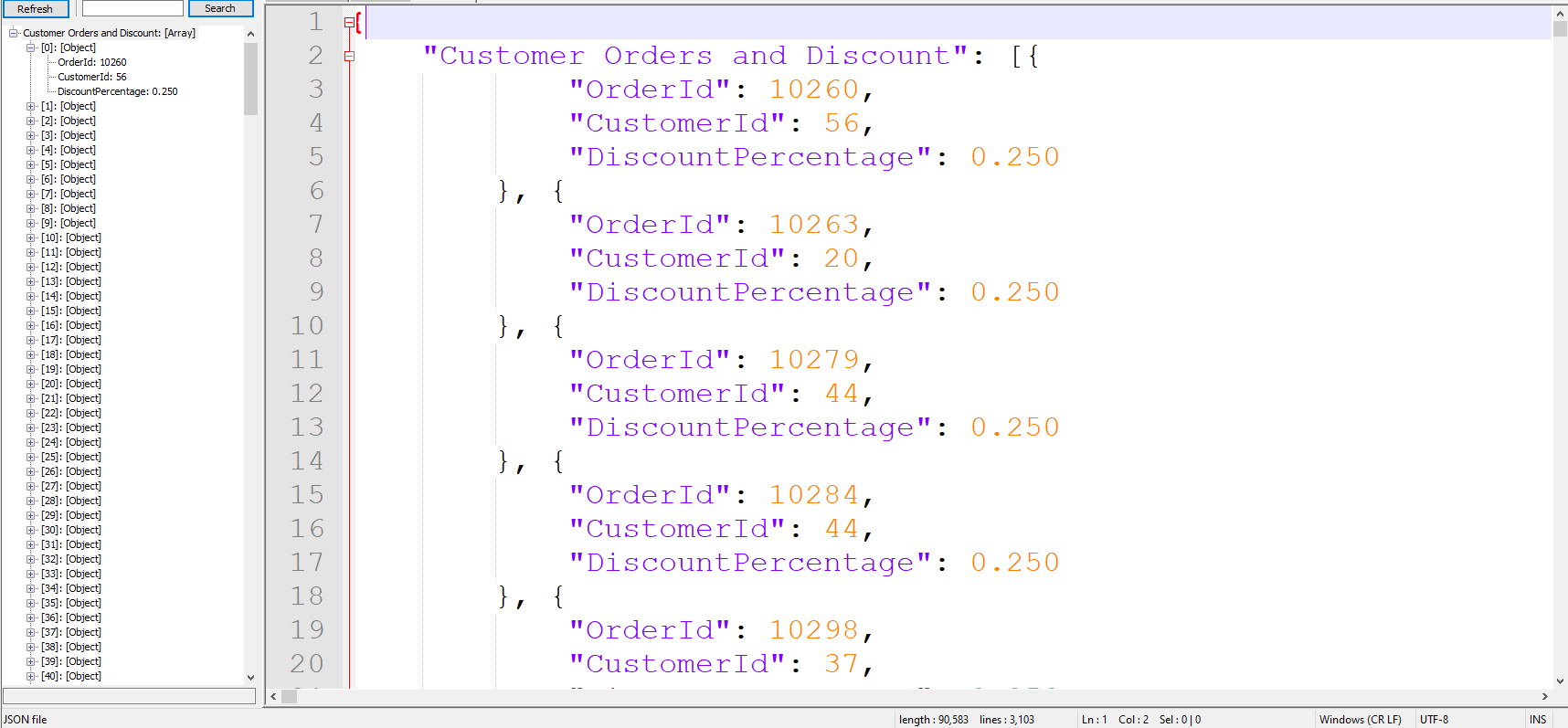
GROUP BY OD.DiscountPercentage, O.CustomerId, OD.OrderId

ORDER BY OD.DiscountPercentage DESC

--Uncomment Below to get JSON output

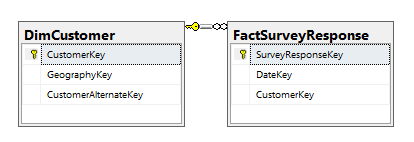
--FOR JSON PATH, ROOT ('Customer Orders and Discount'), INCLUDE\_NULL\_VALUES;





Proposition #3 (Simple)

Show the Customers who left reviews and their Yearly Income





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| FactSurveyResponse | CustomerKey |
| DimCustomer | YearlyIncome |

**Order By**

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

USE AdventureWorksDW2017;

SELECT FSR.CustomerKey,

DC.YearlyIncome

FROM dbo.FactSurveyResponse AS FSR

LEFT OUTER JOIN dbo.DimCustomer AS DC

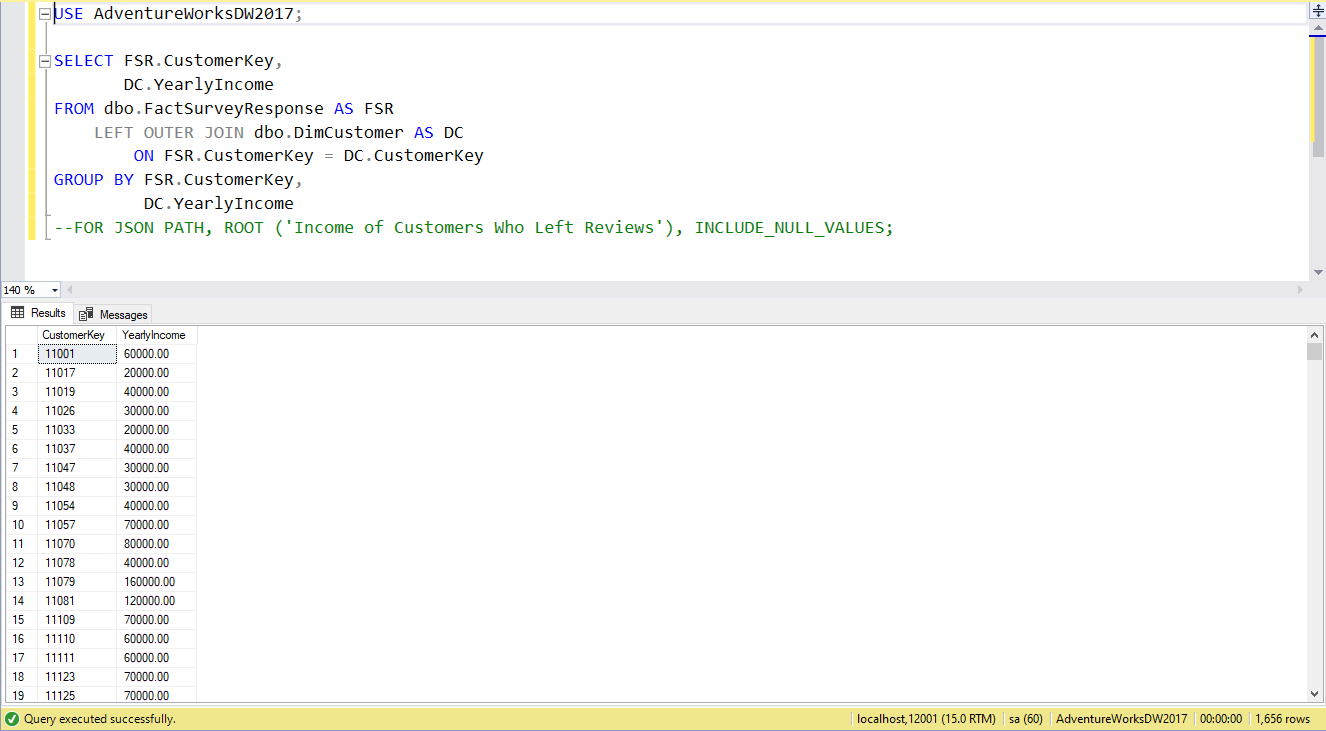
ON FSR.CustomerKey = DC.CustomerKey

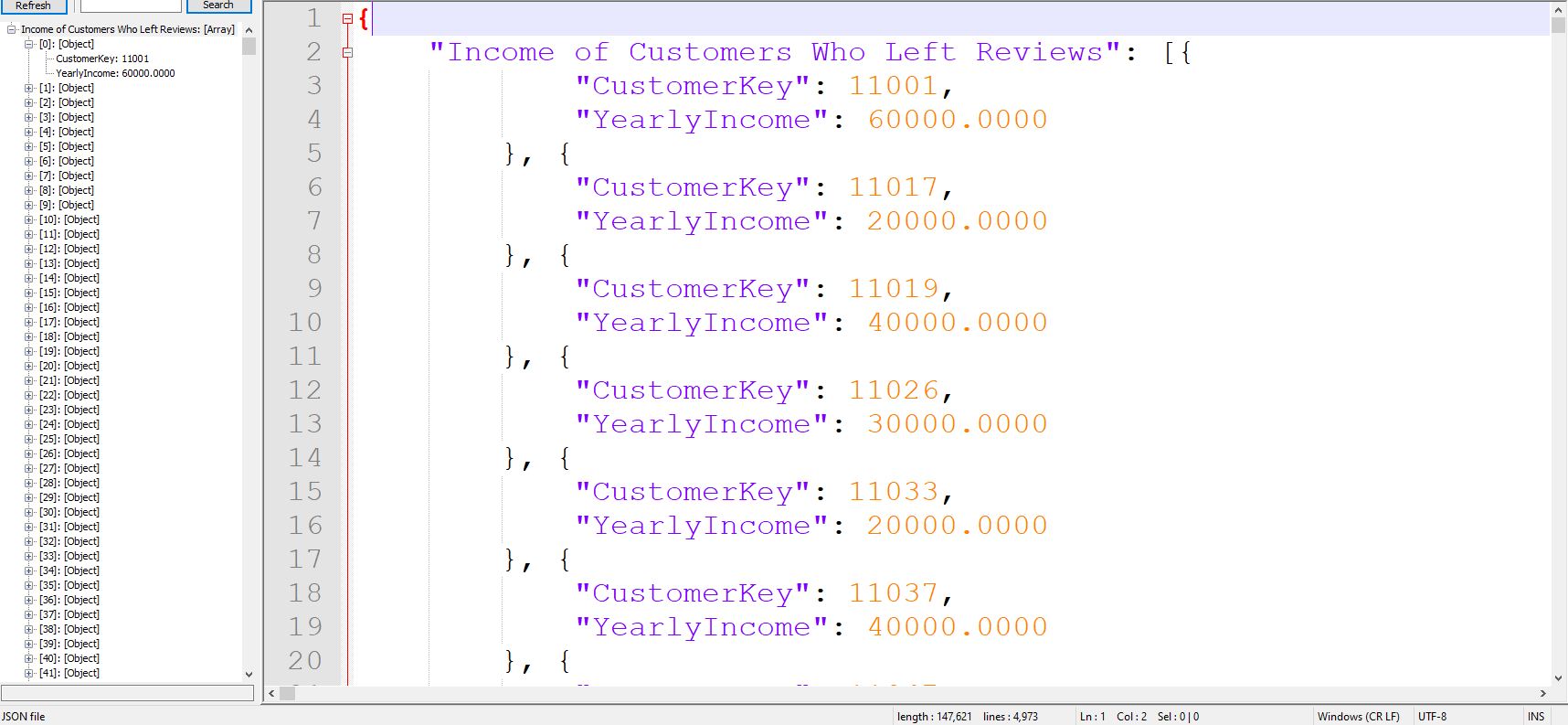
GROUP BY FSR.CustomerKey,

DC.YearlyIncome

--Uncomment Below to get JSON output

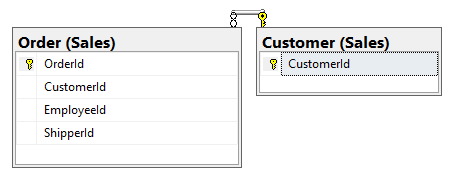
--FOR JSON PATH, ROOT ('Income of Customers Who Left Reviews'), INCLUDE\_NULL\_VALUES;

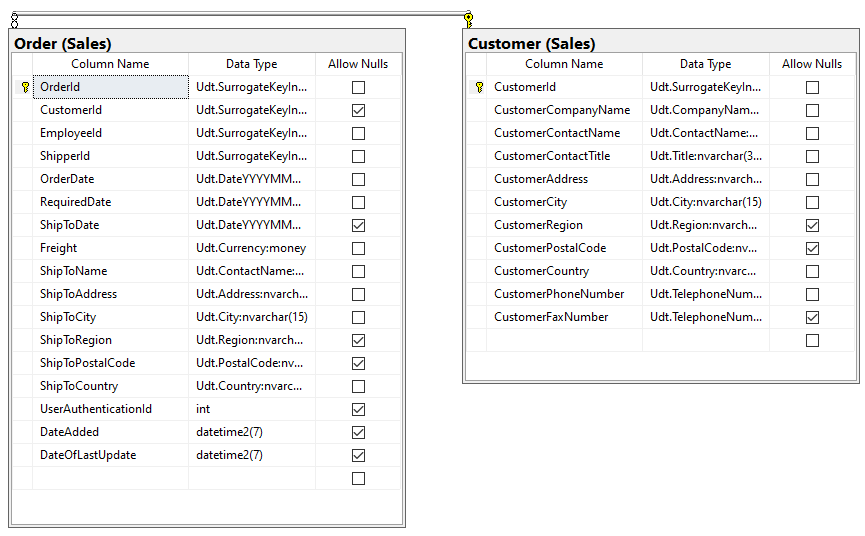




Proposition #4 (Simple)

Show the OrderDates for each order from all Companies





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerCompanyName |
| Order | OrderId, OrderDate |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Customer, Order | CustomerCompanyName, OrderDate | ASC, ASC |

USE Northwinds2020TSQLV6;

SELECT SC.CustomerCompanyName, DS.OrderId, DS.OrderDate

FROM Sales.[Customer] AS SC

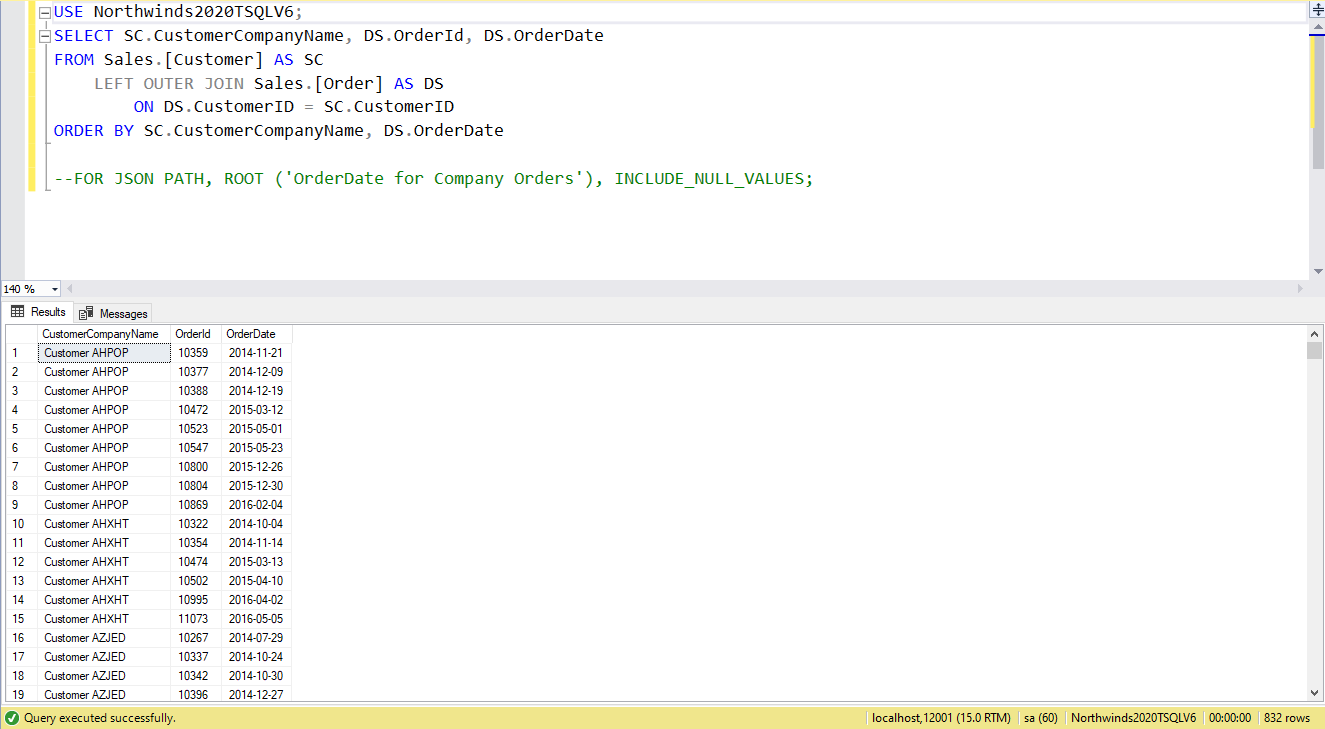
LEFT OUTER JOIN Sales.[Order] AS DS

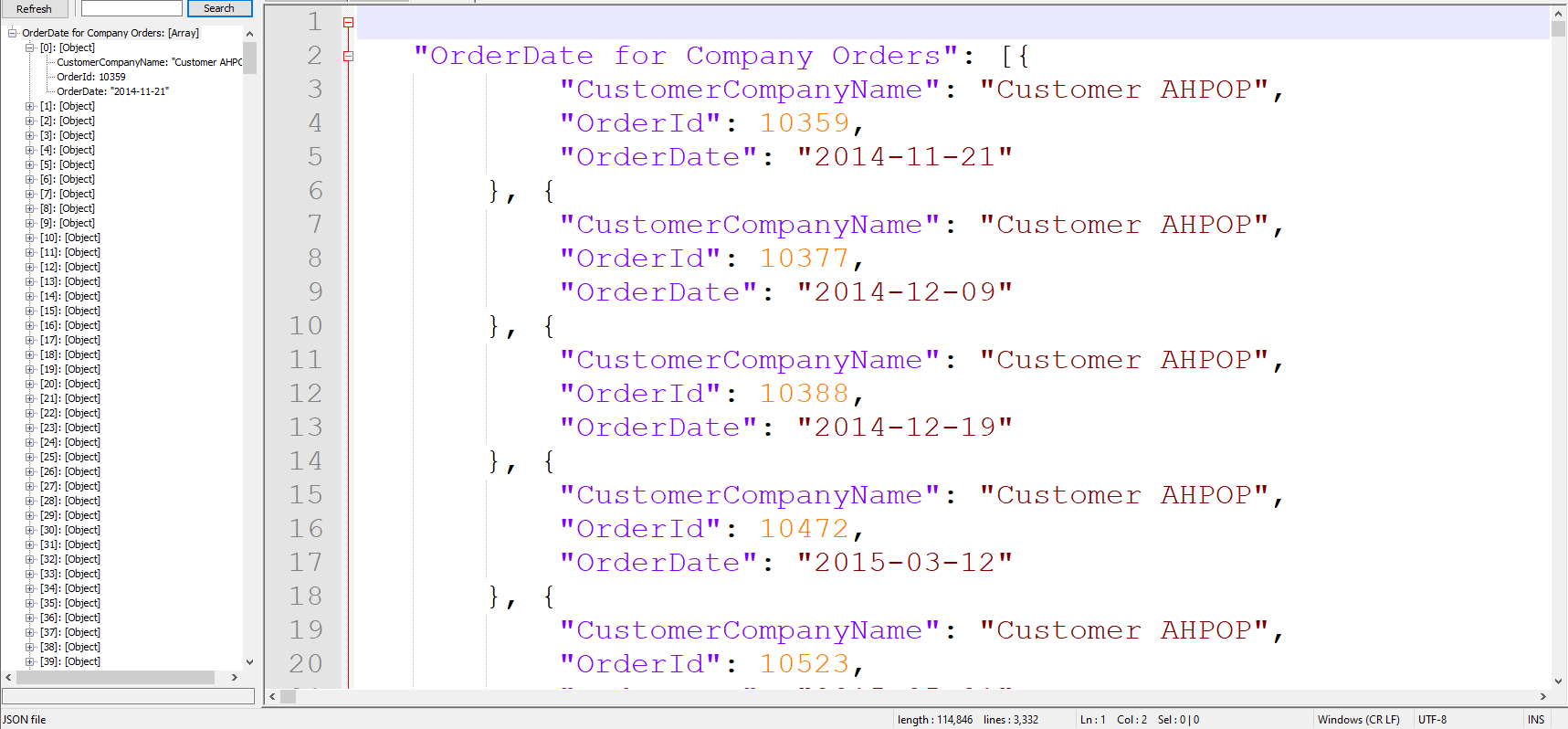
ON DS.CustomerID = SC.CustomerID

ORDER BY SC.CustomerCompanyName, DS.OrderDate

--Uncomment Below to get JSON output

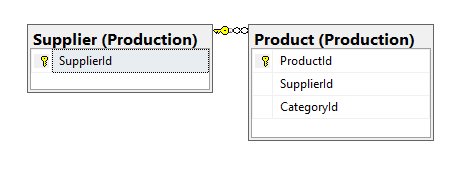
--FOR JSON PATH, ROOT ('OrderDate for Company Orders'), INCLUDE\_NULL\_VALUES;





