Group 8 Project 1 STEVEN NECOLA

Learning how to create diagrams in a database as a navigation tool. Creating diagram views as subject areas that isolates various sub-systems for querying information. Using the subject areas to solve problems for the business and document the necessary information needed to provide the appropriate query resolution. Writing 20 queries by each group member across the five databases identified.

Developing the soft skills needed in the business work environment such as teamwork, documentation and creating workflows.

Learning the structure of a newly acquired database systems by your company without any documentation.

Each Problem should follow the format in Problem 01.

Contents

[Proposition 01: Find the last customer to order from USA using Northwinds2020TSQLV6 [Simple] 7](#_Toc67945645)

[Standard View: 7](#_Toc67945646)

[Key View: 7](#_Toc67945647)

[Select By Columns: 7](#_Toc67945648)

[Order By: 8](#_Toc67945649)

[Problem Solving Query: 8](#_Toc67945650)

[Results: 1 rows 8](#_Toc67945651)

[JSON Output: 8](#_Toc67945652)

[Proposition 02: Find Supplier company information for every product using Northwinds2020TSQLV6 [Simple] 9](#_Toc67945653)

[Standard View: 9](#_Toc67945654)

[Key View: 9](#_Toc67945655)

[Select By Columns: 9](#_Toc67945656)

[Order By: 9](#_Toc67945657)

[Problem Solving Query: 9](#_Toc67945658)

[Results: 77 rows 10](#_Toc67945659)

[JSON Output: 11](#_Toc67945660)

[Proposition 03: Get all orders placed by customers in the USA in 2015. [Simple] 12](#_Toc67945661)

[Standard View: 12](#_Toc67945662)

[Key View: 12](#_Toc67945663)

[Select By Columns: 12](#_Toc67945664)

[Order By: 12](#_Toc67945665)

[Problem Solving Query: 13](#_Toc67945666)

[Results: 60 rows 13](#_Toc67945667)

[JSON Output: 14](#_Toc67945668)

[Proposition 04: Return the Sales person freight and bonus for every sales person using AdventureWorks2017 [Simple] 15](#_Toc67945669)

[Standard View: 15](#_Toc67945670)

[Key View: 16](#_Toc67945671)

[Select By Columns: 16](#_Toc67945672)

[Order By: 16](#_Toc67945673)

[Problem Solving Query: 16](#_Toc67945674)

[Results: 17 Rows 17](#_Toc67945675)

[JSON Output: 17](#_Toc67945676)

[Proposition 05: Find the name of every Employee in the Sales department using AdventureWorks2017 [Simple] 18](#_Toc67945677)

[Standard View: 18](#_Toc67945678)

[Key View: 18](#_Toc67945679)

[Select By Columns: 18](#_Toc67945680)

[Order By: 18](#_Toc67945681)

[Problem Solving Query: 19](#_Toc67945682)

[Results: 17 rows 19](#_Toc67945683)

[JSON Output: 20](#_Toc67945684)

[Proposition 06: Find the Customer name, Employee Last Name, First Name , Shipper Company Name for every order using Northwinds2020TSQLV6 [Medium] 21](#_Toc67945685)

[Standard View: 21](#_Toc67945686)

[Key View: 21](#_Toc67945687)

[Select By Columns: 21](#_Toc67945688)

[Order By: 22](#_Toc67945689)

[Problem Solving Query: 22](#_Toc67945690)

[Results: 830 rows 22](#_Toc67945691)

[JSON Output: 23](#_Toc67945692)

[Proposition 07: Find the 10 longest shipping times it took to ship each order, the shipping company name, phone number, and city using Northwinds2020TSQLV6 [Medium] 24](#_Toc67945693)

[Standard View: 24](#_Toc67945694)

[Key View: 24](#_Toc67945695)

[Select By Columns: 24](#_Toc67945696)

[Order By: 25](#_Toc67945697)

[Problem Solving Query: 25](#_Toc67945698)

[Results: 10 rows 25](#_Toc67945699)

[JSON Output: 26](#_Toc67945700)

[Proposition 08: List the order details of every order from the Brazil using Northwinds2020TSQLV6 [Medium] 27](#_Toc67945701)

[Standard View: 27](#_Toc67945702)

[Key View: 27](#_Toc67945703)

[Select By Columns: 27](#_Toc67945704)

[Order By: 27](#_Toc67945705)

[Problem Solving Query: 27](#_Toc67945706)

[Results: 203 rows 28](#_Toc67945707)

[JSON Output: 29](#_Toc67945708)

[Proposition 09: Show the top count of sales made by employees using Northwinds2020TSQLV6 [Medium] 30](#_Toc67945709)

[Standard View: 30](#_Toc67945710)

[Key View: 30](#_Toc67945711)

[Select By Columns: 30](#_Toc67945712)

[Order By: 31](#_Toc67945713)

[Problem Solving Query: 31](#_Toc67945714)

[Results: 9 rows 31](#_Toc67945715)

[JSON Output: 32](#_Toc67945716)

[Proposition 10: Return the total quantity for each employee and year using Northwinds2020TSQLV6 [Medium] 32](#_Toc67945717)

[Standard View: 32](#_Toc67945718)

[Key View: 33](#_Toc67945719)

[Select By Columns: 33](#_Toc67945720)

[Order By: 33](#_Toc67945721)

[Problem Solving Query: 33](#_Toc67945722)

[Results: 27 rows 34](#_Toc67945723)

[JSON Output: 35](#_Toc67945724)

[Proposition 11: Count how many employees are single or married using AdventureWorks2017 [Medium] 36](#_Toc67945725)

[Standard View: 36](#_Toc67945726)

[Key View: 36](#_Toc67945727)

[Select By Columns: 36](#_Toc67945728)

[Order By: 36](#_Toc67945729)

[Problem Solving Query: 37](#_Toc67945730)

[Results: 2 rows 37](#_Toc67945731)

[JSON Output: 37](#_Toc67945732)

[Proposition 12: Find employees who are also customers using AdventureWorksDW2017. 38](#_Toc67945733)

[Standard View: 38](#_Toc67945734)

[Key View: 39](#_Toc67945735)

[Select By Columns: 39](#_Toc67945736)

[Order By: 39](#_Toc67945737)

[Problem Solving Query: 39](#_Toc67945738)

[Results: 24 rows 40](#_Toc67945739)

[JSON Output: 41](#_Toc67945740)

[Proposition 13: Find the average income of customers grouped by their education using AdventureWorksDW2017 [Medium] 42](#_Toc67945741)

[Standard View: 42](#_Toc67945742)

[Key View: 43](#_Toc67945743)

[Select By Columns: 43](#_Toc67945744)

[Order By: 43](#_Toc67945745)

[Problem Solving Query: 43](#_Toc67945746)

[Results: 5 rows 43](#_Toc67945747)

[JSON Output: 44](#_Toc67945748)

[Proposition 14: Show the top count of sales made by employees and their total sales amount using Northwinds2020TSQLV6 [Complex] 44](#_Toc67945749)

[Standard View: 44](#_Toc67945750)

[Key View: 45](#_Toc67945751)

[Select By Columns: 45](#_Toc67945752)

[Order By: 45](#_Toc67945753)

[Problem Solving Query: 45](#_Toc67945754)

[Results: 9 rows 46](#_Toc67945755)

[JSON Output: 47](#_Toc67945756)

[Proposition 15: Find the top selling Products in AdventureWorks2017 [Complex] 48](#_Toc67945757)

[Standard View: 48](#_Toc67945758)

[Key View: 49](#_Toc67945759)

[Select By Columns: 49](#_Toc67945760)

[Order By: 49](#_Toc67945761)

[Problem Solving Query: 49](#_Toc67945762)

[Results: 266 rows 50](#_Toc67945763)

[JSON Output: 51](#_Toc67945764)

[Proposition 16: Find out how many customers have children but do not have them at home anymore using AdventureWorksDW2017 [Complex] 52](#_Toc67945765)