Proposition #5 (Simple)

Show the Supplier for each Product





|  |  |
| --- | --- |
| Table Name | Column Name |
| Product | ProductId |
| Supplier | SupplierCompanyName |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Product, Supplier | ProductId, SupplierCompanyName | ASC, ASC |

USE Northwinds2020TSQLV6;

SELECT PP.ProductId, PS.SupplierCompanyName

FROM Production.[Product] AS PP

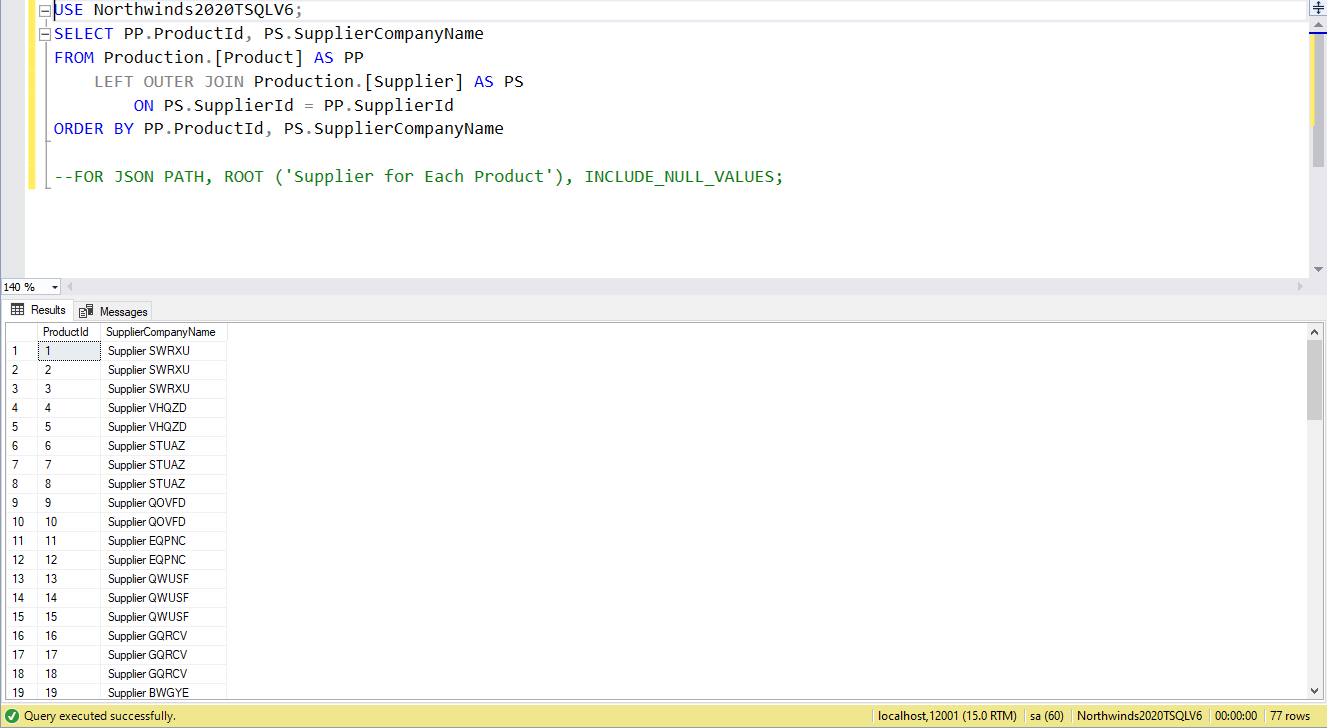
LEFT OUTER JOIN Production.[Supplier] AS PS

ON PS.SupplierId = PP.SupplierId

ORDER BY PP.ProductId, PS.SupplierCompanyName

--Uncomment Below to get JSON output

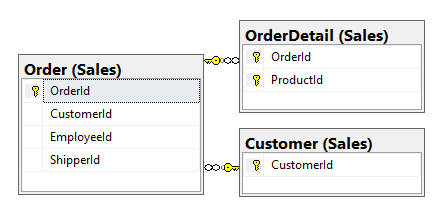
--FOR JSON PATH, ROOT ('Supplier for Each Product'), INCLUDE\_NULL\_VALUES;

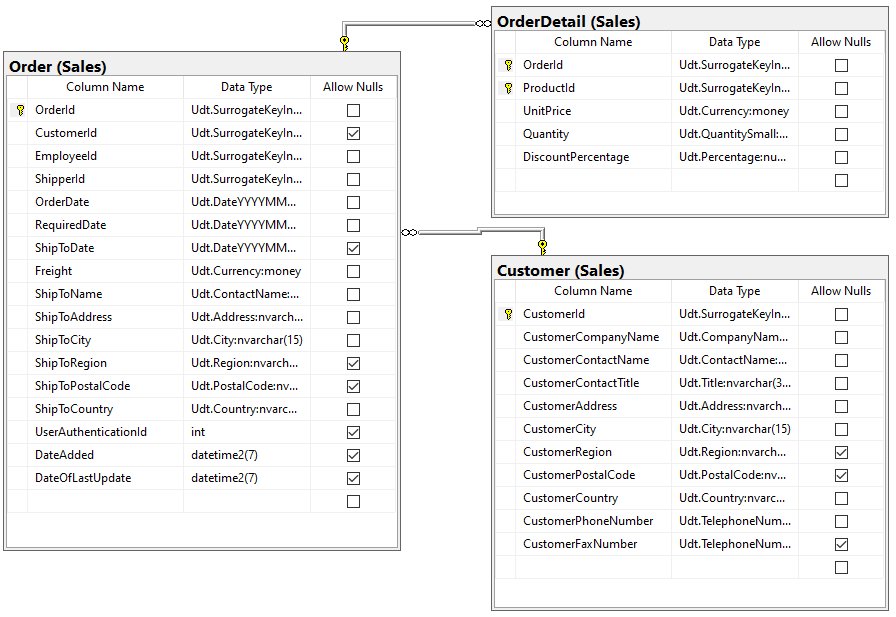




Proposition #6 (Medium)

Shows the Total Units each Employee has sold to Each Customer





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerId |
| Order | OrderId, EmployeeId |
| OrderDetail | Quantity |

**Order By**

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

USE Northwinds2020TSQLV6;

SELECT O.EmployeeId,

C.CustomerId,

SUM(OD.Quantity) AS TotalUnitsSold

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

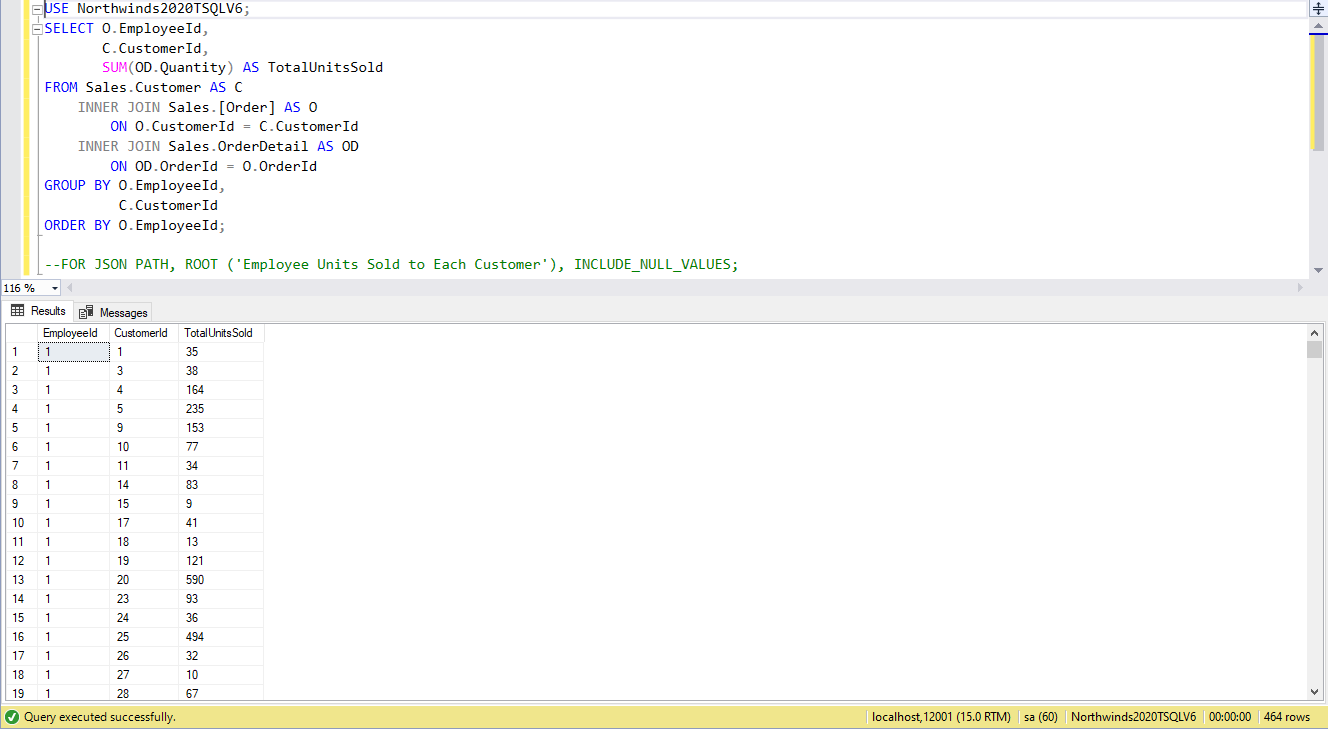
GROUP BY O.EmployeeId,

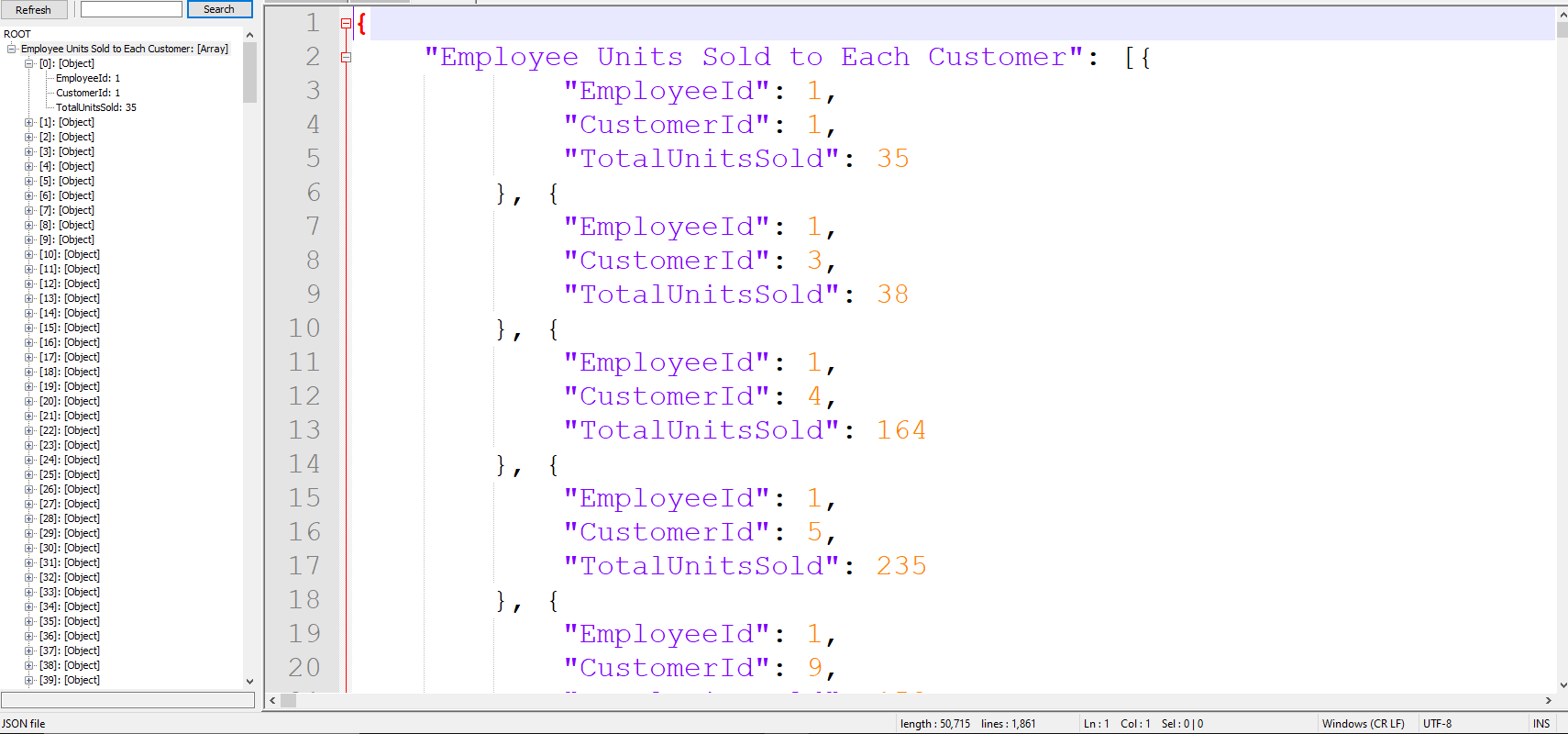
C.CustomerId

ORDER BY O.EmployeeId

--Uncomment Below to get JSON output

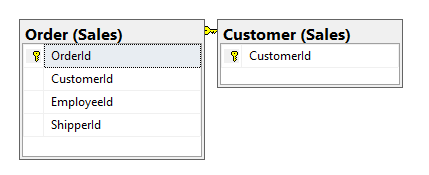
--FOR JSON PATH, ROOT ('Employee Units Sold to Each Customer'), INCLUDE\_NULL\_VALUES;

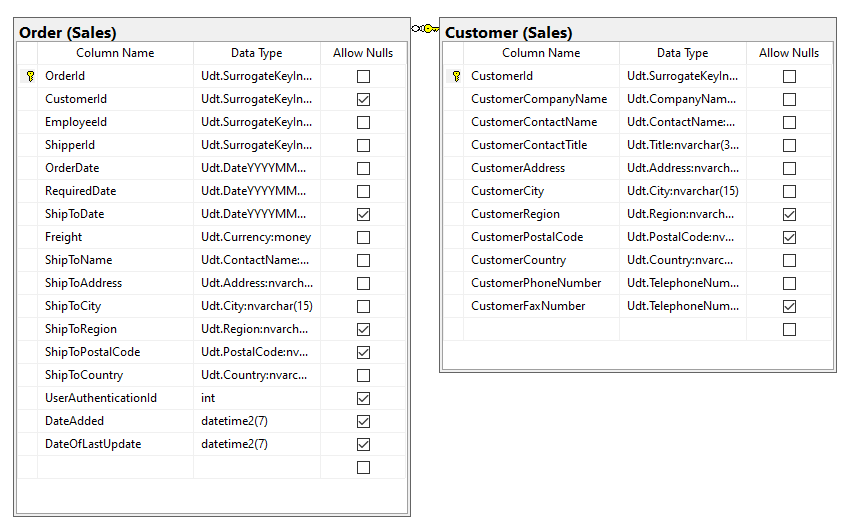




Proposition #7 (Medium)

Show the OrderDate and ShipToDate and Find the amount of days it will take until arrival





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerId |
| Order | OrderId, OrderDate, ShipToDate |

**Order By**

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

USE Northwinds2020TSQLV6;

SELECT C.CustomerId,

O.OrderId,

O.OrderDate,

O.ShipToDate,

DATEDIFF(DAY, O.OrderDate, O.ShipToDate) AS DaysFromOrderToShip

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

WHERE ShipToDate IS NOT NULL

GROUP BY DATEDIFF(DAY, O.OrderDate, O.ShipToDate),

C.CustomerId,

O.OrderId,

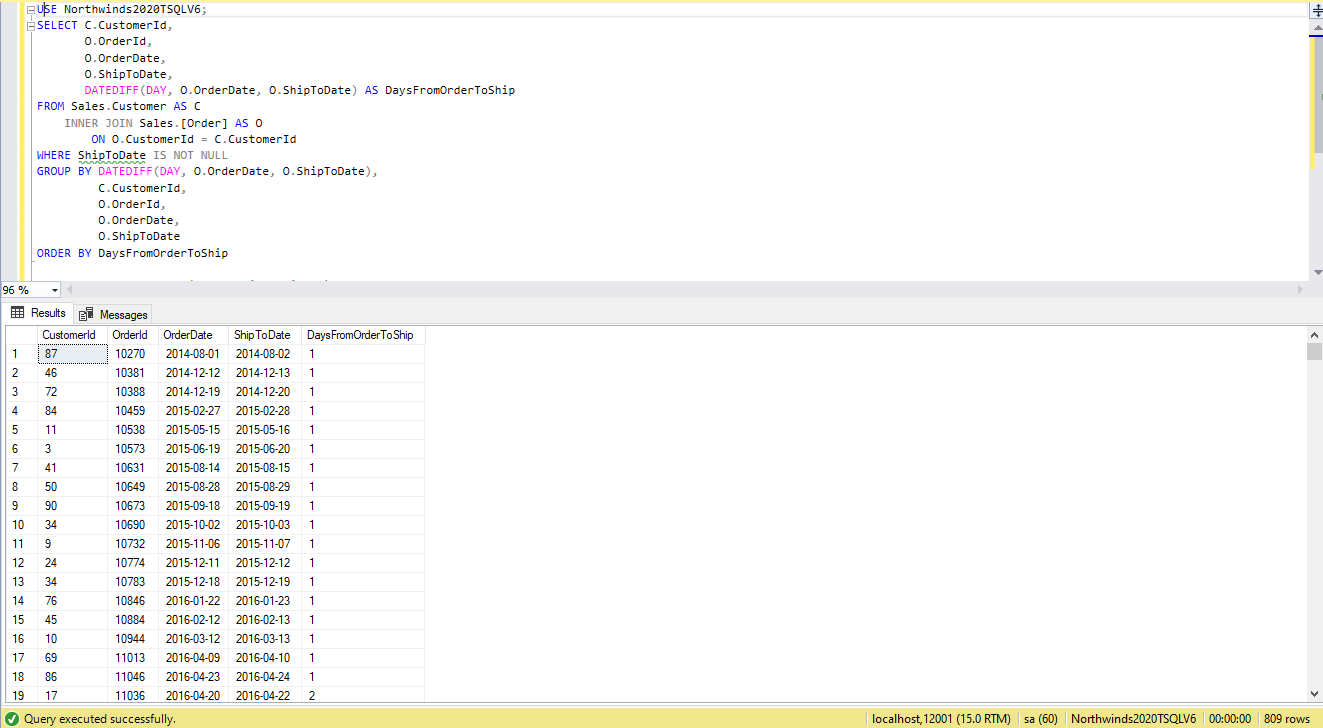
O.OrderDate,

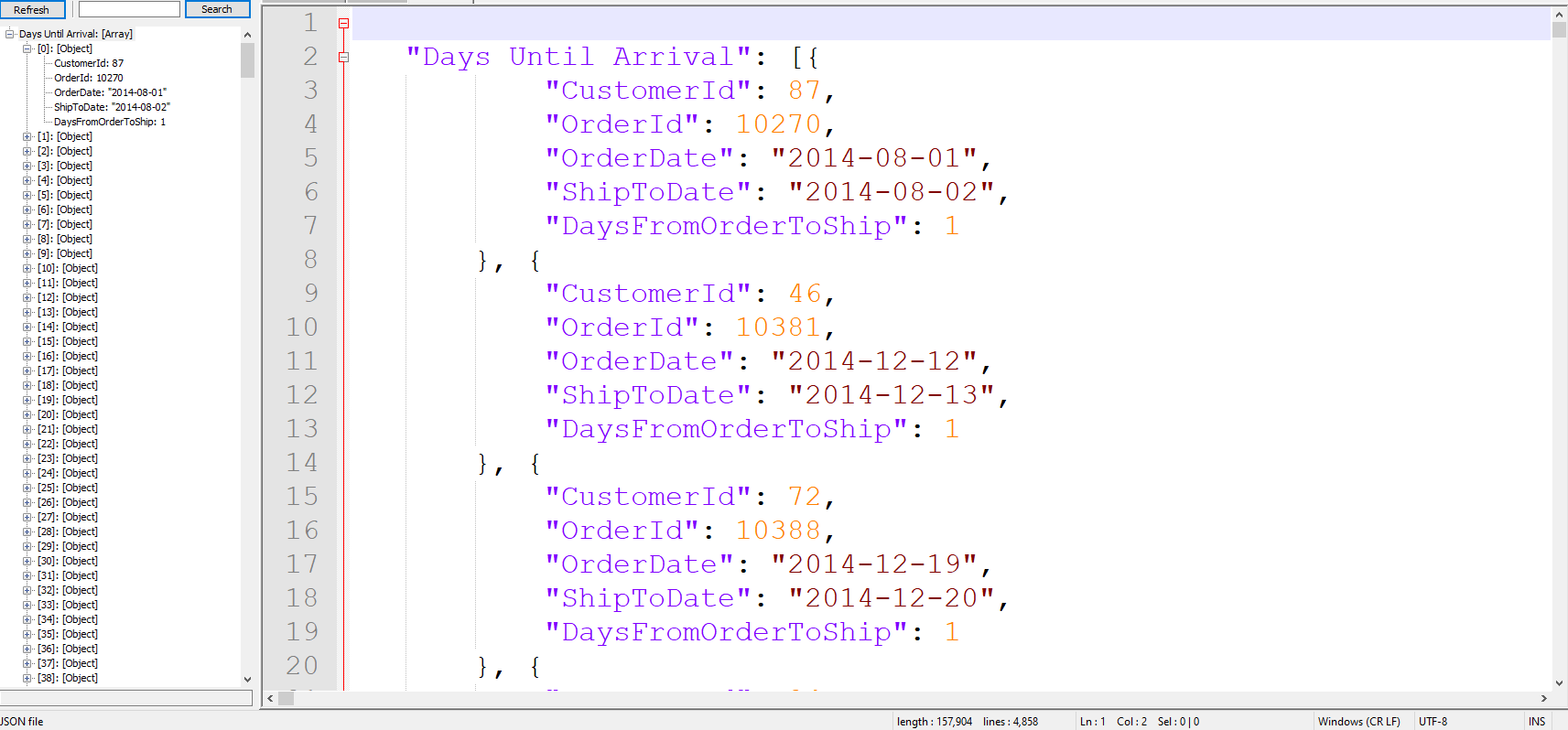
O.ShipToDate

ORDER BY DaysFromOrderToShip

--Uncomment Below to get JSON output

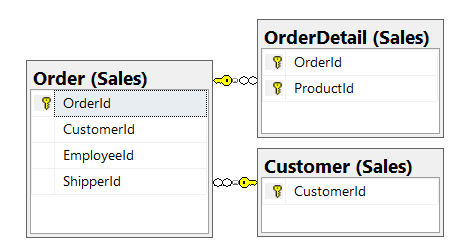
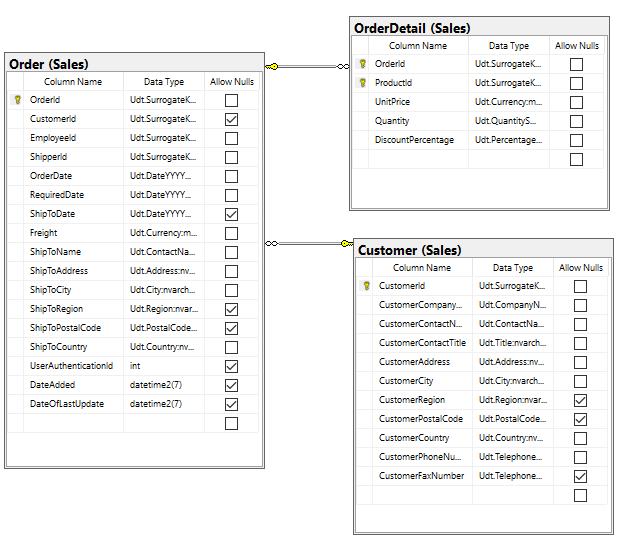
--FOR JSON PATH, ROOT (Days Until Arrival'), INCLUDE\_NULL\_VALUES;





Proposition #8 (Medium)

Shows all Orders going to USA, along with the City and Quantity

**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Customer | CustomerId |
| Order | OrderId, OrderDate, ShipToDate |

**Order By**

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

USE Northwinds2020TSQLV6;

SELECT C.CustomerId,

O.OrderId,

O.ShipToCity,

SUM(OD.Quantity) AS totalqty

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

WHERE C.CustomerCountry = N'SPAIN'

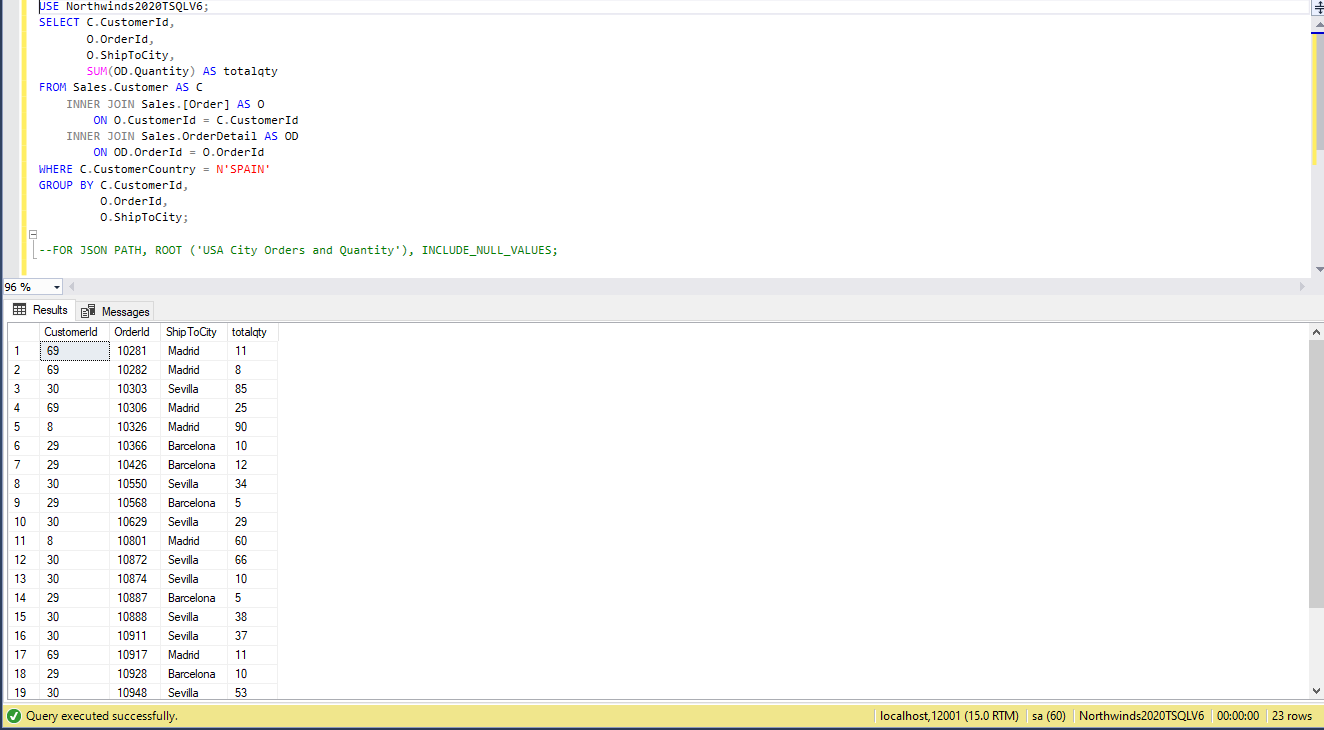
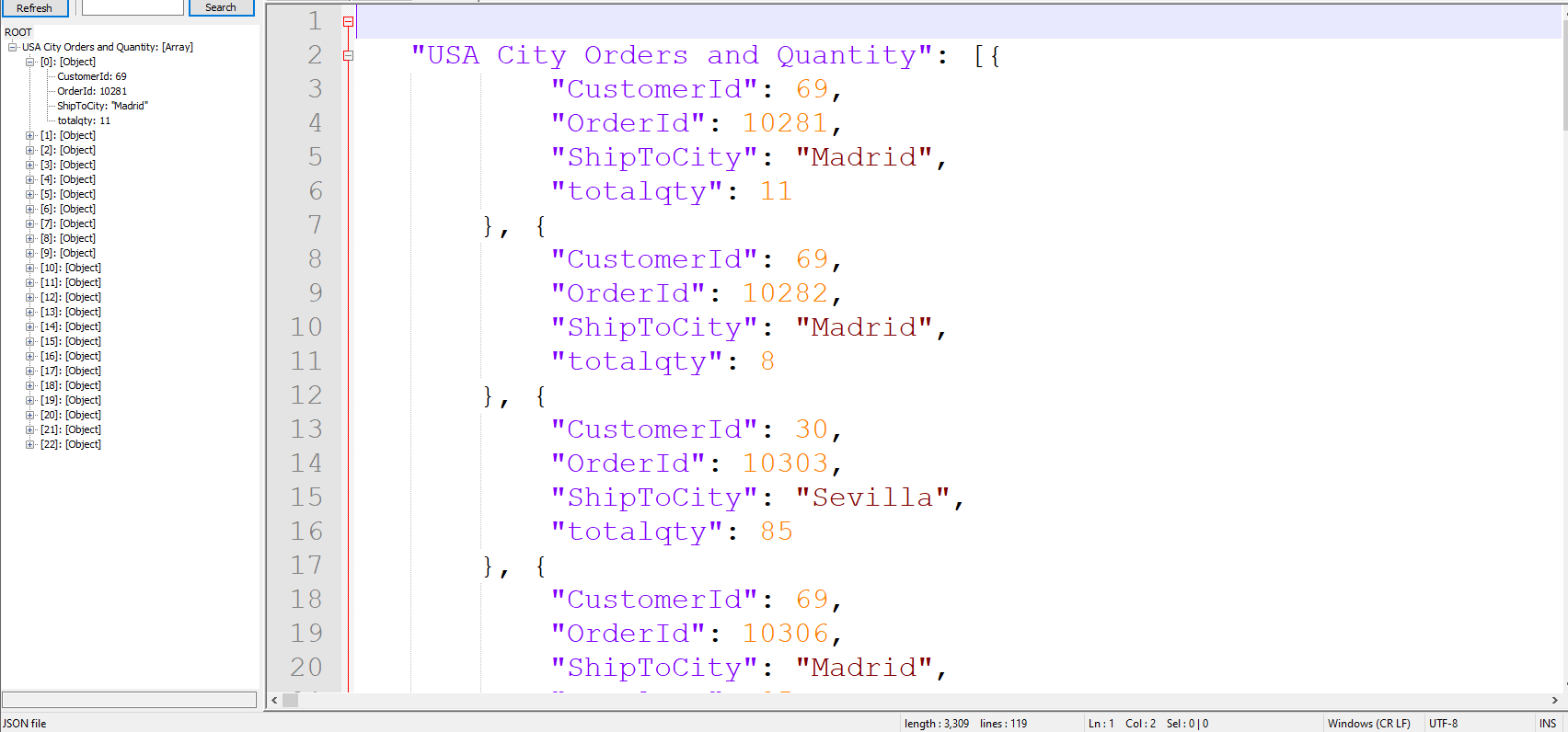
GROUP BY C.CustomerId,

O.OrderId,

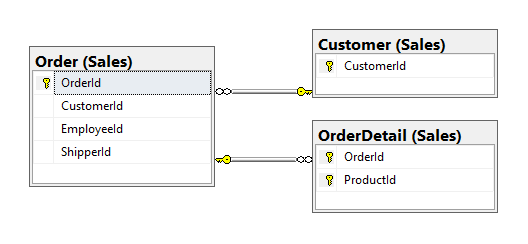
O.ShipToCity

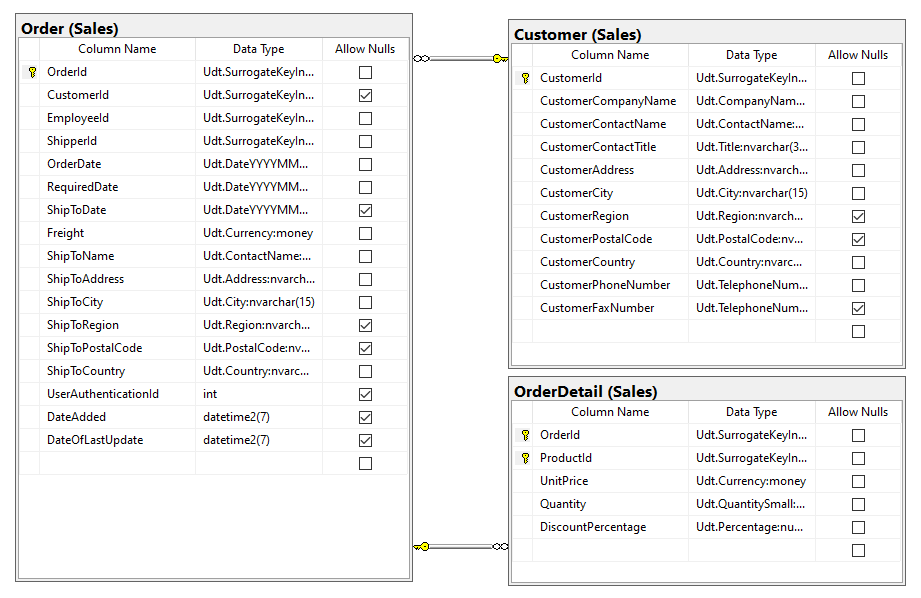
--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('USA City Orders and Quantity'), INCLUDE\_NULL\_VALUES;

Proposition #9 (Medium)

Shows the total quantity of units going to each country 



**Columns from Tables**

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

**Order By**

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

USE Northwinds2020TSQLV6;

SELECT C.CustomerCountry,

O.OrderId,

SUM(OD.Quantity) AS TotalUnitsSold

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

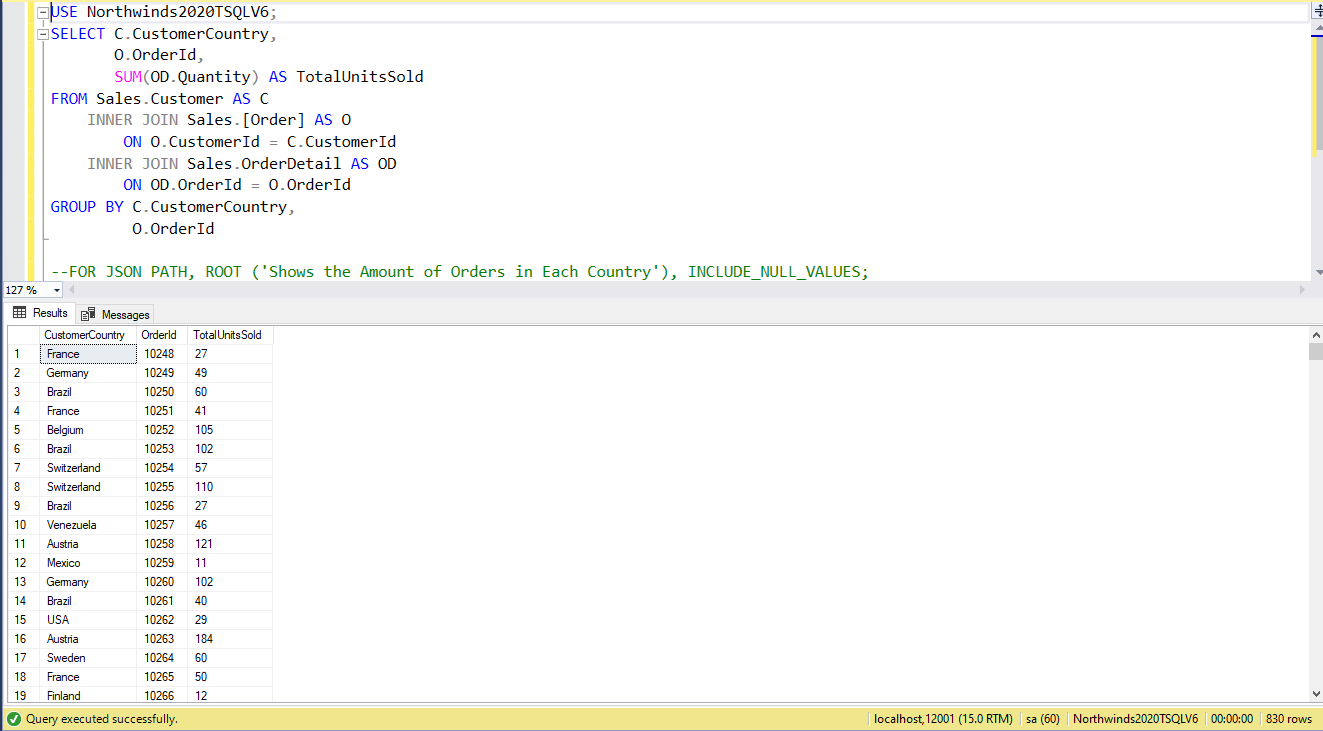
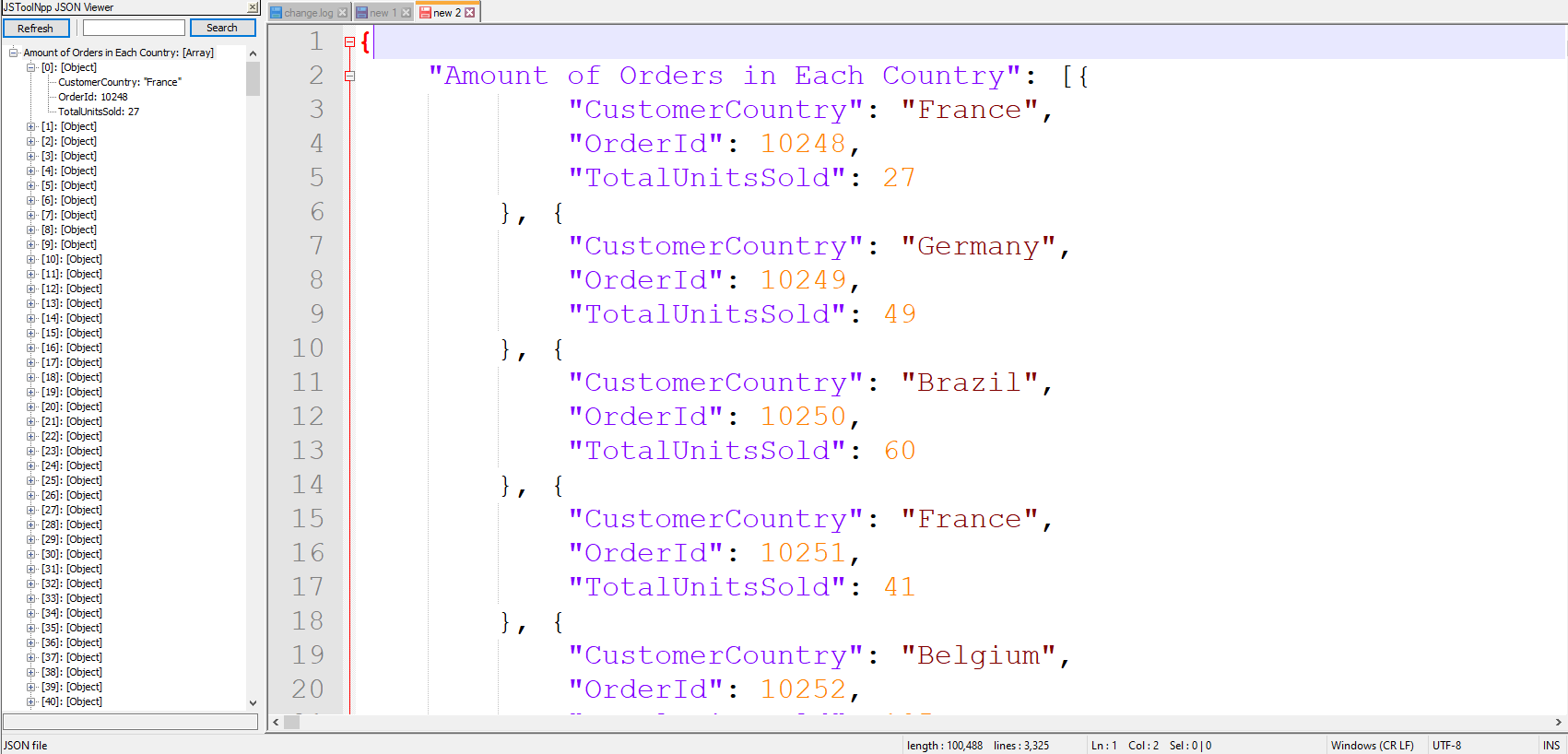
ON OD.OrderId = O.OrderId