[Standard View: 52](#_Toc67945766)

[Key View: 53](#_Toc67945767)

[Select By Columns: 53](#_Toc67945768)

[Order By: 53](#_Toc67945769)

[Problem Solving Query: 53](#_Toc67945770)

[Results: 9,074 rows 54](#_Toc67945771)

[JSON Output: 55](#_Toc67945772)

[Proposition 17: Find the AVG yearly income of employees grouped by department using AdventureWorksDW2017 [Complex] 56](#_Toc67945773)

[Standard View: 56](#_Toc67945774)

[Key View: 57](#_Toc67945775)

[Select By Columns: 57](#_Toc67945776)

[Order By: 57](#_Toc67945777)

[Problem Solving Query: 57](#_Toc67945778)

[Results: 16 rows 58](#_Toc67945779)

[JSON Output: 59](#_Toc67945780)

[Proposition 18: Calculate expected years until retirement for employees using AdventureWorksDW2017 [Complex] 60](#_Toc67945781)

[Standard View: 60](#_Toc67945782)

[Key View: 61](#_Toc67945783)

[Select By Columns: 61](#_Toc67945784)

[Order By: 61](#_Toc67945785)

[Problem Solving Query: 61](#_Toc67945786)

[Results: 277 rows 62](#_Toc67945787)

[JSON Output: 63](#_Toc67945788)

[Proposition 19: Employee of the year based on total sales using Northwinds2020TSQLV6 [Complex] 64](#_Toc67945789)

[Standard View: 64](#_Toc67945790)

[Key View: 64](#_Toc67945791)

[Select By Columns: 64](#_Toc67945792)

[Order By: 64](#_Toc67945793)

[Problem Solving Query: 65](#_Toc67945794)

[Results: 104 rows 66](#_Toc67945795)

[JSON Output: 67](#_Toc67945796)

[Proposition 20: Create a location profile for each employee using Northwinds2020TSQLV6. [Complex] 68](#_Toc67945797)

[Standard View: 68](#_Toc67945798)

[Key View: 68](#_Toc67945799)

[Select By Columns: 68](#_Toc67945800)

[Order By: 69](#_Toc67945801)

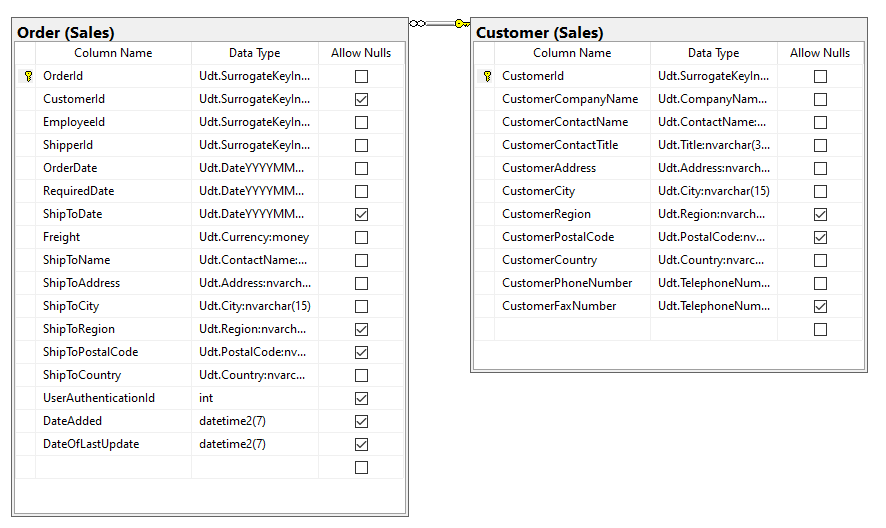
[Problem Solving Query: 69](#_Toc67945802)

[Results: 9 rows 70](#_Toc67945803)

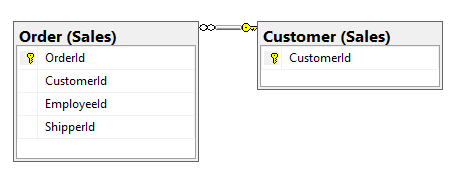
[JSON Output: 70](#_Toc67945804)

# Proposition 01: Find the last customer to order from USA using Northwinds2020TSQLV6 [Simple]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table | Columns |
| Sales.Order | OrderDate |
| Sales.Customer | CustomerId, CustomerContactName, CustomerCompanyName, CustomerAddress, CustomerCity, CustomerRegion, CustomerCountry |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | OrderDate | DESC |

## Problem Solving Query:

USE [Northwinds2020TSQLV6];

GO

SELECT TOP (1)

C.CustomerId,

O.OrderDate,

C.CustomerContactName,

C.CustomerCompanyName,

C.CustomerAddress,

C.CustomerCity,

C.CustomerRegion,

C.CustomerCountry

FROM Sales.[Order] AS O

INNER JOIN Sales.Customer AS C

ON C.CustomerId = O.CustomerId

WHERE C.CustomerCountry LIKE N'USA'

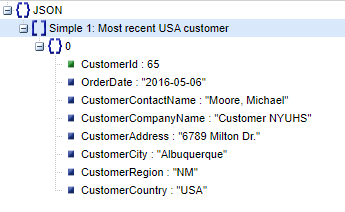
ORDER BY O.OrderDate DESC

--FOR JSON PATH, ROOT('Simple 1: Most recent USA customer'), INCLUDE\_NULL\_VALUES;

## Results: 1 rows

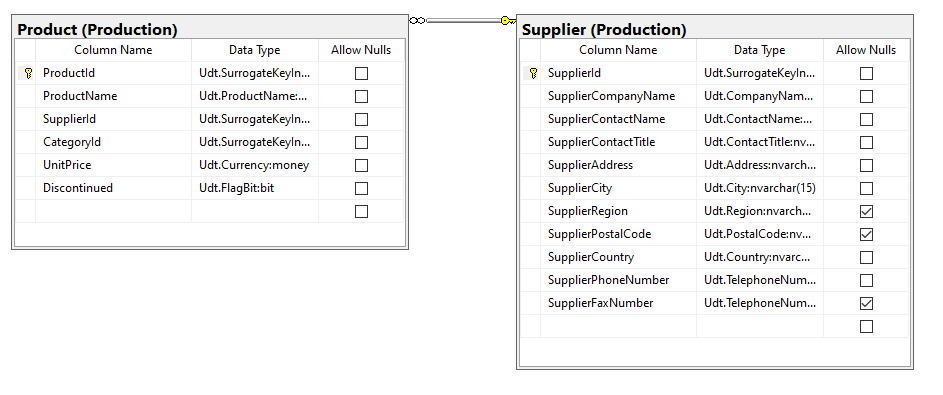


## JSON Output:

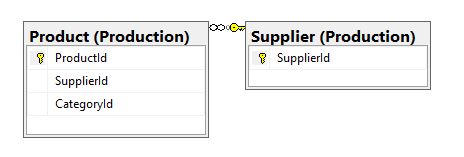


# Proposition 02: Find Supplier company information for every product using Northwinds2020TSQLV6 [Simple]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Production.Product | ProductId, ProductName |
| Production.Supplier | SupplierId, SupplierCompanyName, SupplierContactName |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Production.Product | P.ProductId | ASC |

## Problem Solving Query:

USE [Northwinds2020TSQLV6];

GO

SELECT P.ProductId,

P.ProductName,

P.SupplierId,

S.SupplierCompanyName,

S.SupplierContactName

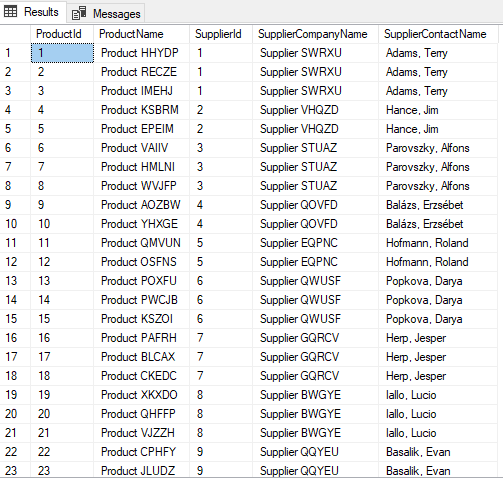
FROM Production.Product AS P

INNER JOIN Production.Supplier AS S

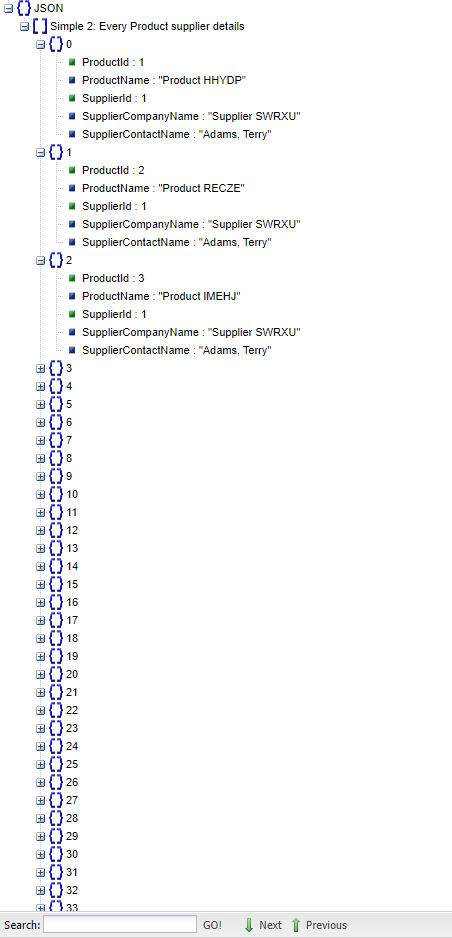
ON P.SupplierId = S.SupplierId

--FOR JSON PATH, ROOT('Simple 2: Every Product supplier details'), INCLUDE\_NULL\_VALUES;

## Results: 77 rows

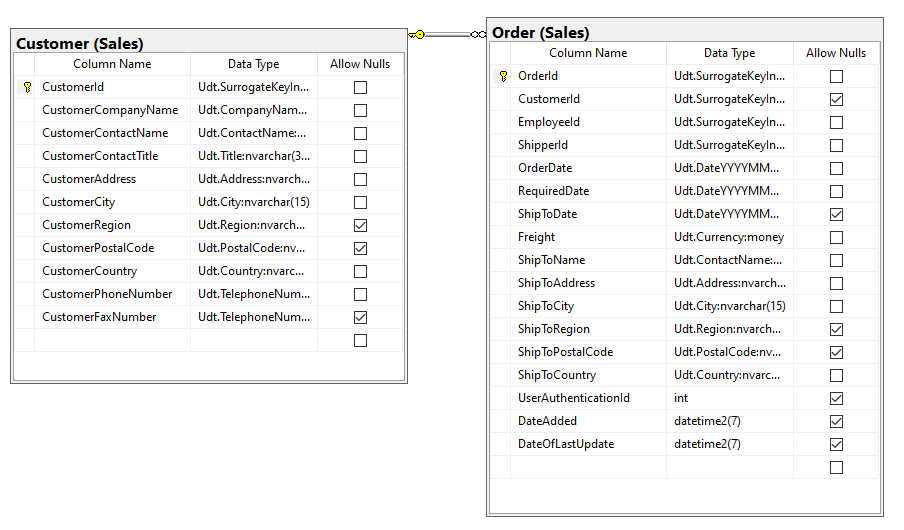


## JSON Output:

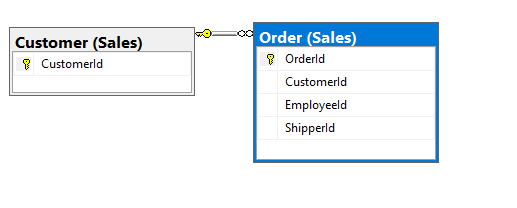


# Proposition 03: Get all orders placed by customers in the USA in 2015. [Simple]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId, OrderDate |
| Sales.Customer | CustomerId, CustomerCountry |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order  Sales.Customer | OrderId  CustomerId | ASC  ASC |

## Problem Solving Query:

USE Northwinds2020TSQLV6;

GO

SELECT O.OrderId,

C.CustomerId,

C.CustomerCountry AS Country,

O.OrderDate AS OrderDate

FROM Sales.[Order] AS O

INNER JOIN Sales.Customer AS C

ON C.CustomerId = O.CustomerId

WHERE C.CustomerCountry LIKE N'USA'

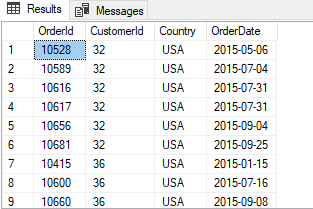
AND YEAR(O.OrderDate) = 2015

ORDER BY C.CustomerId,

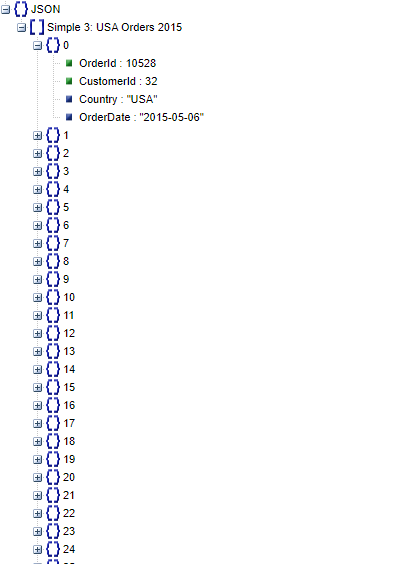
O.OrderId

--FOR JSON PATH, ROOT('Simple 3: USA Orders 2015'), INCLUDE\_NULL\_VALUES;

## Results: 60 rows

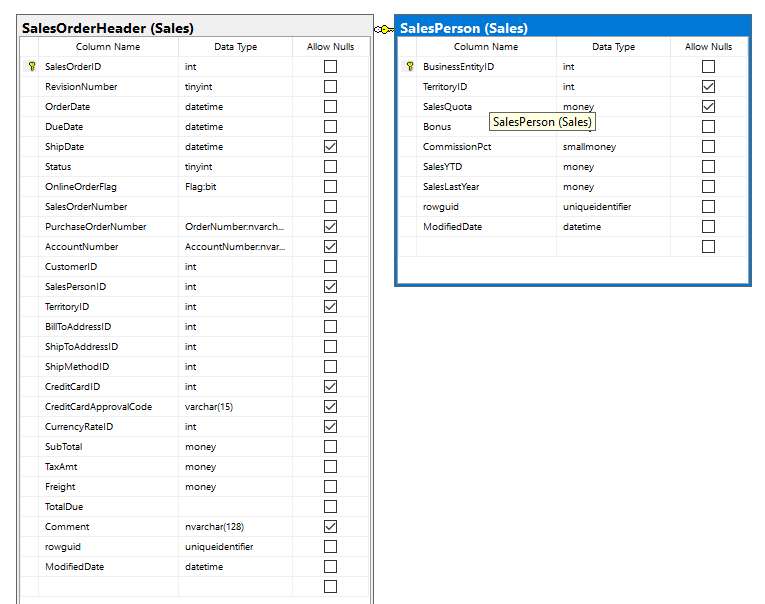


## JSON Output:

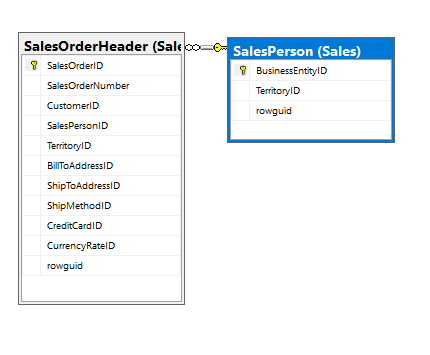


# Proposition 04: Return the Sales person freight and bonus for every sales person using AdventureWorks2017 [Simple]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.SalesOrderHeader | Freight |
| Sales.SalesPerson | BusinessEntityID, Bonus |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.SalesOrderHeader | TotalSales | ASC |