GROUP BY C.CustomerCountry,

O.OrderId

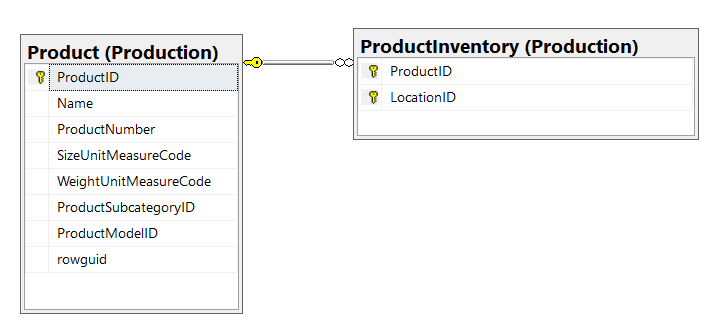
--Uncomment Below to get JSON output

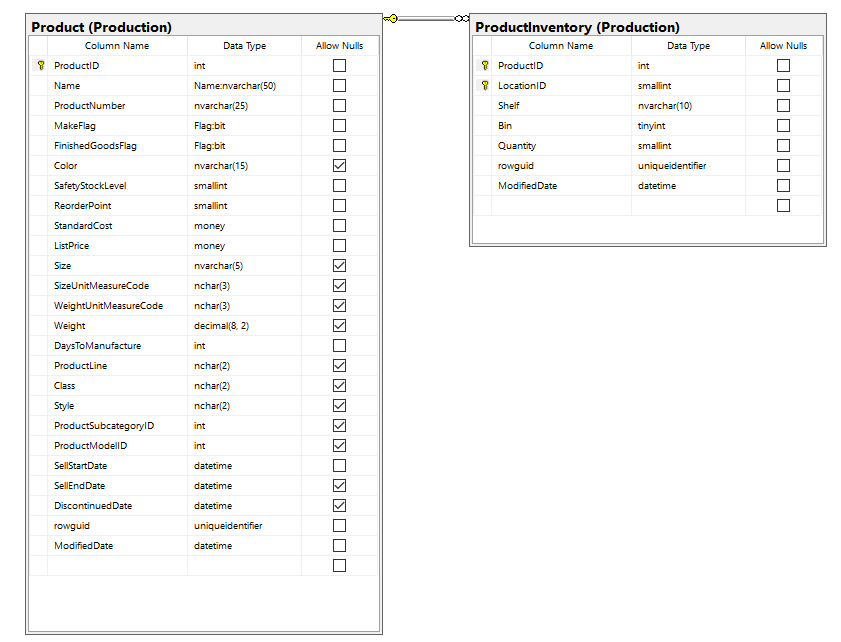
--FOR JSON PATH, ROOT ('Amount of Orders in Each Country'), INCLUDE\_NULL\_VALUES;

Proposition #10 (Medium)

Show Products that are no longer in stock (Quantity Zero)





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Product | ProductID, Name |
| ProductInventory | Quantity |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Product | ProductID | ASC |

USE AdventureWorks2017;

SELECT PP.ProductID,

PP.Name,

PINV.Quantity

FROM Production.[Product] AS PP

LEFT OUTER JOIN Production.[ProductInventory] AS PINV

ON PP.ProductID = PINV.ProductID

GROUP BY PP.ProductID,

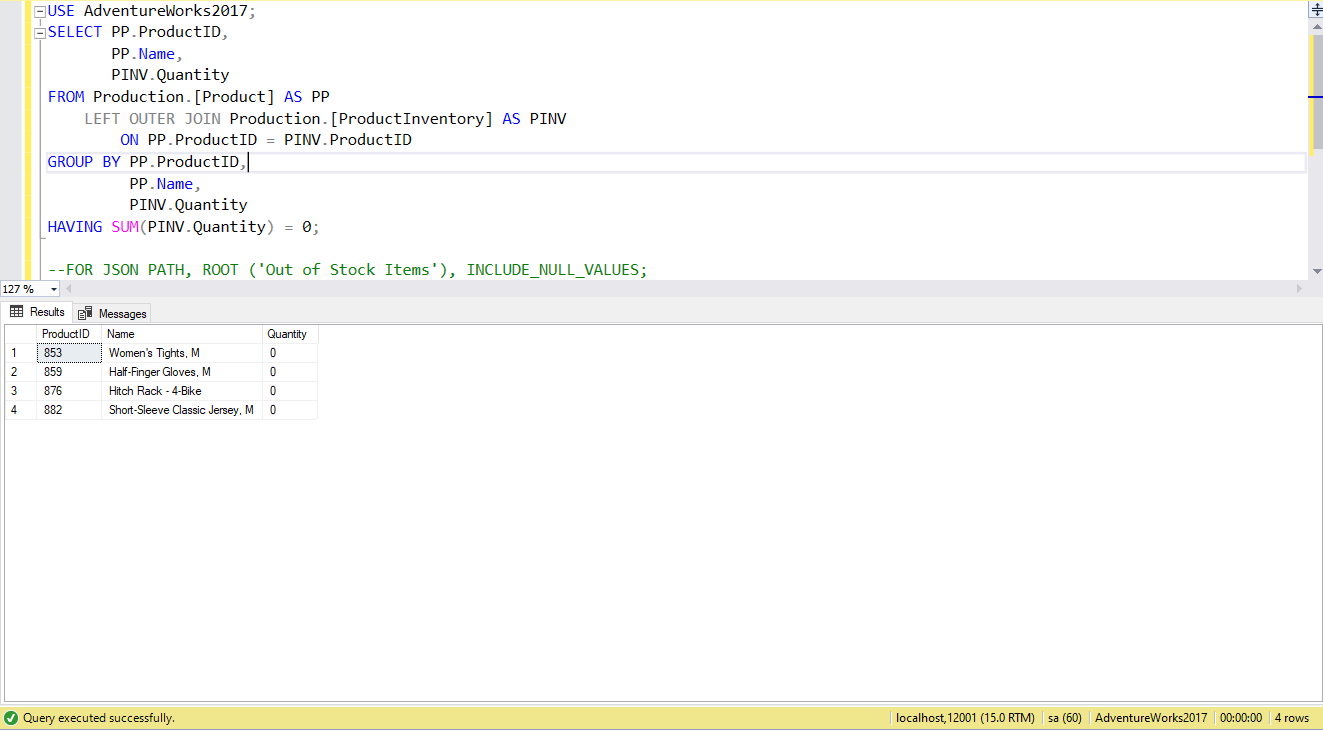
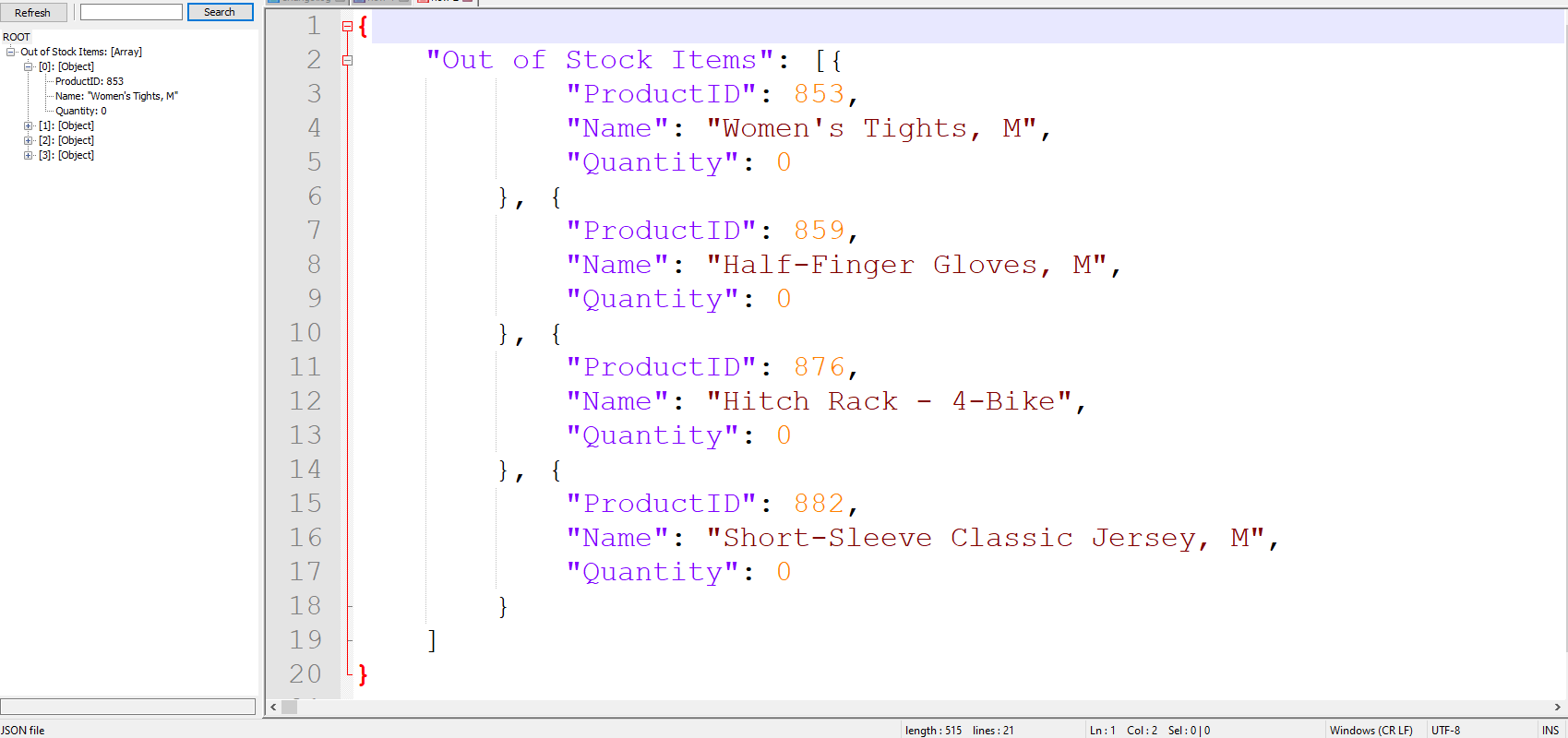
PP.Name,

PINV.Quantity

HAVING SUM(PINV.Quantity) = 0

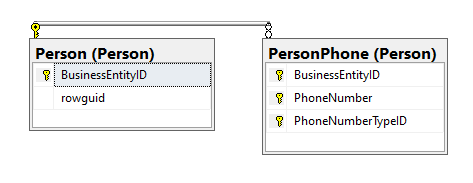
--Uncomment Below to get JSON output

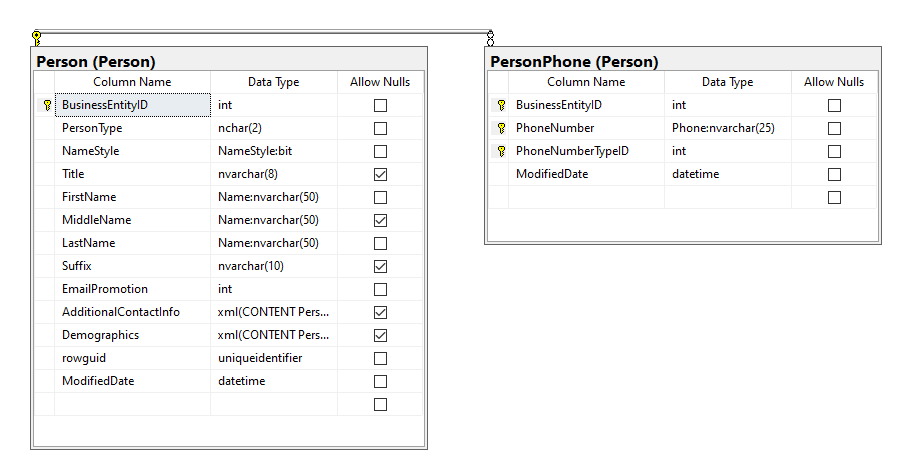
--FOR JSON PATH, ROOT ('Out of Stock Items'), INCLUDE\_NULL\_VALUES;

Proposition #11 (Medium)

Show the FullName of each person and their PhoneNumber





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| PersonPhone | FirstName, LastName |
| Person | PhoneNumber |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Person | FullName | ASC |

USE AdventureWorks2017;

SELECT CONCAT (P.FirstName, ' ', P.LastName) as FullName, PP.PhoneNumber

FROM Person.[Person] AS P

LEFT OUTER JOIN Person.[PersonPhone] AS PP

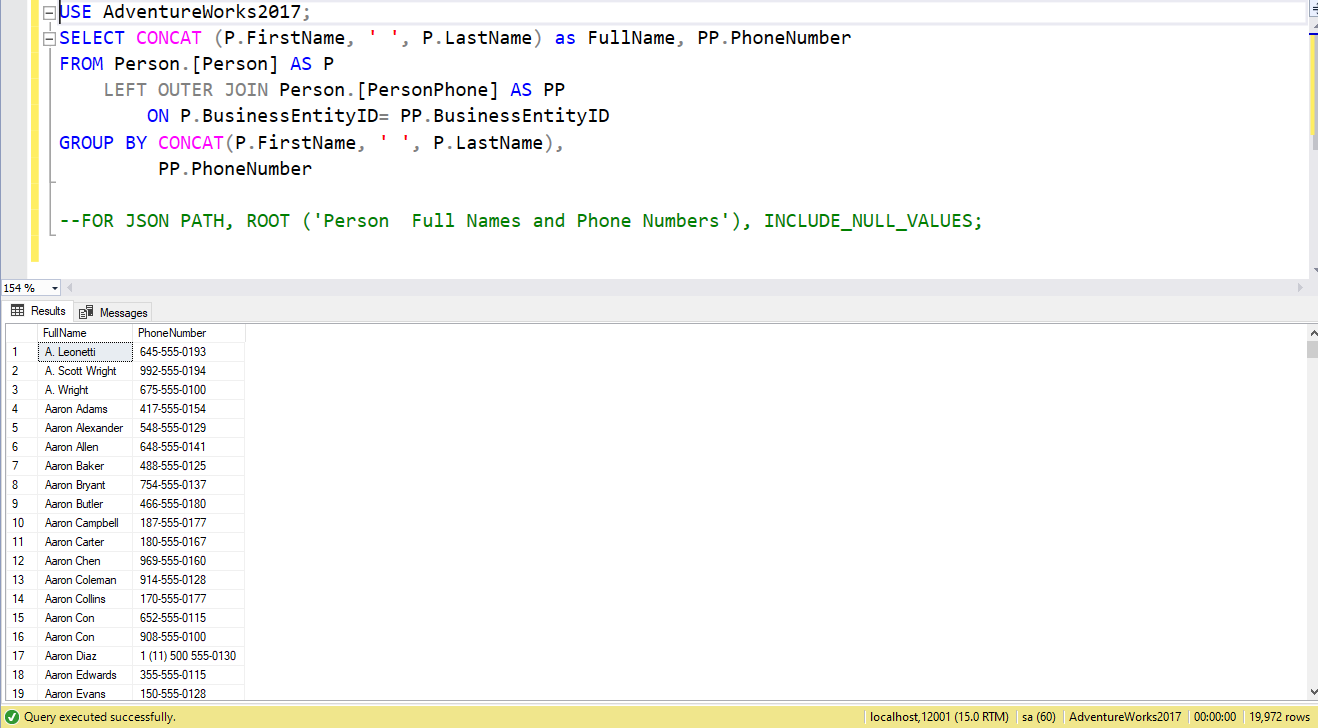
ON P.BusinessEntityID= PP.BusinessEntityID

GROUP BY CONCAT(P.FirstName, ' ', P.LastName),

PP.PhoneNumber

--Uncomment Below to get JSON output

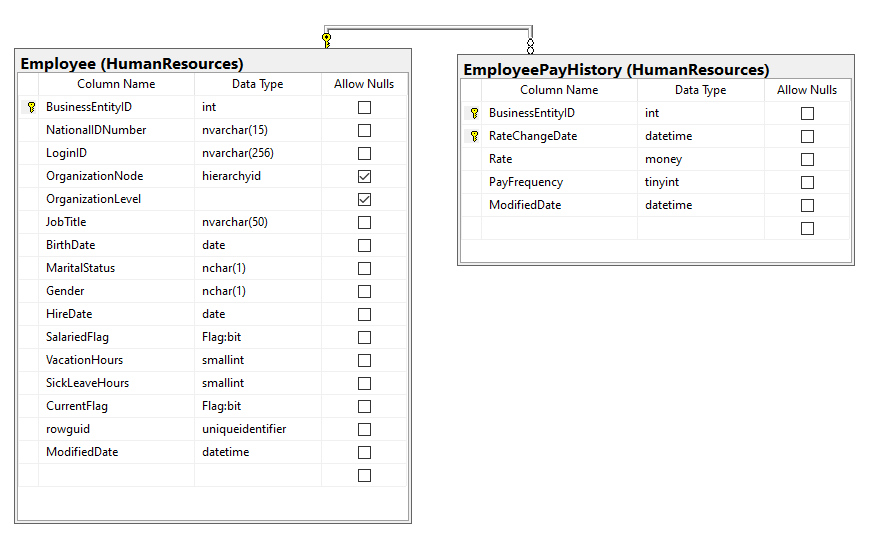
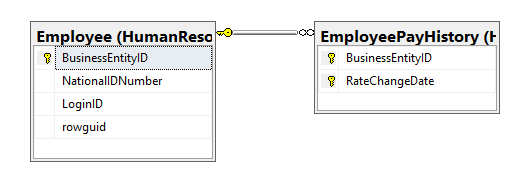
--FOR JSON PATH, ROOT ('Person Full Names and Phone Numbers'), INCLUDE\_NULL\_VALUES;





Proposition #12 (Medium)

Show the Average Pay Rate of the different Job Titles



**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Employee | JobTitle |
| EmployeePayHistory | Rate |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| EmployeePayHistory | AverageRate | ASC |

USE AdventureWorks2017;

SELECT DISTINCT

HE.JobTitle,

AVG(HP.Rate) AS AverageRate

FROM HumanResources.[Employee] AS HE

LEFT OUTER JOIN HumanResources.EmployeePayHistory AS HP

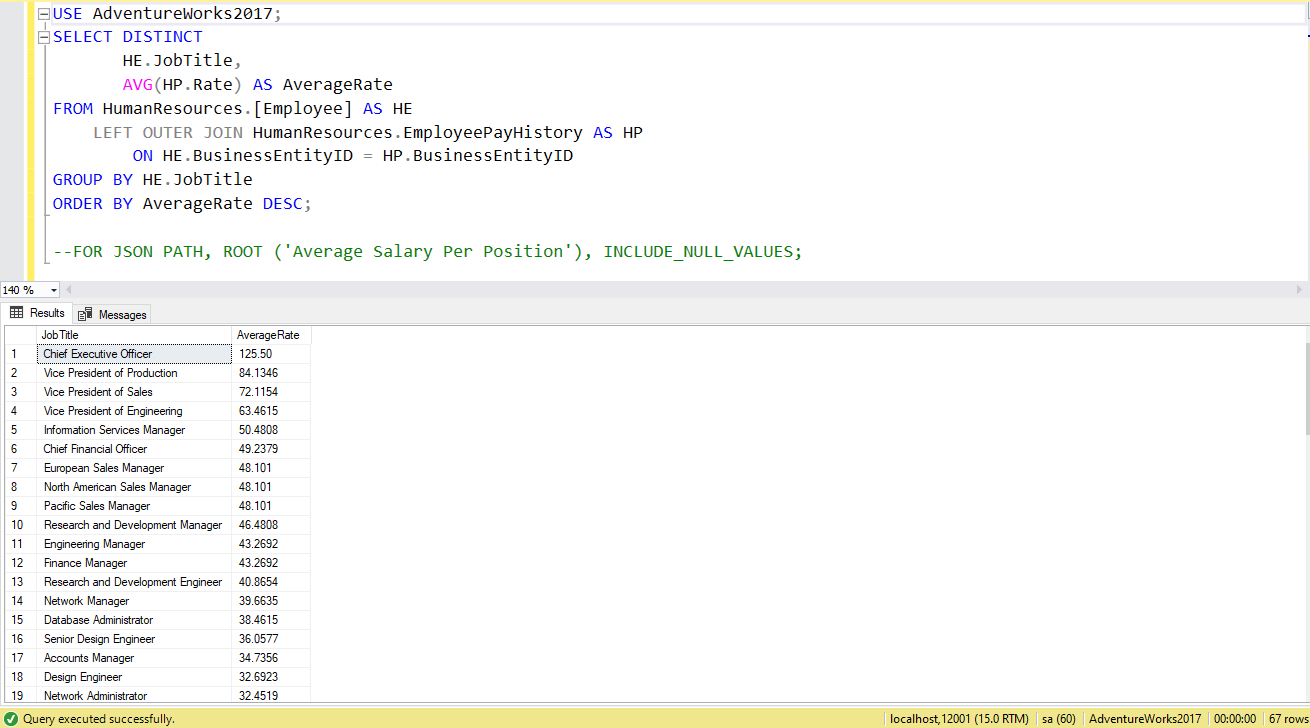
ON HE.BusinessEntityID = HP.BusinessEntityID

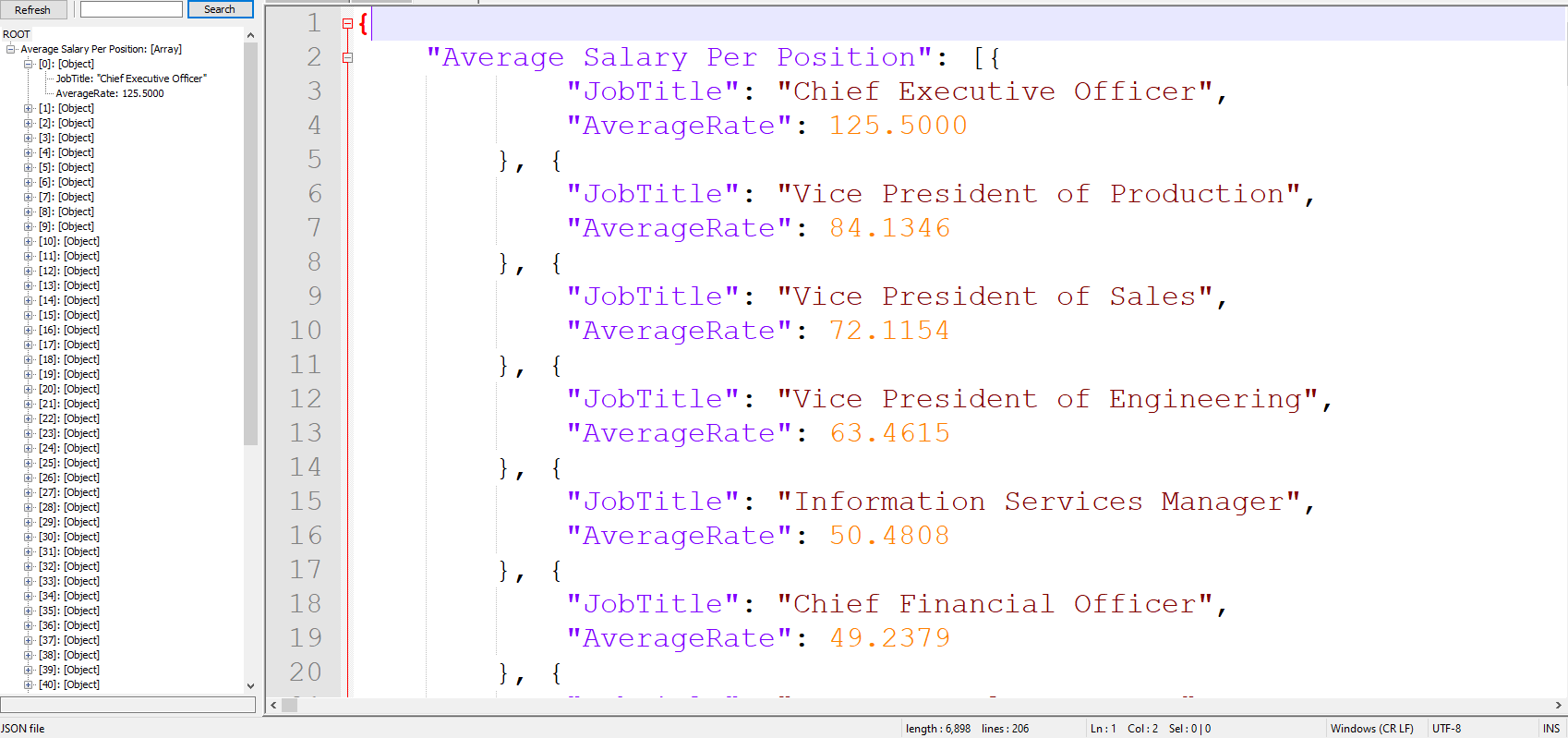
GROUP BY HE.JobTitle

ORDER BY AverageRate DESC

--Uncomment Below to get JSON output

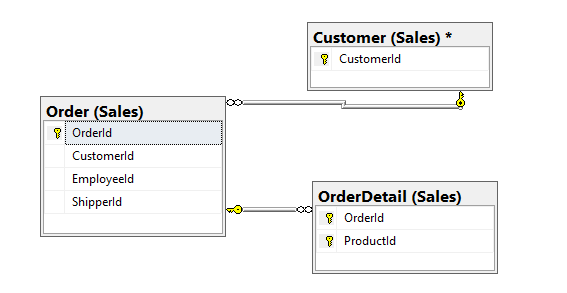
--FOR JSON PATH, ROOT ('Average Salary Per Position'), INCLUDE\_NULL\_VALUES;

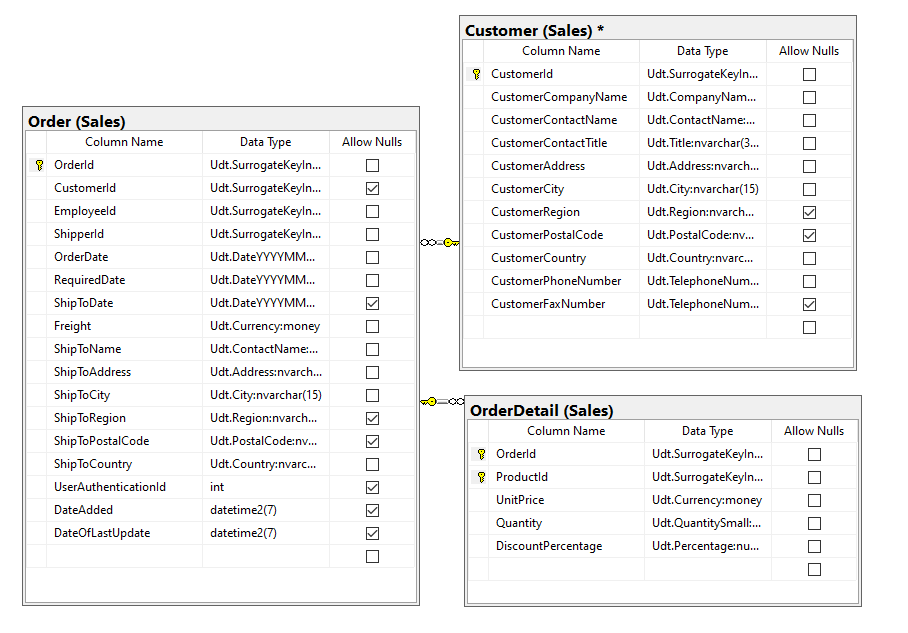




Proposition #13 (Complex)

Show the cost of products in all orders for a specified customer





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId |
| Customer | CustomerId |
| OrderDetail | ProductId, UnitPrice, Quantity |

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.get\_cost\_of\_products\_for\_customer\_order;

GO

CREATE FUNCTION dbo.get\_cost\_of\_products\_for\_customer\_order

(

@customer AS INT

)

RETURNS TABLE

AS

RETURN SELECT CustomerId,

OrderId

FROM Sales.[Order]

WHERE CustomerId = @customer;

GO

DECLARE @CustomerNumber INT;

SET @CustomerNumber = 50;

SELECT SO.CustomerId,

GC.OrderId,

SOD.ProductId,

(SOD.UnitPrice \* SOD.Quantity) AS TotalCost

FROM dbo.get\_cost\_of\_products\_for\_customer\_order(@CustomerNumber) AS GC

INNER JOIN Sales.[Customer] AS SO

ON GC.CustomerId = SO.CustomerId

INNER JOIN Sales.[OrderDetail] AS SOD

ON SOD.OrderId = GC.OrderId

GROUP BY (SOD.UnitPrice \* SOD.Quantity),

SO.CustomerId,

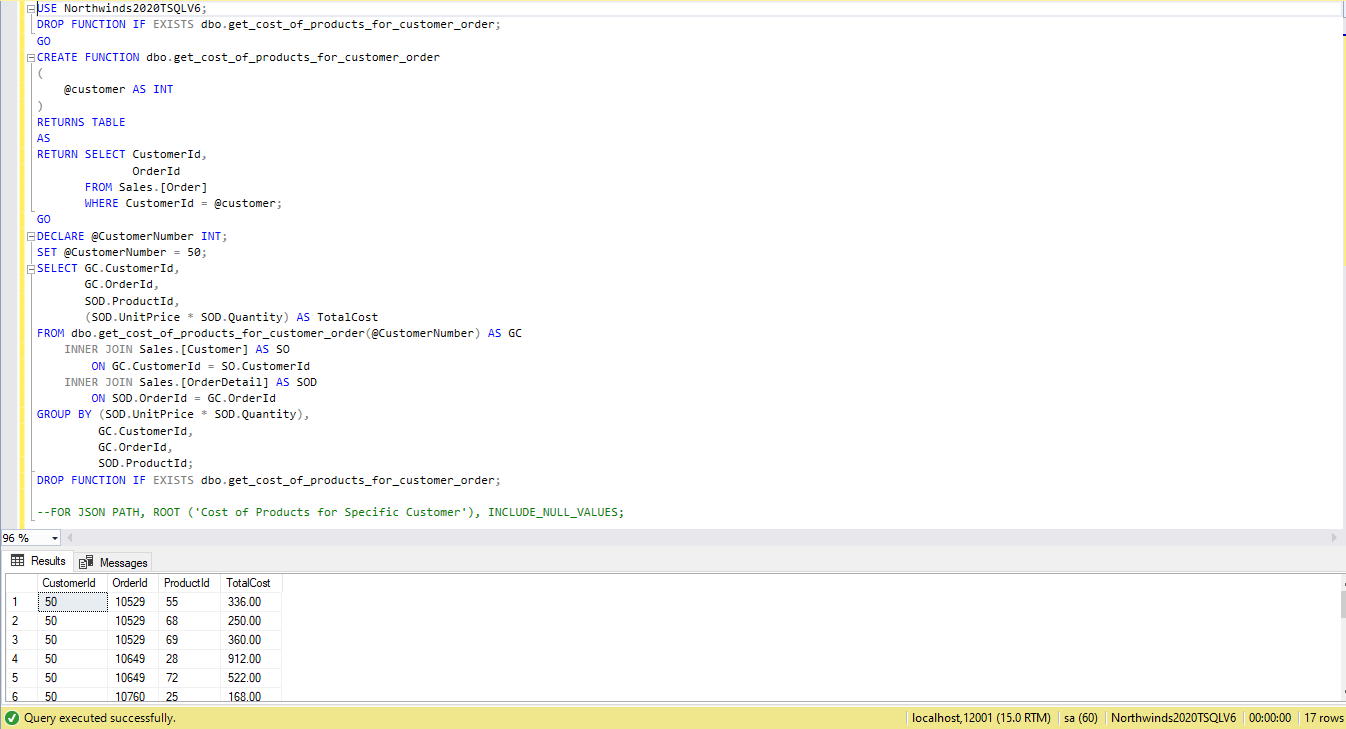
GC.OrderId,

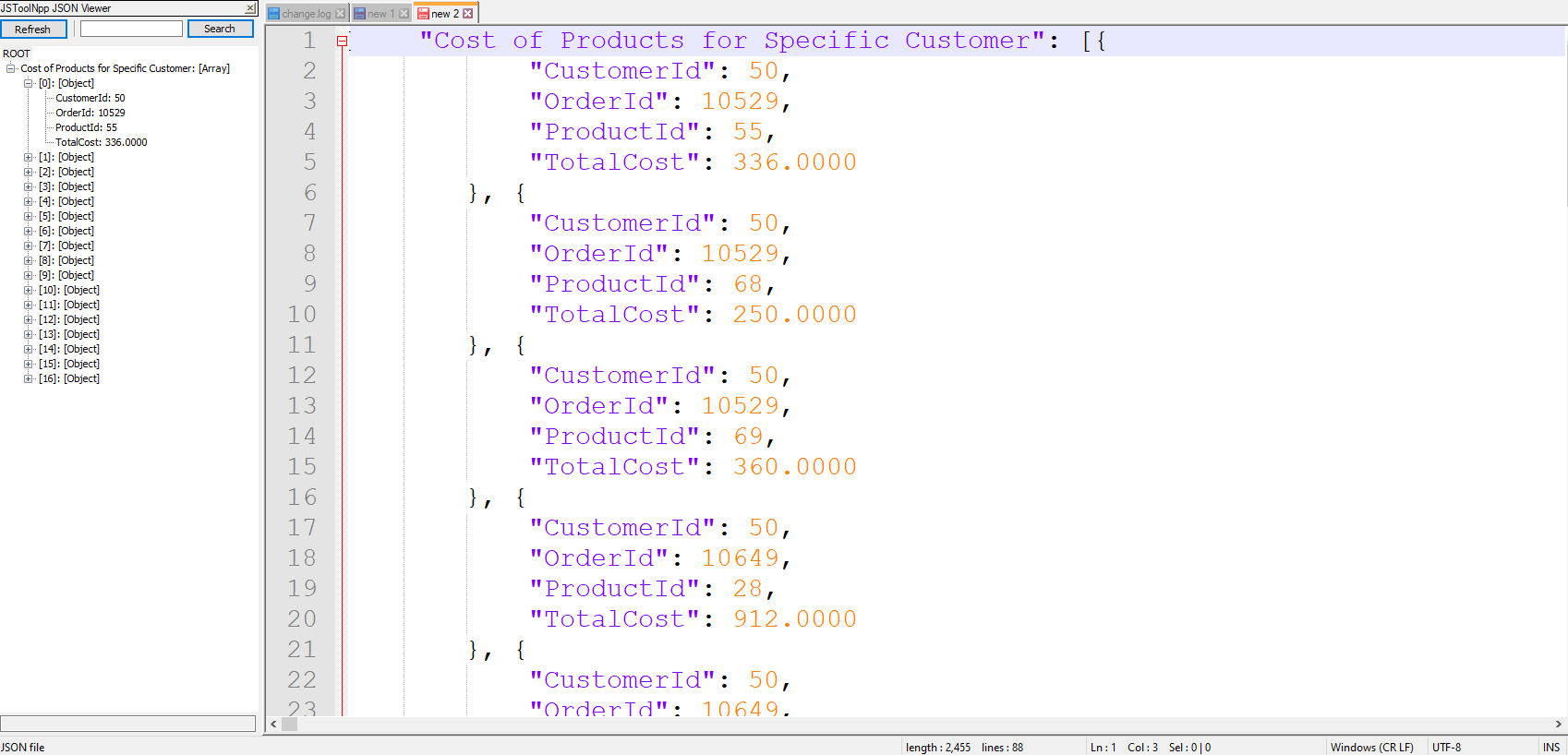
SOD.ProductId

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Cost of Products for Specific Customer'), INCLUDE\_NULL\_VALUES;

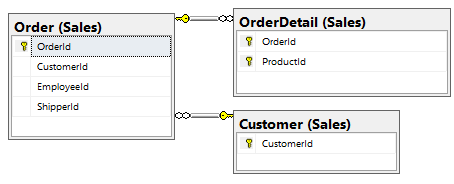
DROP FUNCTION IF EXISTS dbo.get\_cost\_of\_products\_for\_customer\_order