## Problem Solving Query:

USE AdventureWorks2017;

GO

SELECT SP.BusinessEntityID AS ID,

SUM(OH.Freight) AS TotalSales,

SP.Bonus AS Bonus

FROM Sales.SalesOrderHeader AS OH

INNER JOIN Sales.SalesPerson AS SP

ON SP.BusinessEntityID = OH.SalesPersonID

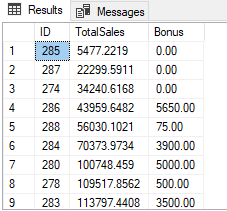
GROUP BY SP.BusinessEntityID,

SP.Bonus

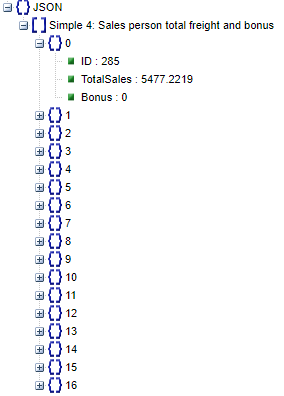
ORDER BY TotalSales

--FOR JSON PATH, ROOT('Simple 4: Sales person total freight and bonus'), INCLUDE\_NULL\_VALUES;

## Results: 17 Rows

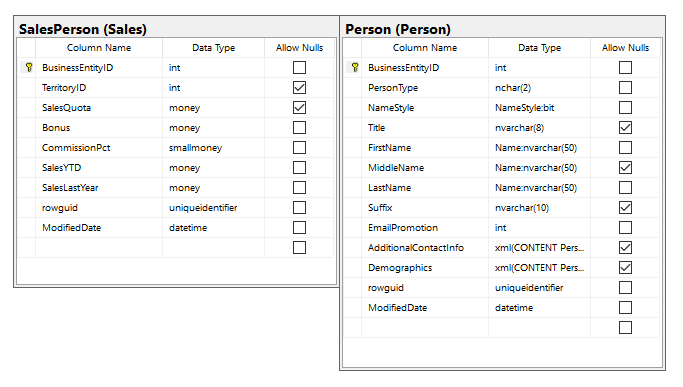


## JSON Output:

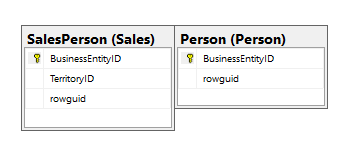


# Proposition 05: Find the name of every Employee in the Sales department using AdventureWorks2017 [Simple]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.SalesPerson | BusinessEntityID |
| Person.Person | BusinessEntityID, FirstName, LastName |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Person.Person | BusinessEntityID | ASC |

## Problem Solving Query:

USE AdventureWorks2017;

GO

SELECT P.BusinessEntityID,

P.FirstName,

P.LastName

FROM Person.Person AS P

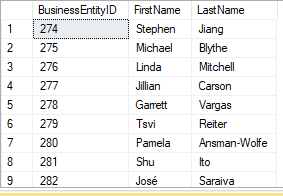
INNER JOIN Sales.SalesPerson AS S

ON S.BusinessEntityID = P.BusinessEntityID

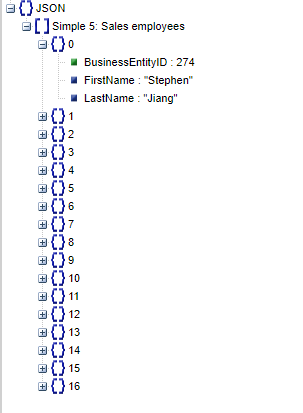
ORDER BY P.BusinessEntityID

--FOR JSON PATH, ROOT('Simple 5: Sales employees'), INCLUDE\_NULL\_VALUES;

## Results: 17 rows

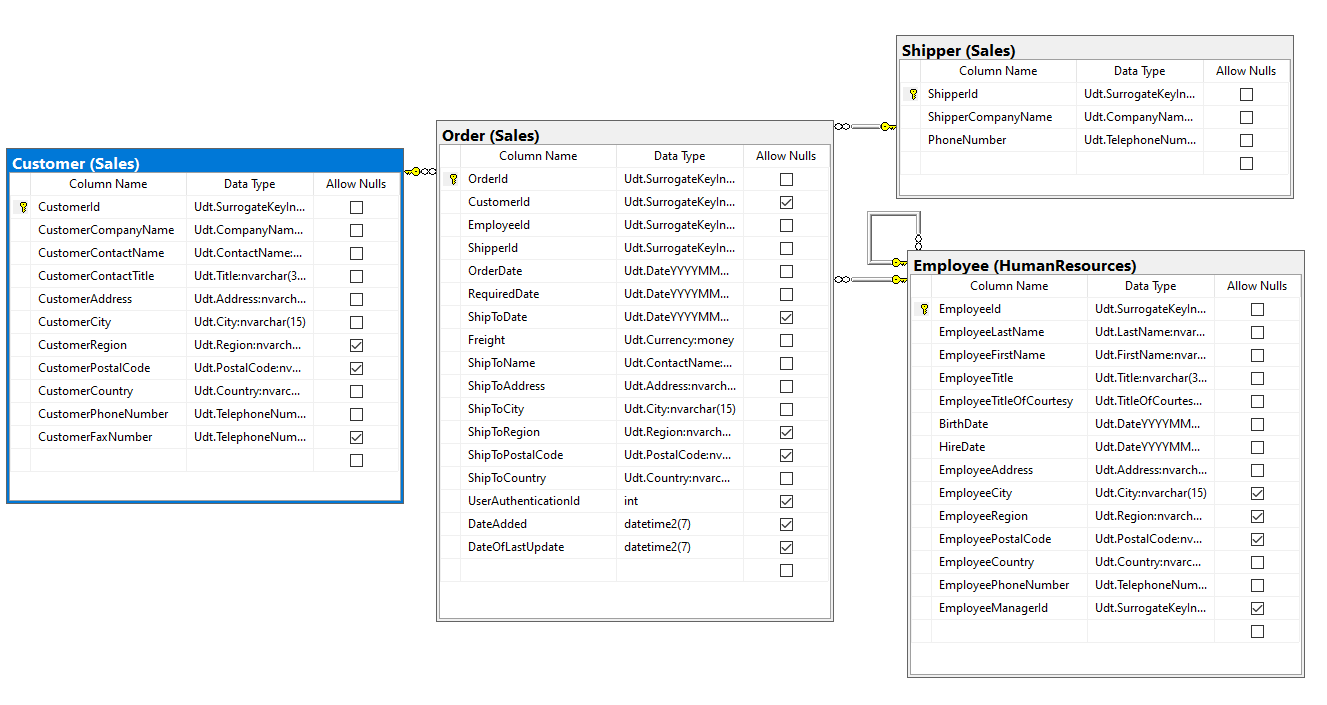


## JSON Output:

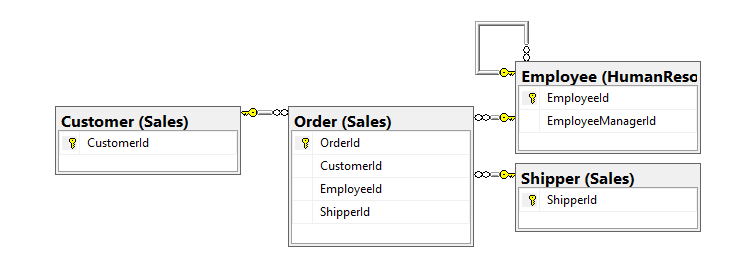


# Proposition 06: Find the Customer name, Employee Last Name, First Name , Shipper Company Name for every order using Northwinds2020TSQLV6 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId |
| Sales.Customer | CustomerContactName, CustomerId |
| HumanResources.Employee | EmployeeLastName, EmployeeFirstName, EmployeeId |
| Sales.Shipper | ShipperCompanyName, ShipperId |

## Order By:

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

## Problem Solving Query:

USE [Northwinds2020TSQLV6];

GO

SELECT O.OrderId AS OrderId,

C.CustomerContactName AS CustomerName,

CONCAT(E.EmployeeLastName, ', ', E.EmployeeFirstName) AS EmployeeName,

S.ShipperCompanyName

FROM Sales.[Order] AS O

INNER JOIN Sales.Customer AS C

ON C.CustomerId = O.CustomerId

INNER JOIN HumanResources.Employee AS E

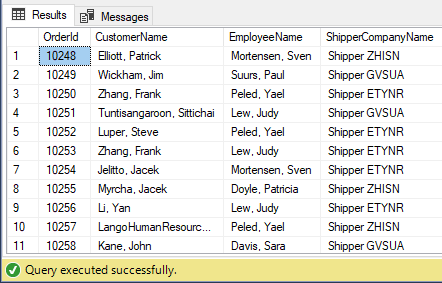
ON E.EmployeeId = O.EmployeeId

INNER JOIN Sales.Shipper AS S

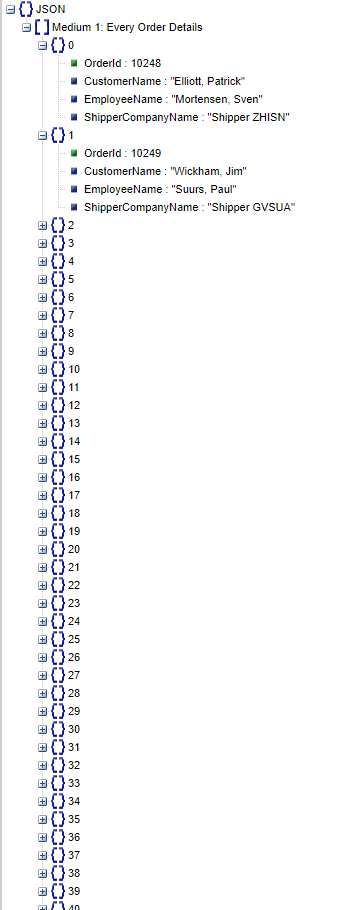
ON S.ShipperId = O.ShipperId

--FOR JSON PATH, ROOT('Medium 1: Every Order Details'), INCLUDE\_NULL\_VALUES;

## Results: 830 rows

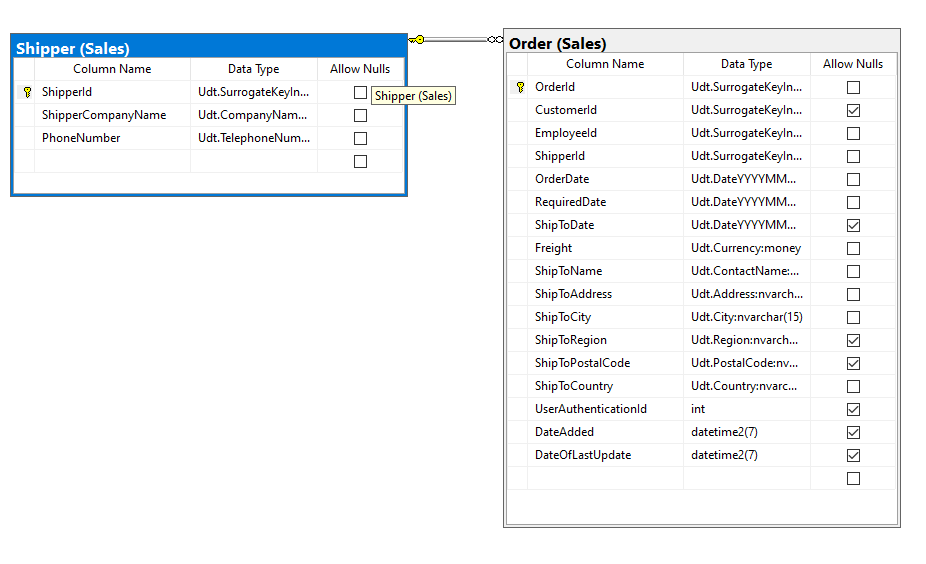


## JSON Output:

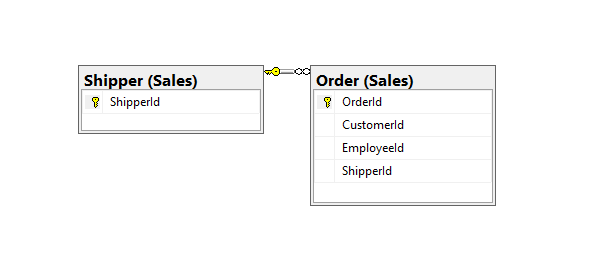


# Proposition 07: Find the 10 longest shipping times it took to ship each order, the shipping company name, phone number, and city using Northwinds2020TSQLV6 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId, OrderDate, ShipToDate, ShipToCity, ShipToCountry |
| Sales.Shipper | ShipperCompanyName, PhoneNumber |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | DaysToShip | DESC |

## Problem Solving Query:

USE [Northwinds2020TSQLV6];

GO

SELECT TOP (10)

O.OrderId AS OrderId,

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

S.ShipperCompanyName,

S.PhoneNumber,

O.ShipToCity AS OrderCity,

O.ShipToCountry AS OrderCountry

FROM Sales.[Order] AS O

INNER JOIN Sales.Shipper AS S

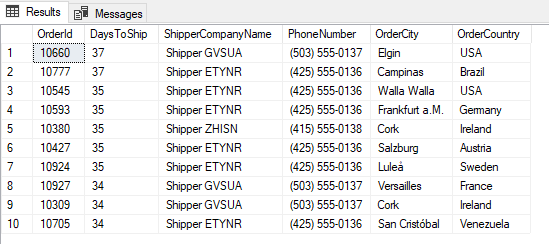
ON S.ShipperId = O.ShipperId

WHERE DATEDIFF(DAY, O.OrderDate, O.ShipToDate) >= 1

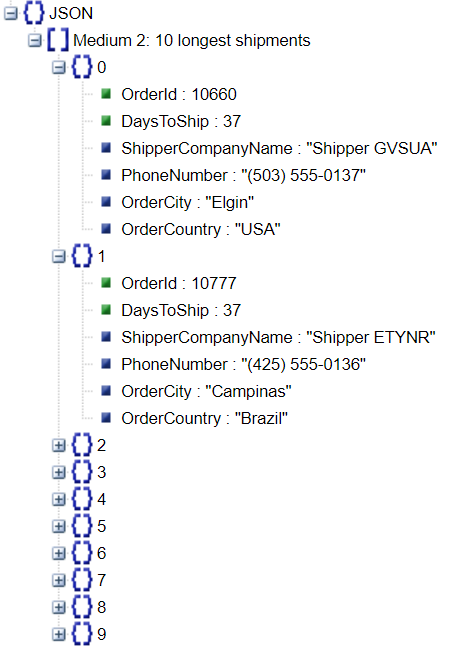
ORDER BY DaysToShip DESC

--FOR JSON PATH, ROOT('Medium 2: 10 longest shipments'), INCLUDE\_NULL\_VALUES;