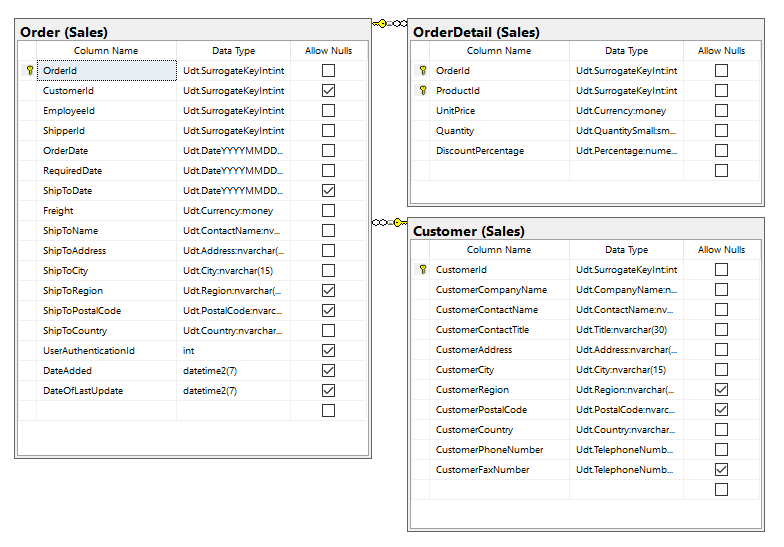




Proposition #14 (Complex)

Show the Amount of Days until the Order is Shipped





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId |
| Customer | CustomerId |
| OrderDetail | OrderDate, ShipToDate |

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.get\_days\_until\_ship;

GO

CREATE FUNCTION dbo.get\_days\_until\_ship(@ordernumber int)

RETURNS INT

AS

BEGIN

DECLARE @days\_until\_ship INT;

SELECT @days\_until\_ship = DATEDIFF(DAY, OrderDate, ShipToDate)

FROM Sales.[Order]

WHERE OrderId = @ordernumber

RETURN @days\_until\_ship;

END;

GO

DECLARE @order\_number int;

SET @order\_number = 10270

SELECT SO.CustomerId, SO.orderid, dbo.get\_days\_until\_ship(@order\_number) AS DAYSUNTILSHIP

FROM SALES. [Order] AS SO

INNER JOIN Sales.[OrderDetail] AS OD

ON SO.ORDERID = OD.OrderId

INNER JOIN Sales.[CUSTOMER] AS SC

ON SO.CUSTOMERID = SC.CUSTOMERID

WHERE SO.ORDERID = @order\_number

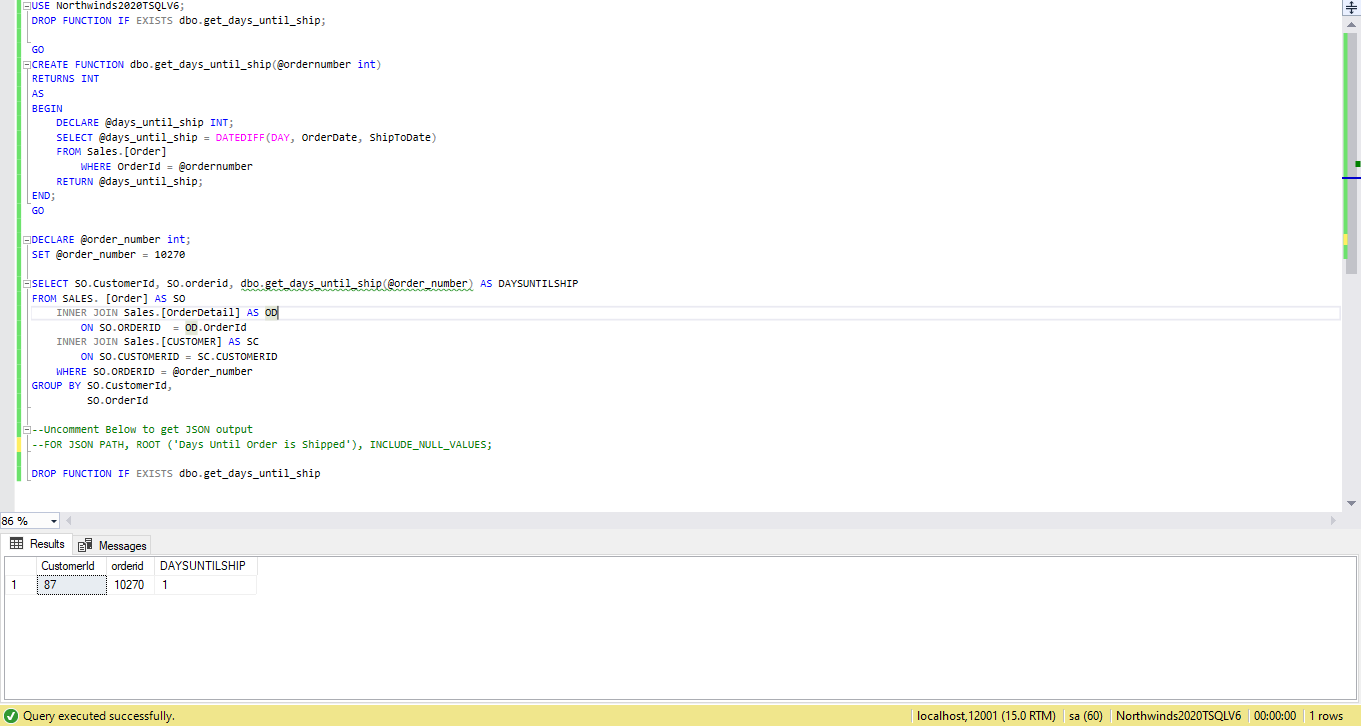
GROUP BY SO.CustomerId,

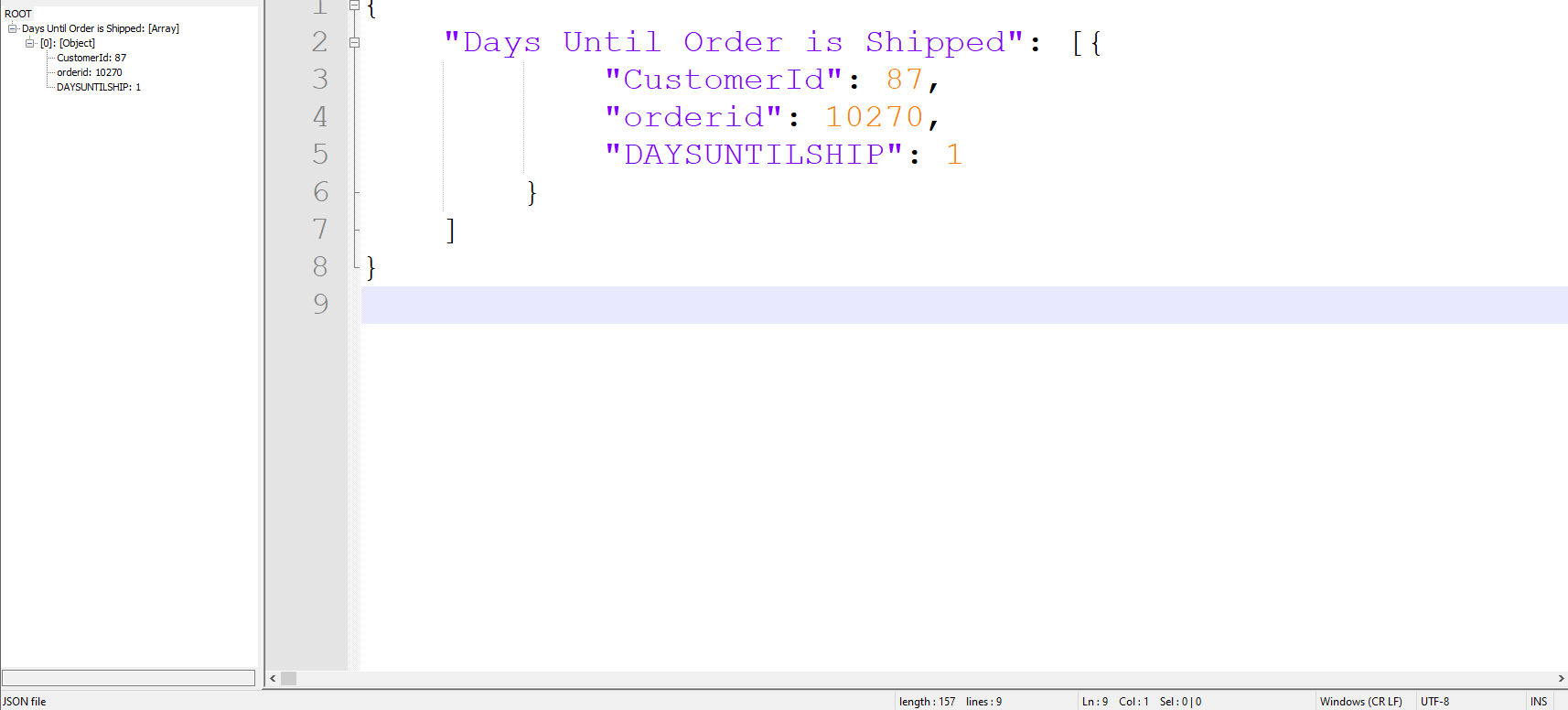
SO.OrderId

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Days Until Order is Shipped'), INCLUDE\_NULL\_VALUES;

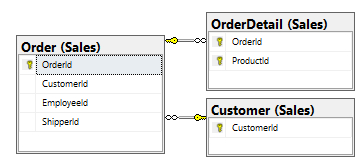
DROP FUNCTION IF EXISTS dbo.get\_days\_until\_ship

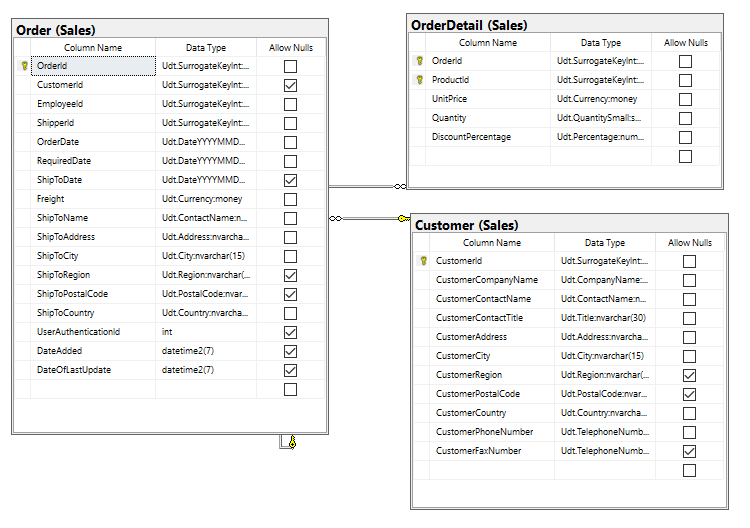




Proposition #15 (Complex)

Write a function that shows the total quantity going to each region of specified country





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | ShipToRegion |
| Customer | CustomerId |
| OrderDetail | Quantity, ShipToRegion |

**Order By**

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| OrderDetail | ShipToRegion | ASC |

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.UnitsToCountry;

GO

CREATE FUNCTION dbo.UnitsToCountry(@Country NVARCHAR(50))

RETURNS TABLE

AS

RETURN SELECT SUM(SOD.Quantity) AS TotalUnits, SO.ShipToRegion

FROM SALES.[ORDER] AS SO

LEFT JOIN SALES.[ORDERDETAIL] AS SOD

ON SO.ORDERID = SOD.ORDERID

LEFT JOIN SALES.[CUSTOMER] AS SC

ON SO.CustomerId = SC.CustomerId

GROUP BY SO.ShipToRegion, SO.ShipToCountry

HAVING SO.ShipToCountry = @Country;

GO

DECLARE @COUNTRY nvarchar(50)

SET @country = 'USA'

SELECT C.ShipToRegion, C.TotalUnits

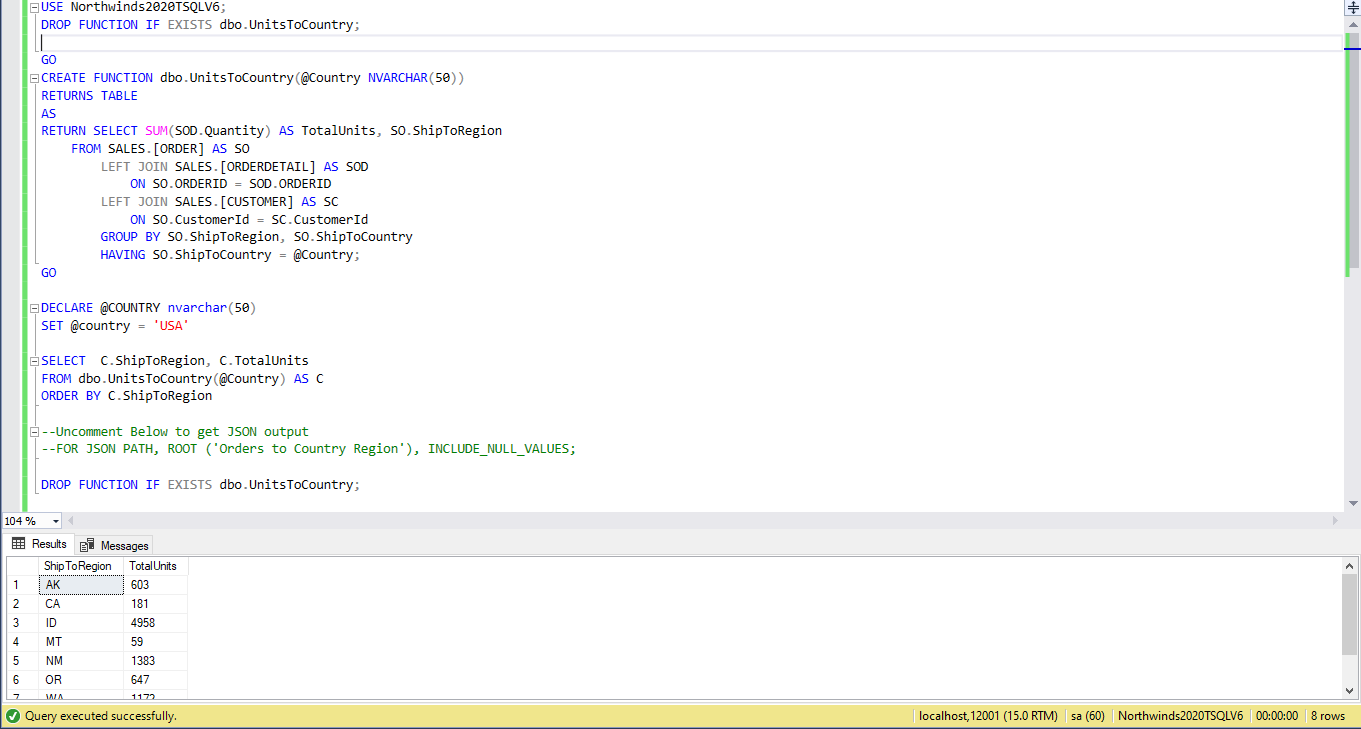
FROM dbo.UnitsToCountry(@Country) AS C

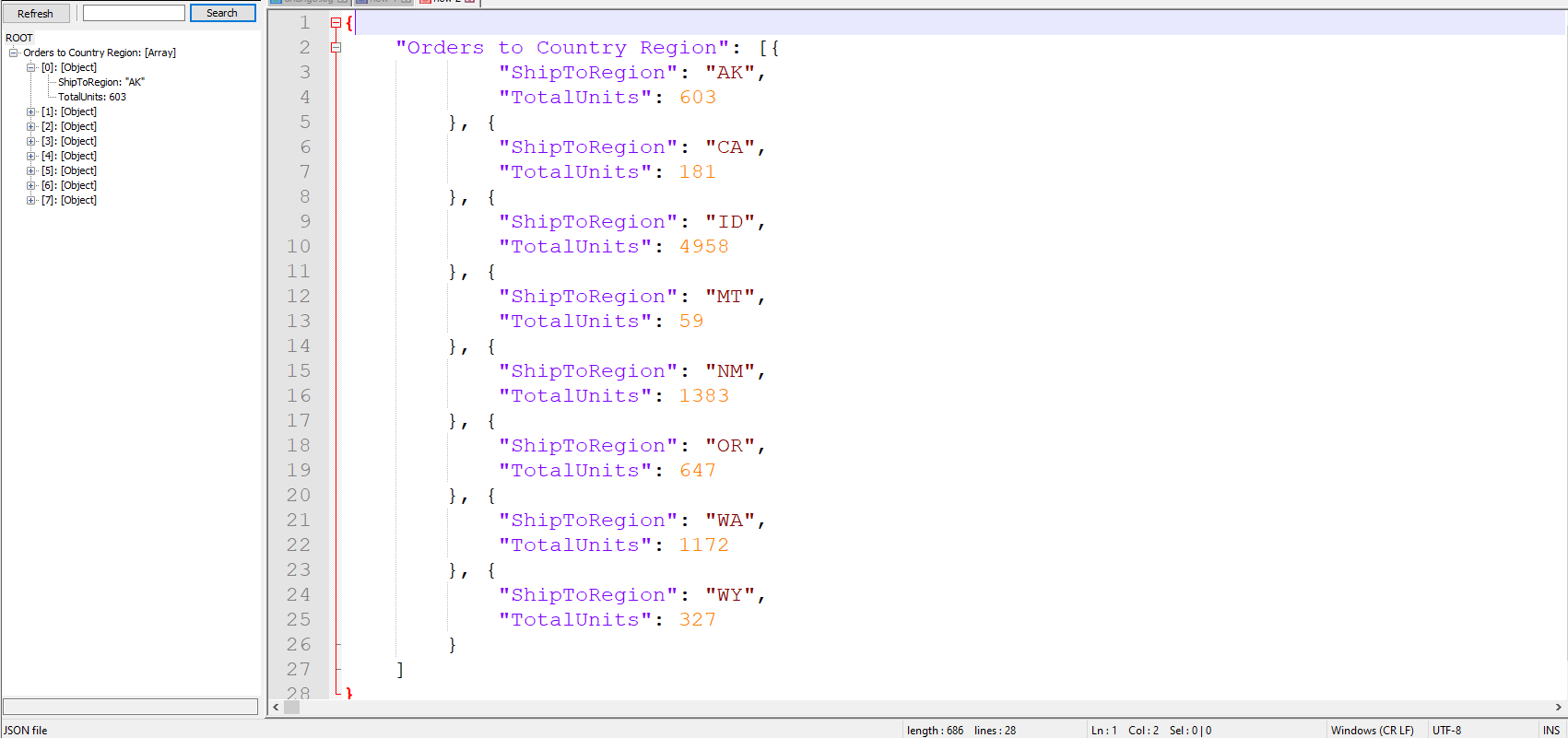
ORDER BY C.ShipToRegion

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Orders to Country Region'), INCLUDE\_NULL\_VALUES;

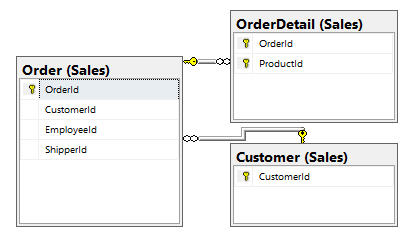
DROP FUNCTION IF EXISTS dbo.UnitsToCountry;

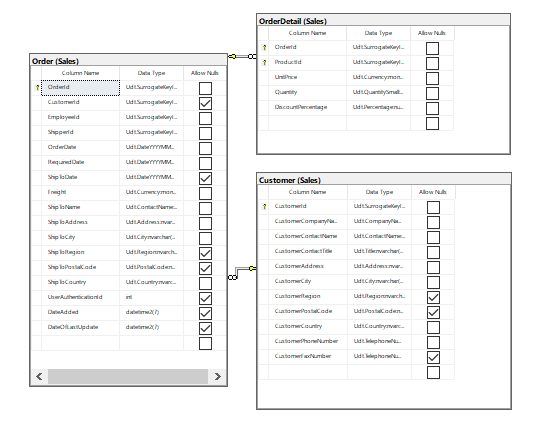




Proposition #16 (Complex)

Write a function that shows the total quantity going to each region of specified country





**Columns from Tables**

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

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.UnitsToCity;

GO

CREATE FUNCTION dbo.UnitsToCity

(

@City NVARCHAR(50)

)

RETURNS TABLE

AS

RETURN SELECT SO.OrderId,

SO.CustomerId,

SO.ShipToCity

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS SO

ON C.CustomerId = SO.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = SO.OrderId

WHERE SO.ShipToCity = @City

GROUP BY SO.OrderId,

SO.CustomerId,

SO.ShipToCity;

GO

DECLARE @City NVARCHAR(50);

SET @City = N'London';

SELECT OrderId,

CustomerId,

ShipToCity

FROM dbo.UnitsToCity(@City)

GROUP BY OrderId,

CustomerId,

ShipToCity

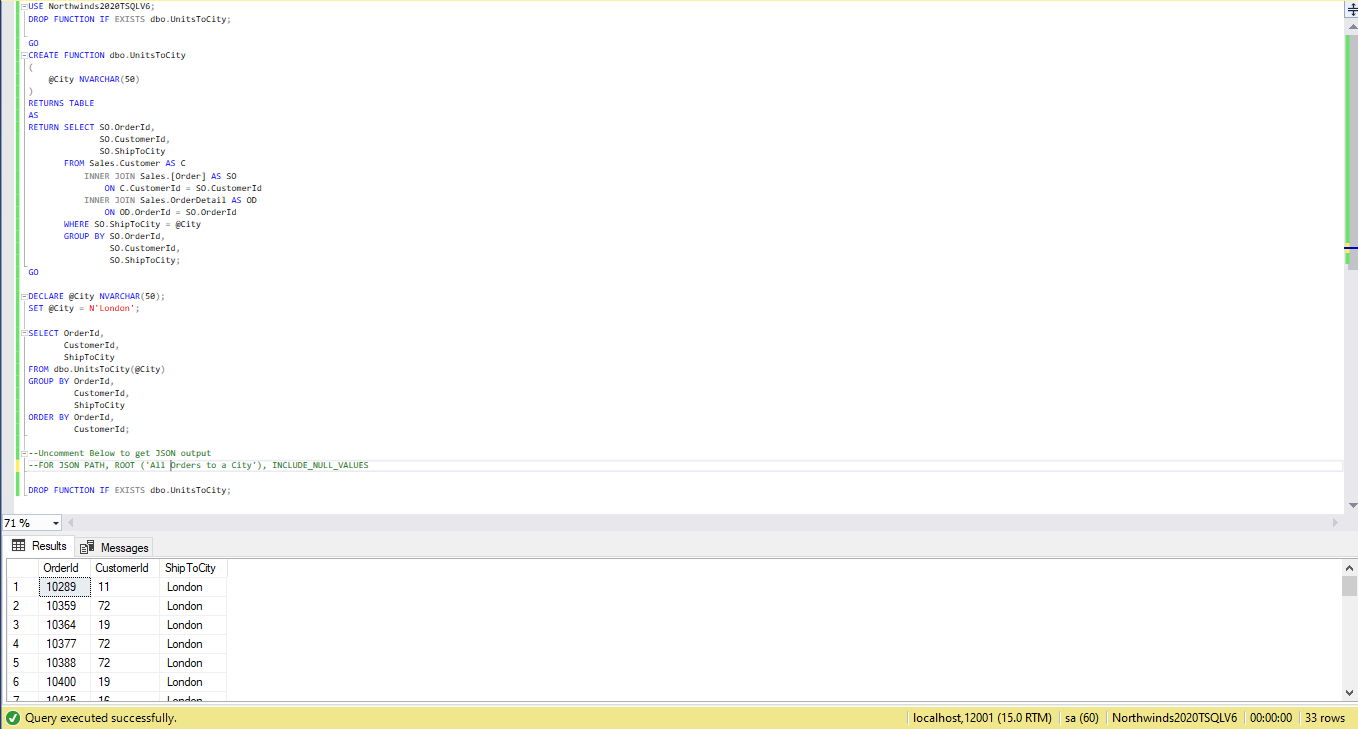
ORDER BY OrderId,

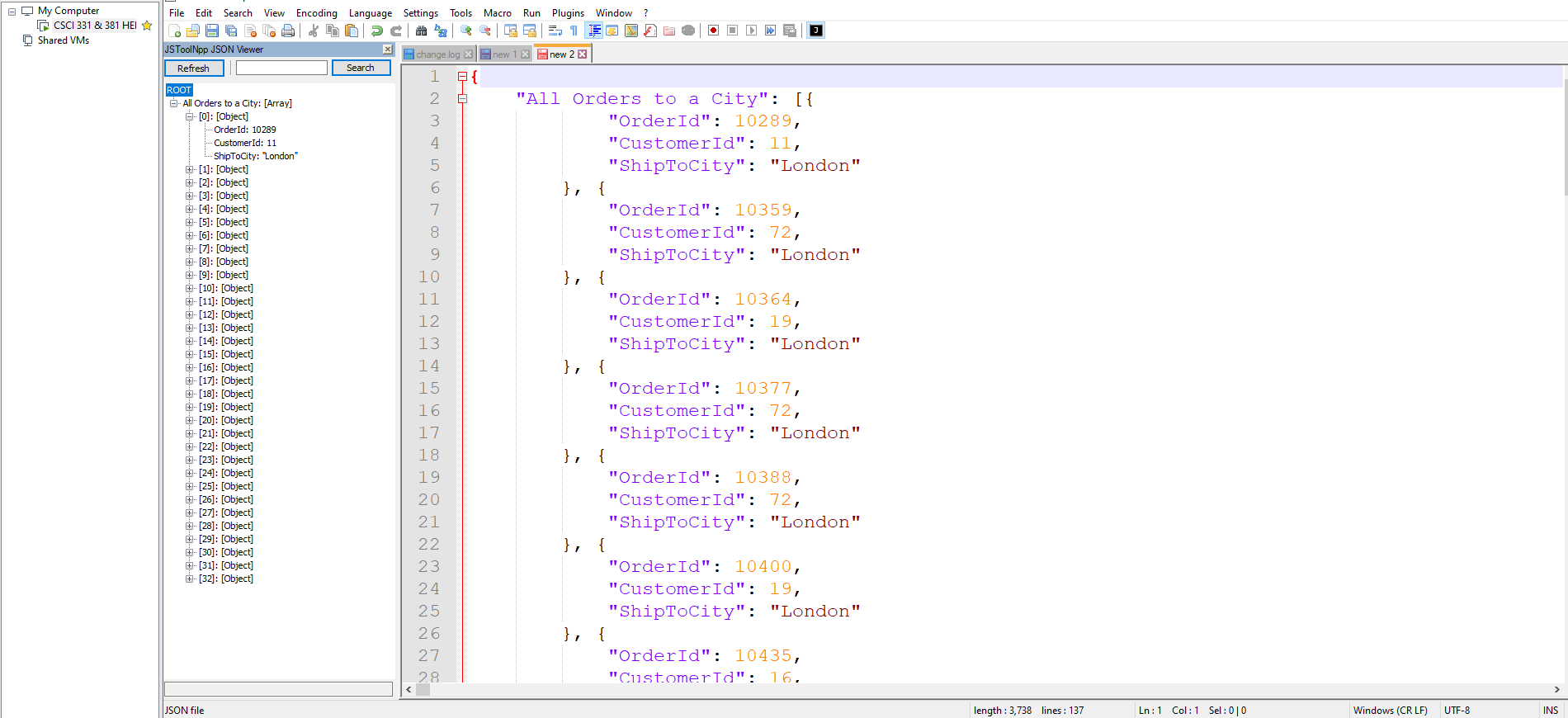
CustomerId

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('All Orders to a City'), INCLUDE\_NULL\_VALUES

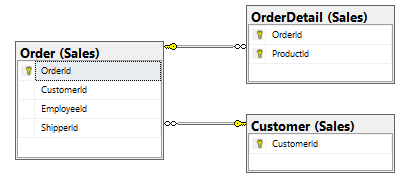
DROP FUNCTION IF EXISTS dbo.UnitsToCity;

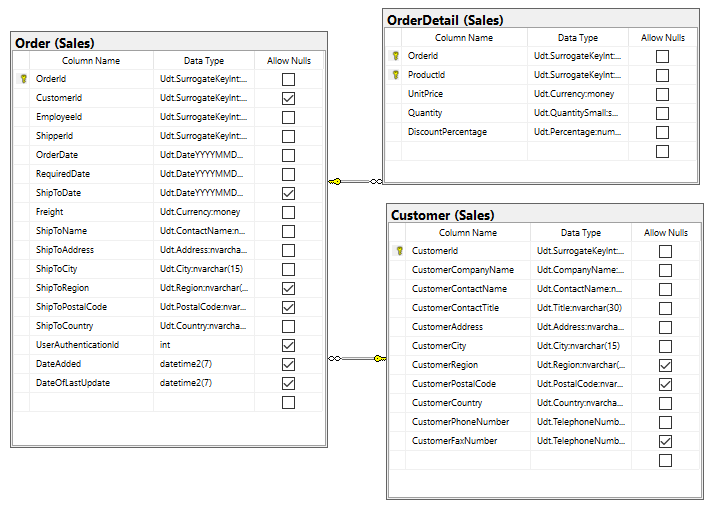




Proposition #17 (Complex)

Show the Total Units an Employee has sold to Each Customer





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | EmployeeId |
| Customer | CustomerId |
| OrderDetail | Quantity |

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.EmployeeSaleAmountPerCustomer;

GO

CREATE FUNCTION dbo.EmployeeSaleAmountPerCustomer

(

@EmployeeID INT

)

RETURNS TABLE

AS

RETURN SELECT SUM(OD.Quantity) AS NumSold,

C.CustomerId,

O.EmployeeId

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

WHERE O.EmployeeId = @EmployeeID

GROUP BY C.CustomerId,

O.EmployeeId;

GO

DECLARE @EmployeeId INT;

SET @EmployeeId = 1;

SELECT EmployeeId,

CustomerId,

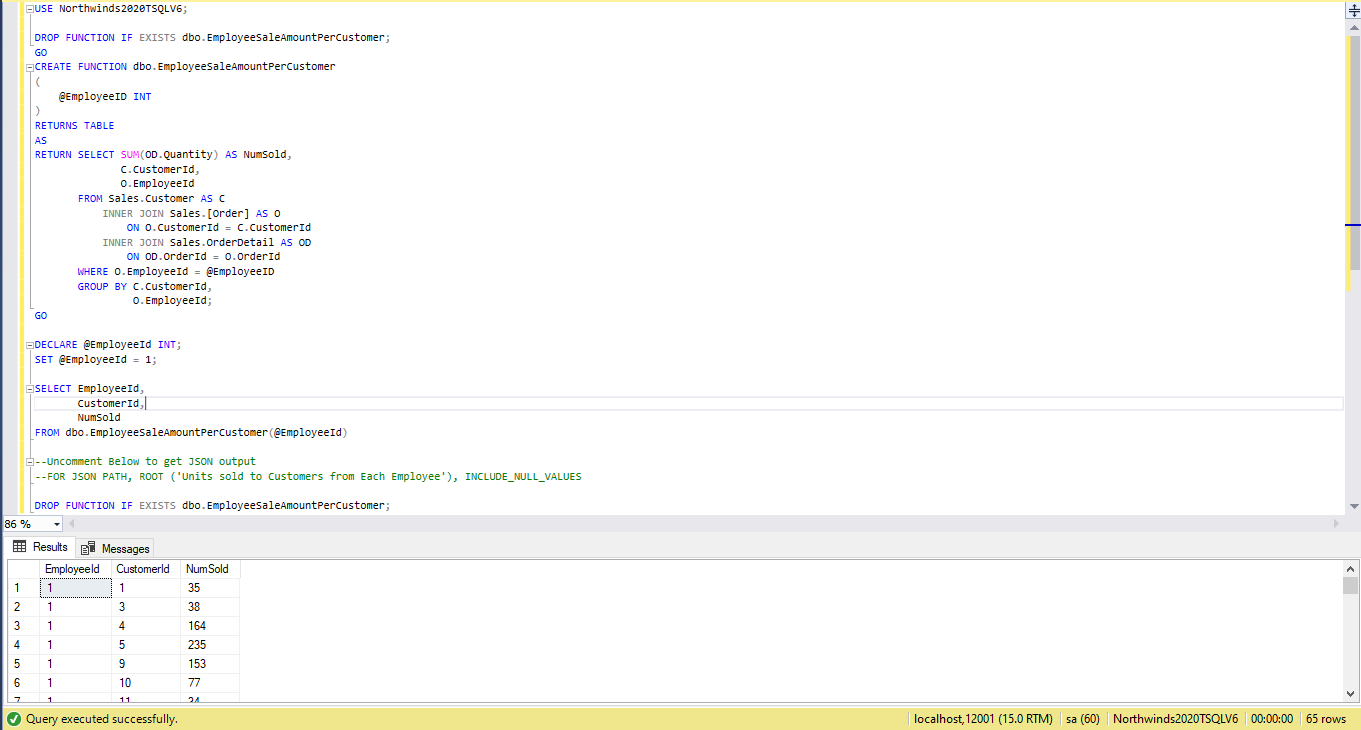
NumSold

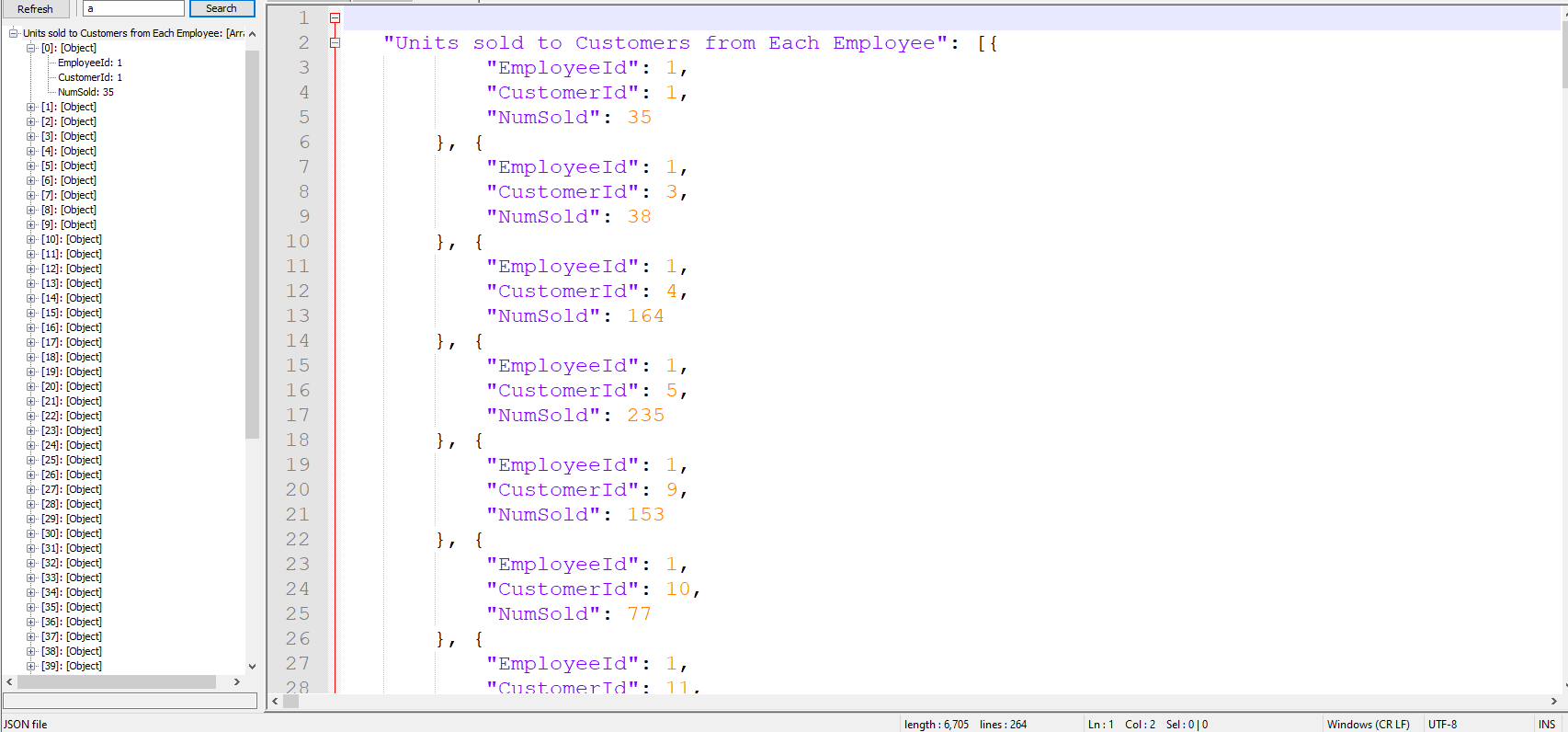
FROM dbo.EmployeeSaleAmountPerCustomer(@EmployeeId)

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Units sold to Customers from Each Employee'), INCLUDE\_NULL\_VALUES

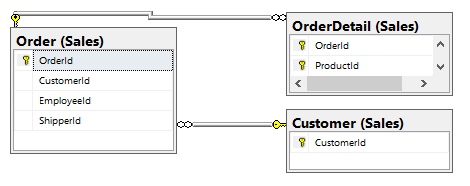
DROP FUNCTION IF EXISTS dbo.EmployeeSaleAmountPerCustomer;

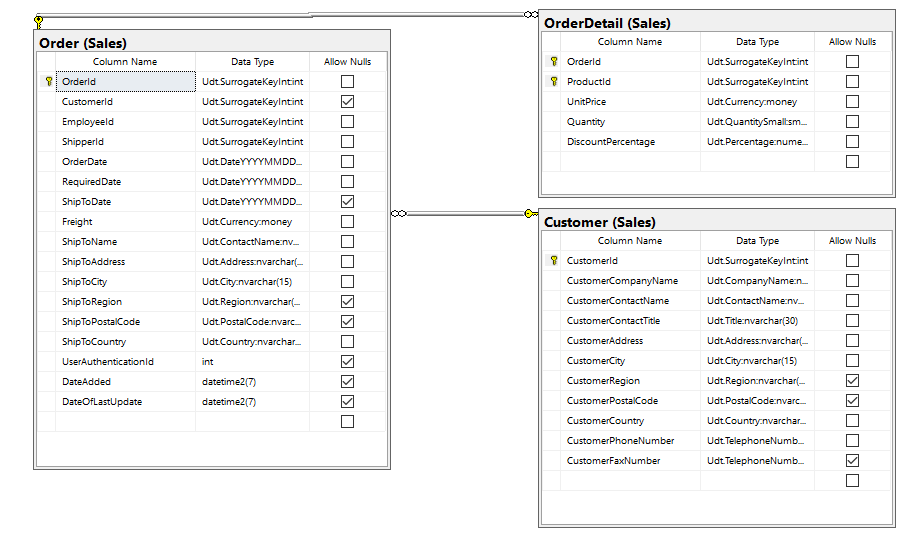




Proposition #18 (Complex)

Show the Average Discount to a Specific Customer





**Columns from Tables**

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

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.AvgerageCustomerDiscount;

GO

CREATE FUNCTION dbo.AvgerageCustomerDiscount

(

@CustomerID INT

)

RETURNS TABLE

AS

RETURN SELECT C.CustomerId,

AVG(OD.DiscountPercentage) AS AverageDiscountPercent

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

WHERE O.CustomerId = @CustomerID

GROUP BY C.CustomerId;

GO

DECLARE @EmployeeId INT;

SET @EmployeeId = 1;