## Results: 10 rows

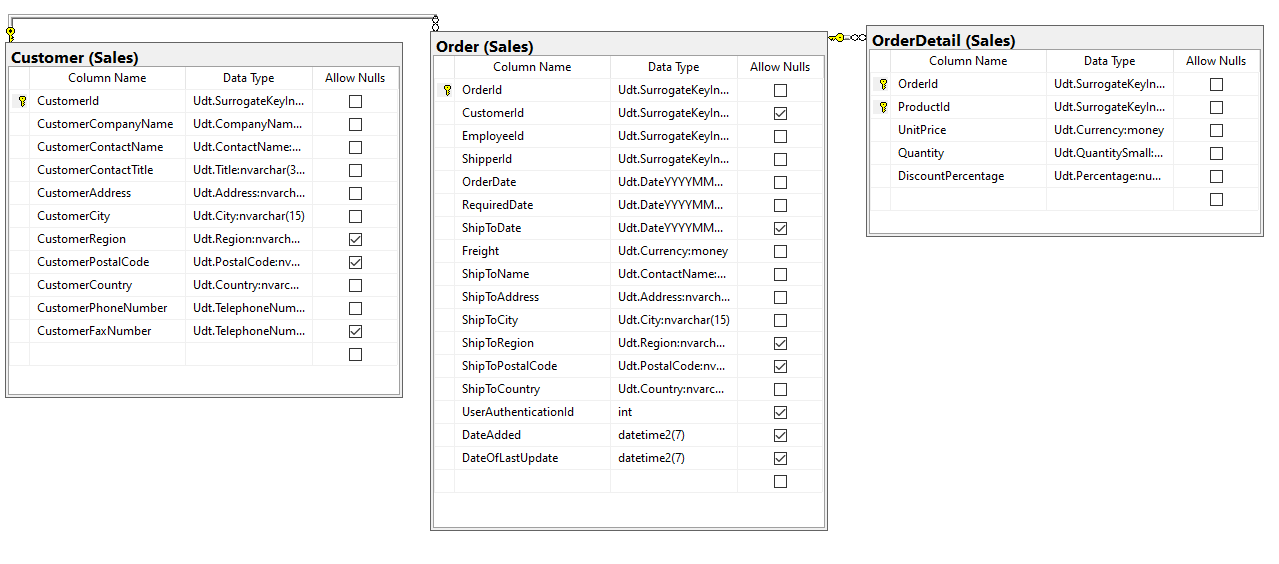


## JSON Output:

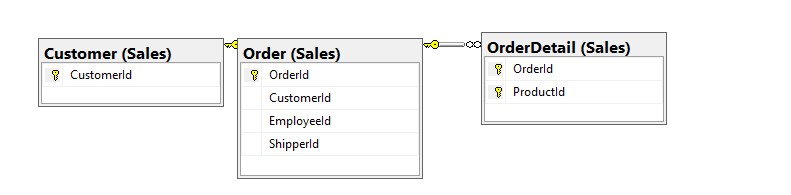


# Proposition 08: List the order details of every order from the Brazil using Northwinds2020TSQLV6 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId |
| Sales.Customer | CustomerCountry |
| Sales.OrderDetail | ProductId, UnitPrice, Quantity, LineAmount, Discount, LineAmountDiscounted |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | OrderId | ASC |
| Sales.OrderDetail | LineAmountDiscounted | ASC |

## Problem Solving Query:

USE [Northwinds2020TSQLV6];

GO

SELECT O.OrderId,

C.CustomerCountry,

OD.ProductId,

OD.UnitPrice,

OD.Quantity,

(OD.UnitPrice \* OD.Quantity) AS LineAmount,

OD.DiscountPercentage,

(OD.UnitPrice \* OD.Quantity \* (1 - OD.DiscountPercentage)) AS LineAmountDiscounted

FROM Sales.[Order] AS O

INNER JOIN Sales.Customer AS C

ON C.CustomerId = O.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON OD.OrderId = O.OrderId

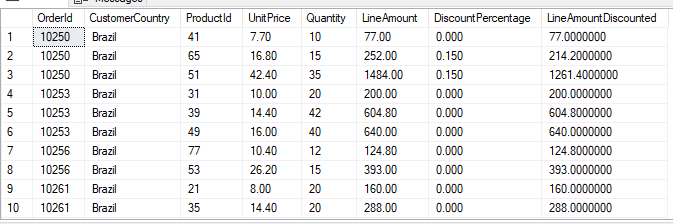
WHERE C.CustomerCountry LIKE N'Brazil'

ORDER BY O.OrderId,

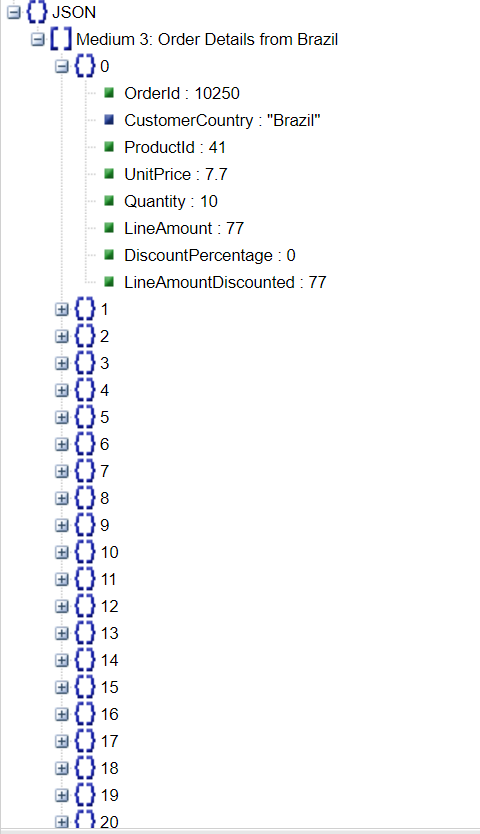
LineAmountDiscounted

--FOR JSON PATH, ROOT('Medium 3: Order Details from Brazil'), INCLUDE\_NULL\_VALUES;

## Results: 203 rows

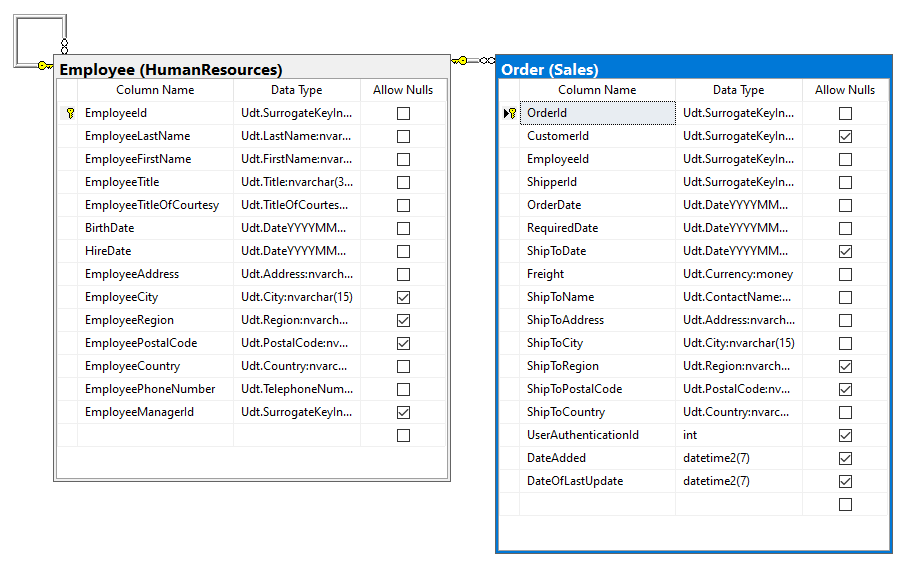


## JSON Output:

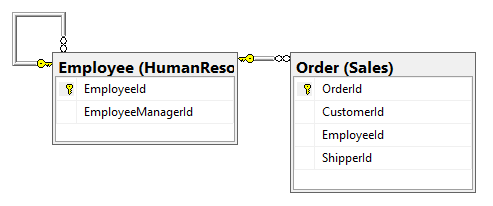


# Proposition 09: Show the top count of sales made by employees using Northwinds2020TSQLV6 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId |
| HumanResources.Employee | EmployeeId, EmployeeLastName, EmployeeFirstName |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | Orders | DESC |