SELECT CustomerId,

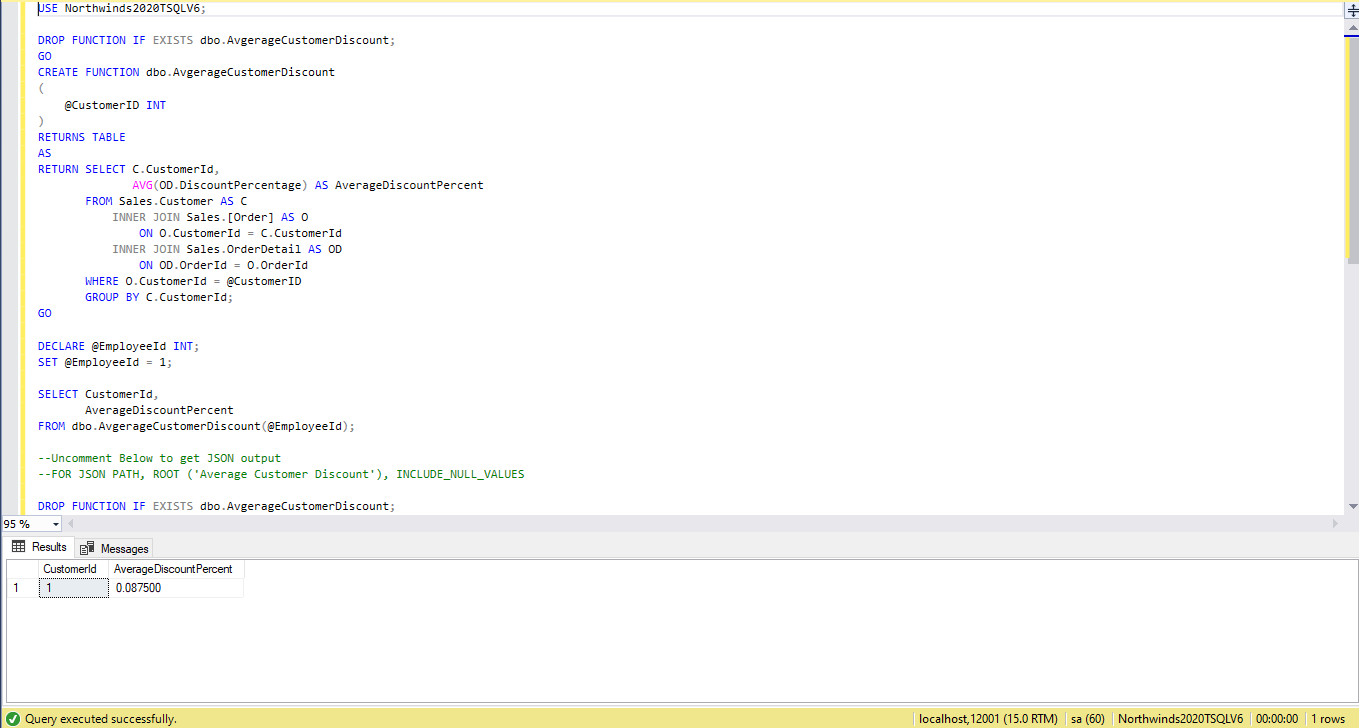
AverageDiscountPercent

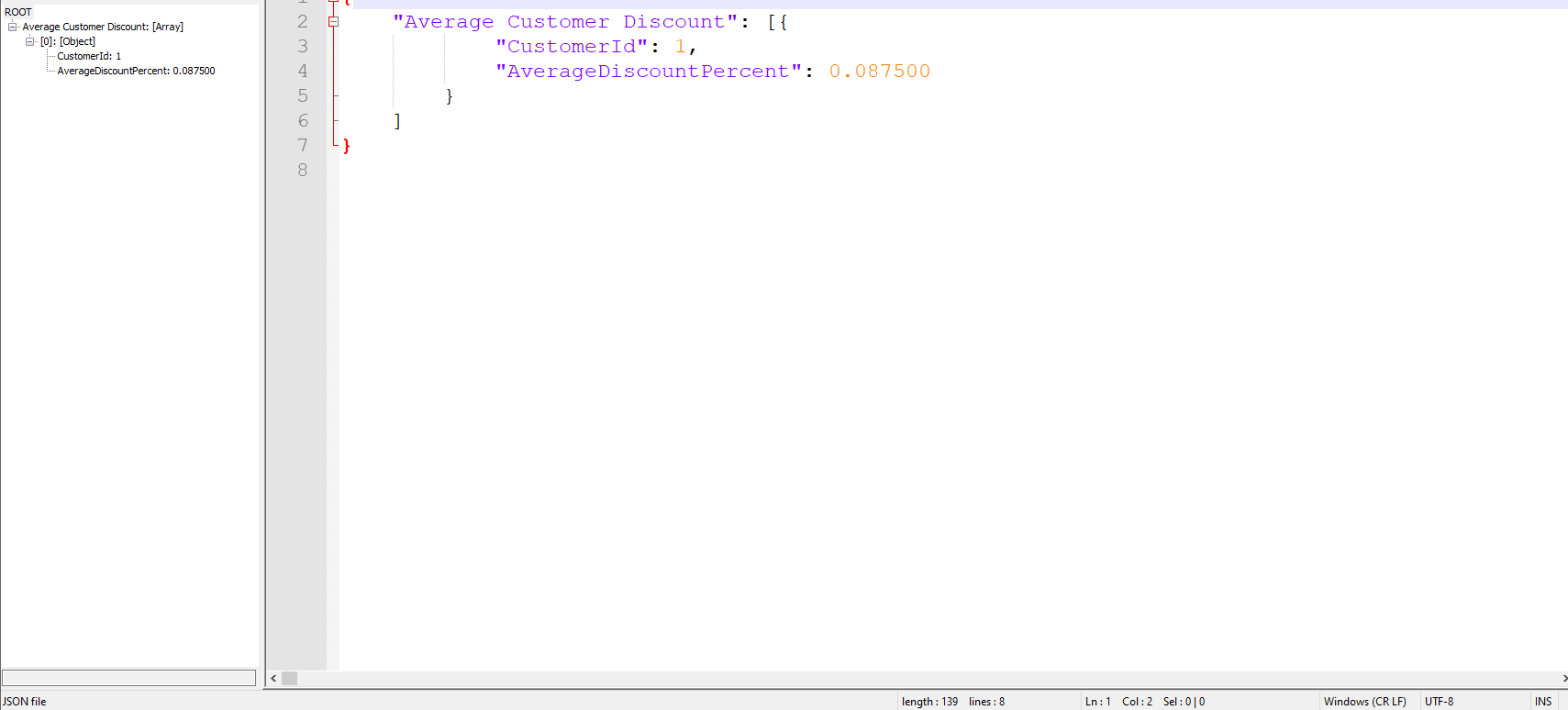
FROM dbo.AvgerageCustomerDiscount(@EmployeeId)

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Average Customer Discount'), INCLUDE\_NULL\_VALUES

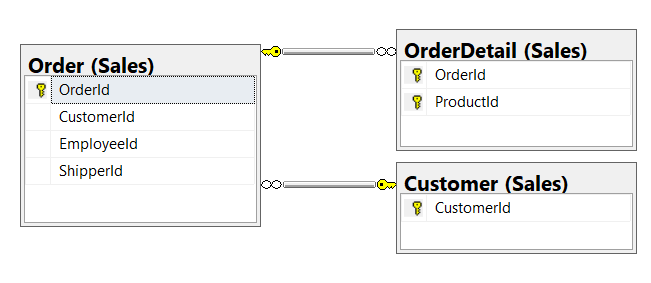
DROP FUNCTION IF EXISTS dbo.AvgerageCustomerDiscount;

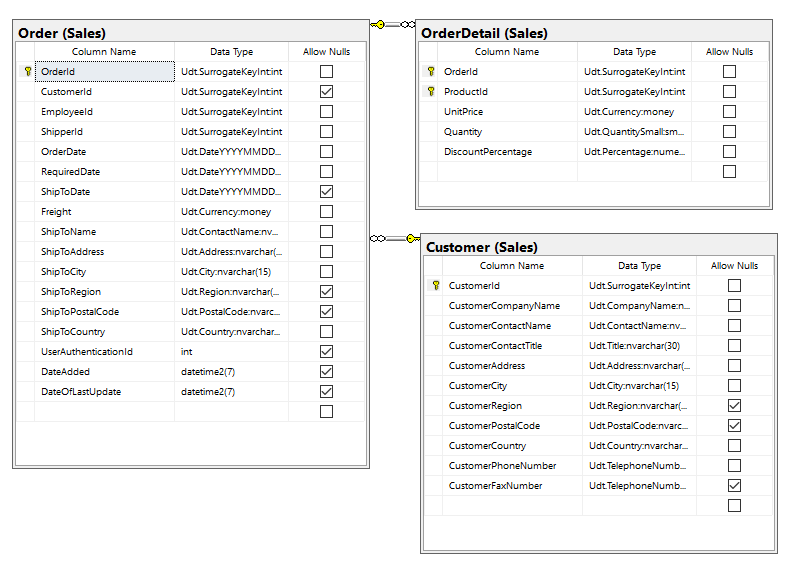




Proposition #19 (Complex)

Return the Customers Favorite Item (Defined by Largest Quantity Ordered)





**Columns from Tables**

|  |  |
| --- | --- |
| Table Name | Column Name |
| Order | OrderId |
| Customer | CustomerId |
| OrderDetail | Quantity, ProductId |

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.CustomersFavoriteItem;

GO

CREATE FUNCTION dbo.CustomersFavoriteItem

(

@CustomerID INT

)

RETURNS TABLE

AS

RETURN SELECT TOP (1)

C.CustomerId,

OD.Quantity,

OD.ProductId

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

WHERE O.CustomerId = @CustomerID

GROUP BY C.CustomerId,

OD.Quantity,

OD.ProductId

ORDER BY OD.Quantity DESC;

GO

DECLARE @CustomerId INT;

SET @CustomerId = 1;

SELECT CustomerId,

ProductId,

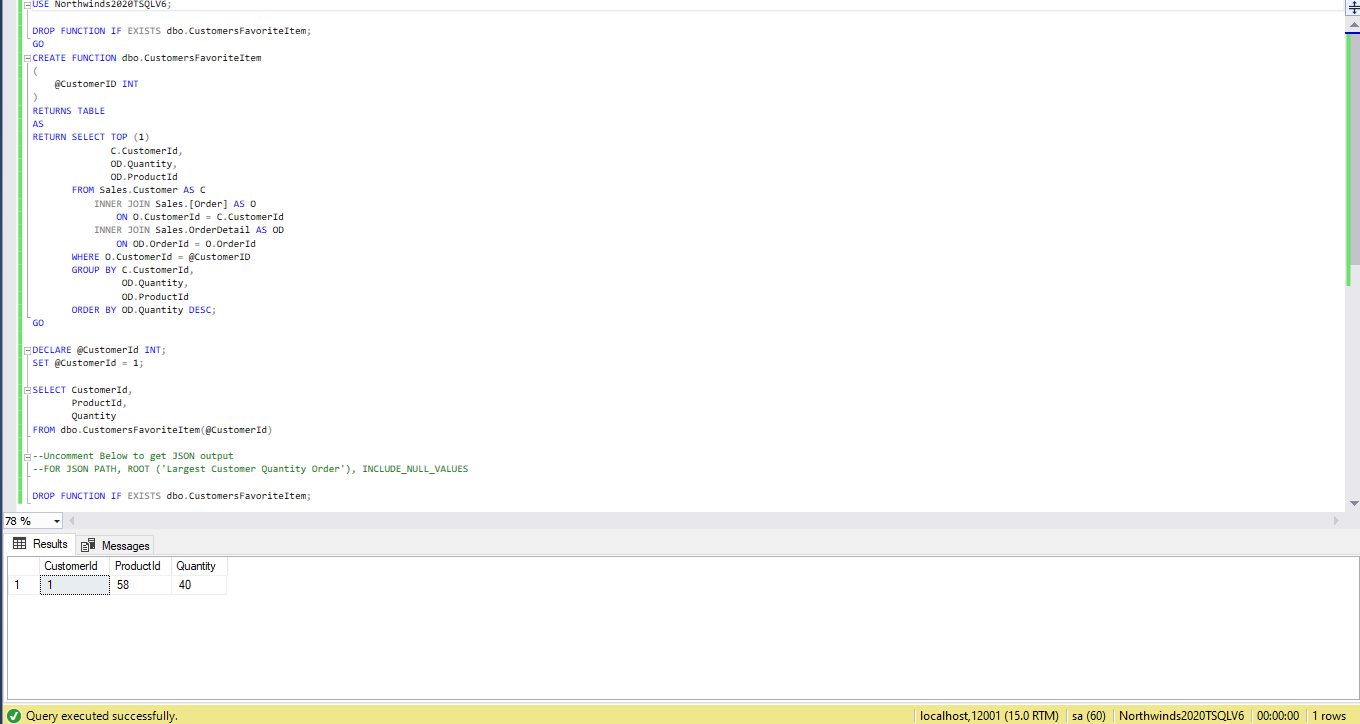
Quantity

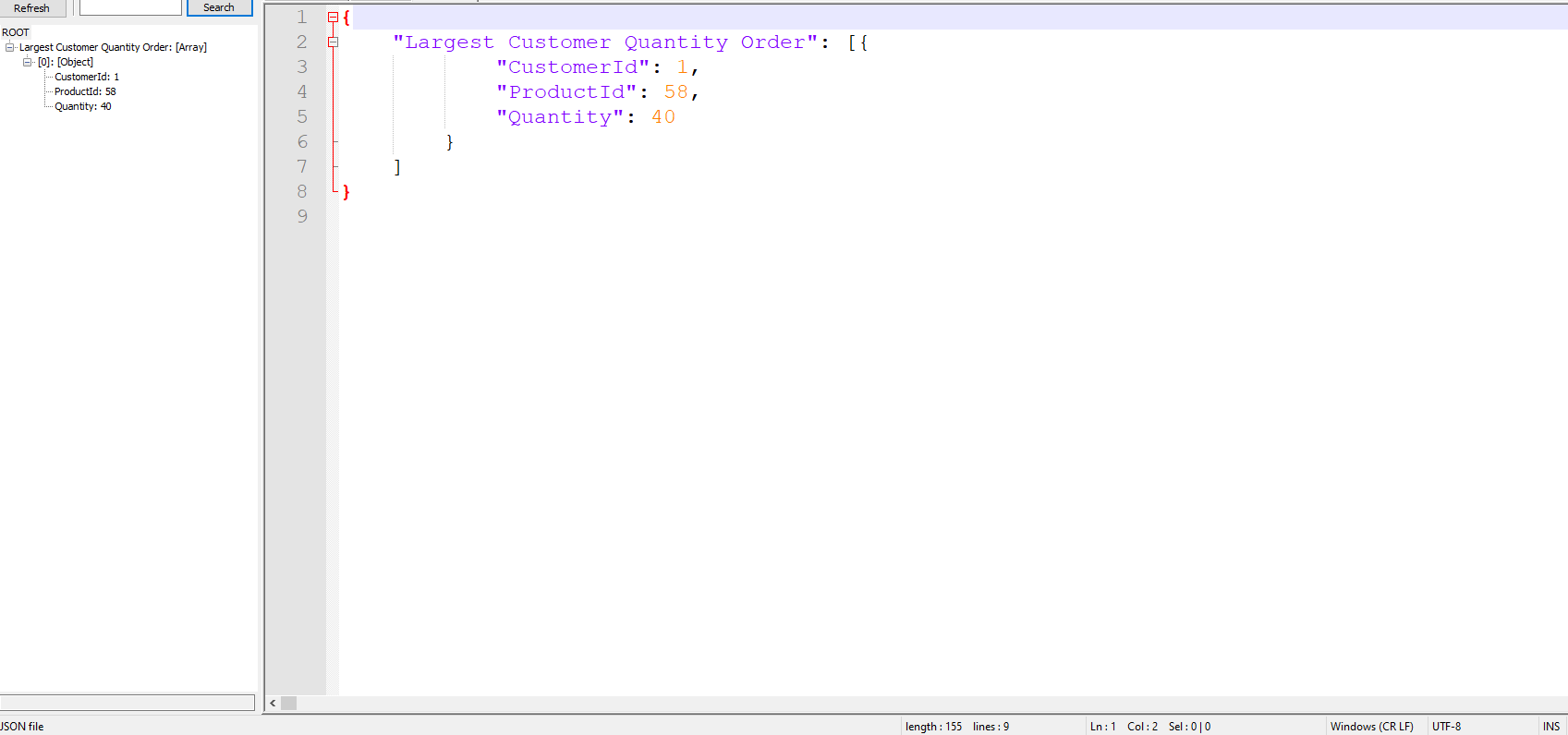
FROM dbo.CustomersFavoriteItem(@CustomerId

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Largest Customer Quantity Order'), INCLUDE\_NULL\_VALUES

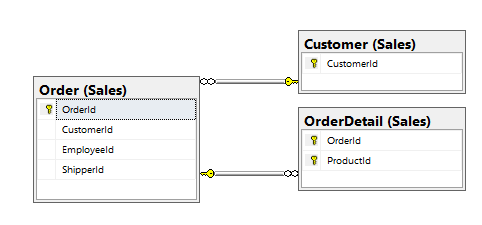
DROP FUNCTION IF EXISTS dbo.CustomersFavoriteItem;

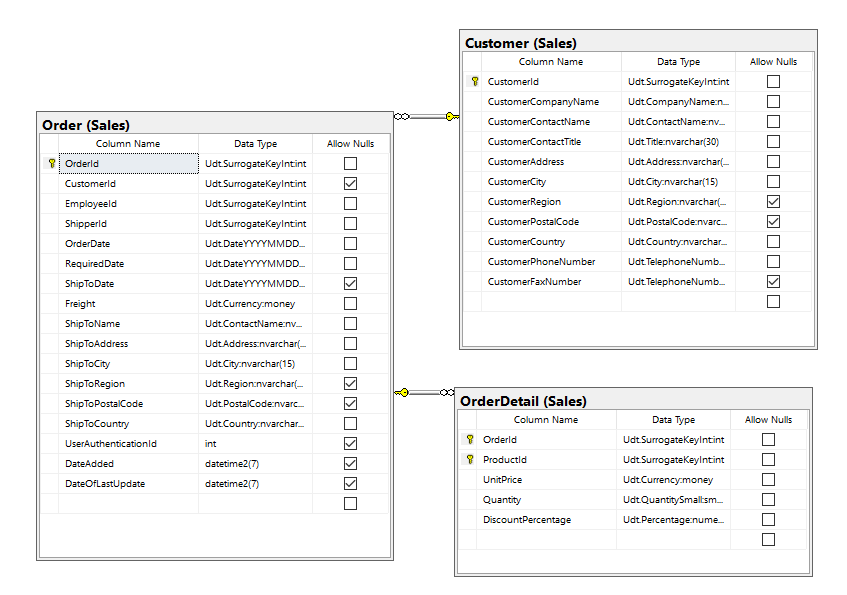




Proposition #20 (Complex)

Return the Customers Favorite Item (Defined by Largest Quantity Ordered)





**Columns from Tables**

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

**Order By**

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

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.EmployeesFavoriteCustomer;

GO

CREATE FUNCTION dbo.EmployeesFavoriteCustomer

(

@EmployeeId INT

)

RETURNS TABLE

AS

RETURN SELECT TOP (1)

O.EmployeeId,

COUNT(C.CustomerId) AS CustomerCount,

C.CustomerId

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

WHERE O.EmployeeId = @EmployeeId

GROUP BY O.EmployeeId,

C.CustomerId

ORDER BY COUNT(C.CustomerId) DESC;

GO

DECLARE @EmployeeId INT;

SET @EmployeeId = 1;

SELECT EmployeeId,

CustomerId,

CustomerCount

FROM dbo.EmployeesFavoriteCustomer(@EmployeeId)

--Uncomment Below to get JSON output

--FOR JSON PATH, ROOT ('Most Orders between an Employee and Customer'), INCLUDE\_NULL\_VALUES

DROP FUNCTION IF EXISTS dbo.EmployeesFavoriteCustomer;