## Problem Solving Query:

USE [Northwinds2020TSQLV6]

GO

SELECT CONCAT(E.EmployeeLastName, ', ', E.EmployeeFirstName) AS EmployeeFullName,

COUNT(DISTINCT O.OrderId) AS Orders

FROM Sales.[Order] AS O

INNER JOIN HumanResources.Employee AS E

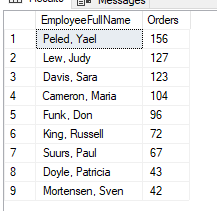
ON E.EmployeeId = O.EmployeeId

GROUP BY CONCAT(E.EmployeeLastName, ', ', E.EmployeeFirstName)

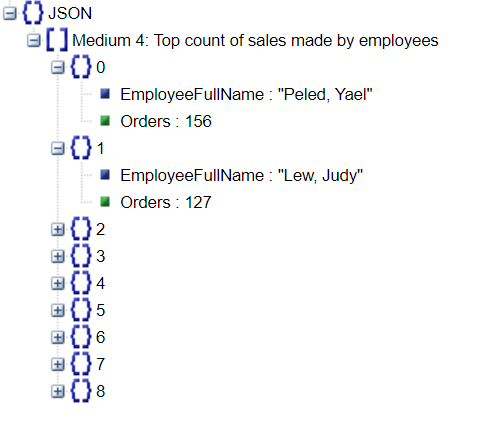
ORDER BY Orders DESC

--FOR JSON PATH, ROOT('Medium 4: Top count of sales made by employees'), INCLUDE\_NULL\_VALUES;

## Results: 9 rows

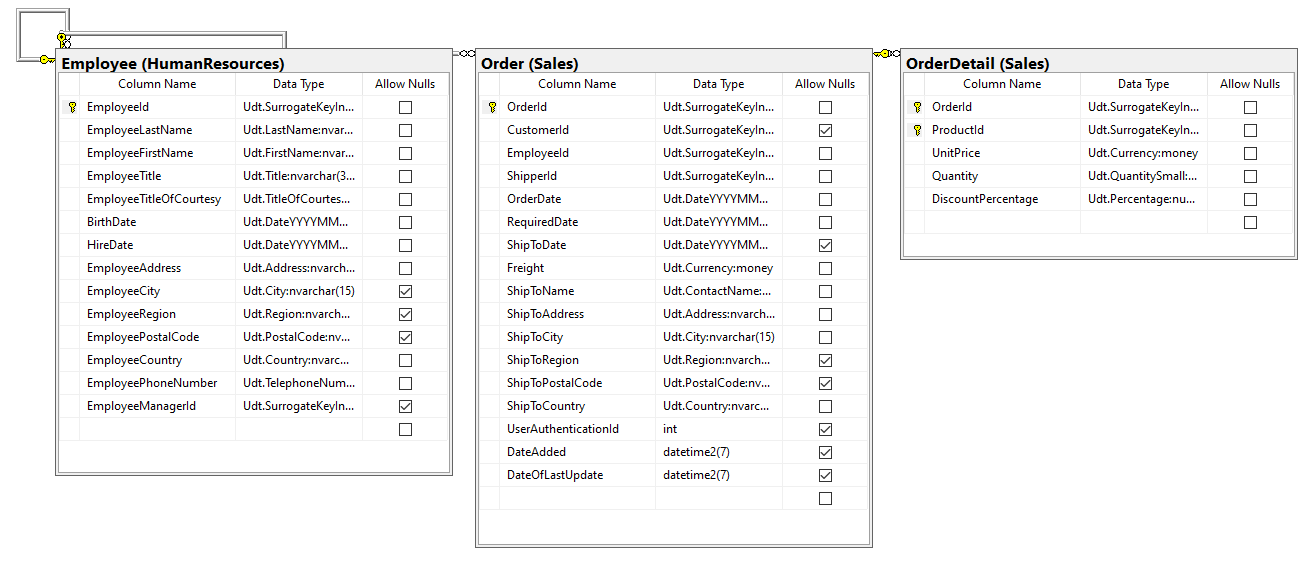


## JSON Output:

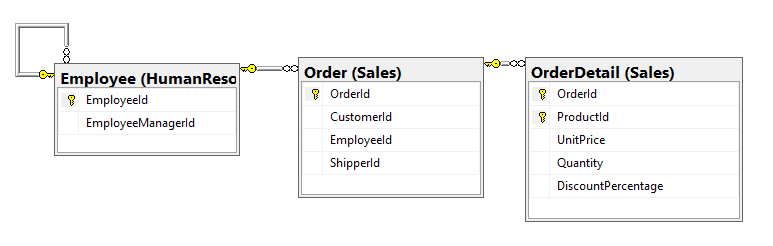


# Proposition 10: Return the total quantity for each employee and year using Northwinds2020TSQLV6 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | EmployeeId, OrderDate |
| Sales.OrderDetail | Quantity |
| HumanResources.Employee | EmployeeId |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | OrderYear, EmployeeId | ASC, ASC |

## Problem Solving Query:

USE Northwinds2020TSQLV6;

GO

SELECT O.EmployeeId,

YEAR(O.OrderDate) AS OrderYear,

SUM(Quantity) AS Quantity

FROM Sales.[Order] AS O

INNER JOIN Sales.OrderDetail AS OD

ON O.OrderId = OD.OrderId

GROUP BY O.EmployeeId,

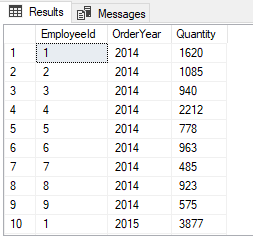
YEAR(O.OrderDate)

ORDER BY OrderYear,

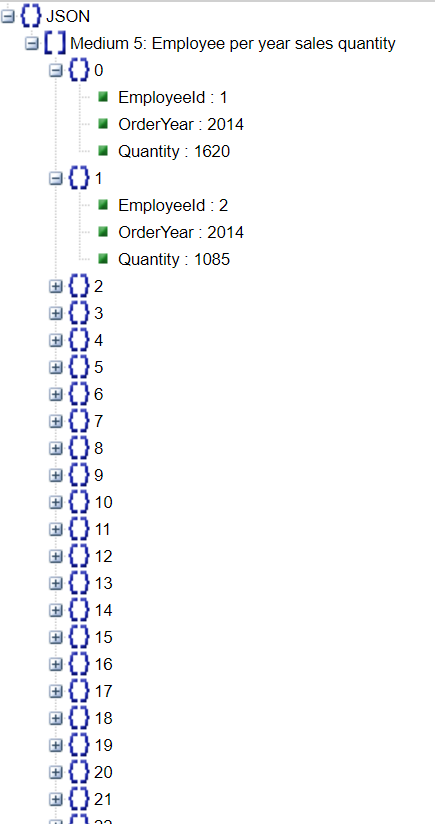
O.EmployeeId

--FOR JSON PATH, ROOT('Medium 5: Employee per year sales quantity'), INCLUDE\_NULL\_VALUES;

## Results: 27 rows

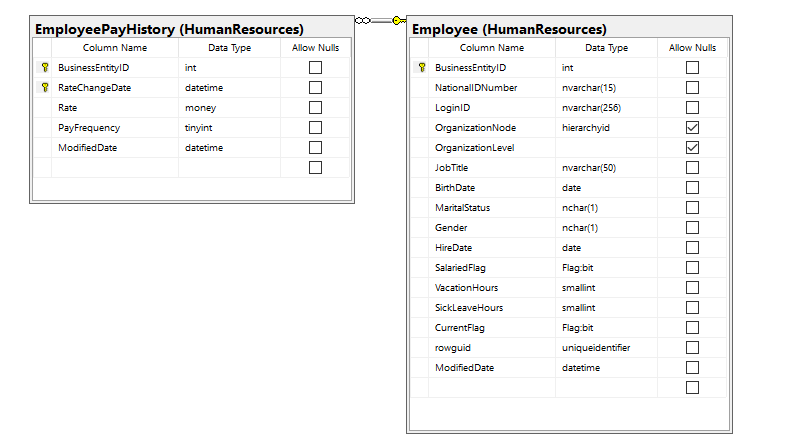


## JSON Output:

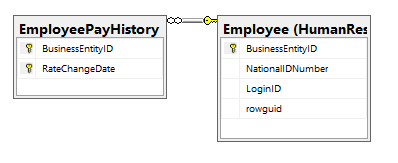


# Proposition 11: Count how many employees are single or married using AdventureWorks2017 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| HumanResources.Employee | BusinessEntityID, MaritalStatus |
| HumanResources.EmployeePayHistory | BusinessEntityID |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| HumanResources.Employee | MaritalStatus | ASC |

## Problem Solving Query:

USE [AdventureWorks2017];

GO

SELECT MaritalStatus = CASE

WHEN E.MaritalStatus LIKE N'M' THEN

'Married'

WHEN E.MaritalStatus LIKE N'S' THEN

'Single'

ELSE

'Unknown'

END,

COUNT(DISTINCT E.BusinessEntityID) AS StatusCount

FROM HumanResources.Employee AS E

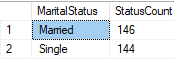
INNER JOIN HumanResources.EmployeePayHistory AS EP

ON EP.BusinessEntityID = E.BusinessEntityID

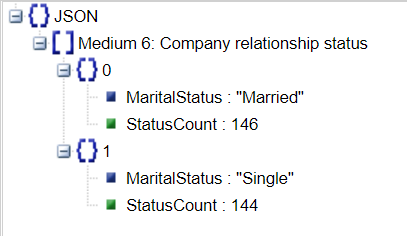
GROUP BY E.MaritalStatus

--FOR JSON PATH, ROOT('Medium 6: Company relationship status'), INCLUDE\_NULL\_VALUES;

## Results: 2 rows

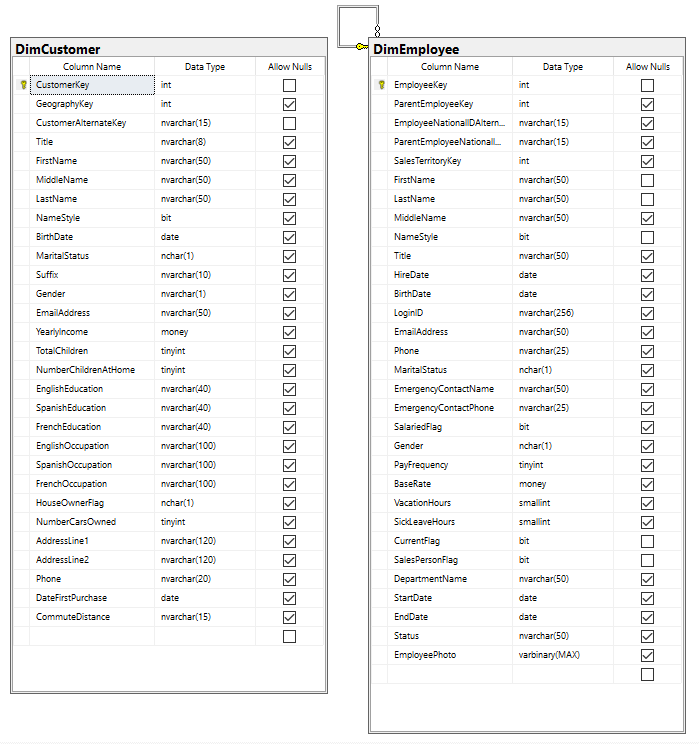


## JSON Output:



# Proposition 12: Find employees who are also customers using AdventureWorksDW2017.

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| dbo.DimCustomer | CustomerKey, FirstName, LastName |
| dbo.DimEmployee | EmployeeKey, FirstName, LastName |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| dbo.DimCustomer | CustomerName | ASC |

## Problem Solving Query:

USE AdventureWorksDW2017;

GO

SELECT DISTINCT

C.CustomerKey,

CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,

E.EmployeeKey,

CONCAT(E.FirstName, ' ', E.LastName) AS EmployeeName

FROM dbo.DimEmployee AS E

INNER JOIN dbo.DimCustomer AS C

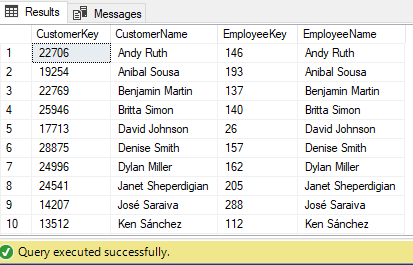
ON C.FirstName = E.FirstName

AND C.LastName = E.LastName

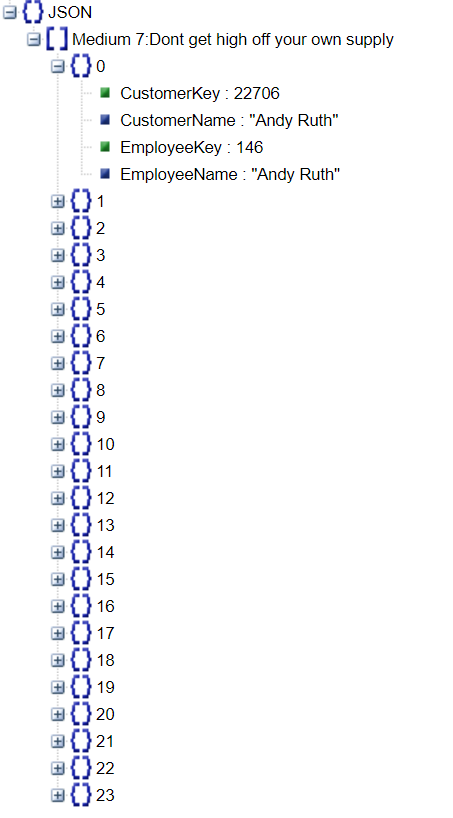
ORDER BY CustomerName

--FOR JSON PATH, ROOT('Medium 7:Don't get high off your own supply'), INCLUDE\_NULL\_VALUES;

## Results: 24 rows



## JSON Output:

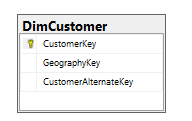


# Proposition 13: Find the average income of customers grouped by their education using AdventureWorksDW2017 [Medium]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Dbo.DimCustomer | EnglishEducation, YearlyIncome |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Dbo.DimCustomer | EnglishEducation | ASC |

## Problem Solving Query:

USE AdventureWorksDW2017;

GO

SELECT C.EnglishEducation,

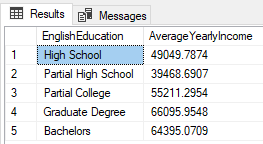
AVG(C.YearlyIncome) AS AverageYearlyIncome

FROM dbo.DimCustomer AS C

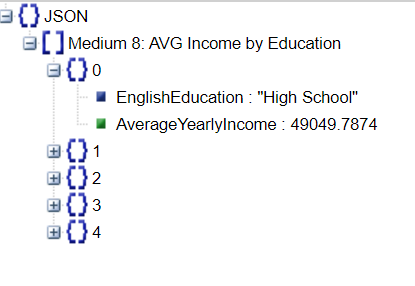
GROUP BY C.EnglishEducation

--FOR JSON PATH, ROOT('Medium 8: AVG Income by Education'), INCLUDE\_NULL\_VALUES;

## Results: 5 rows

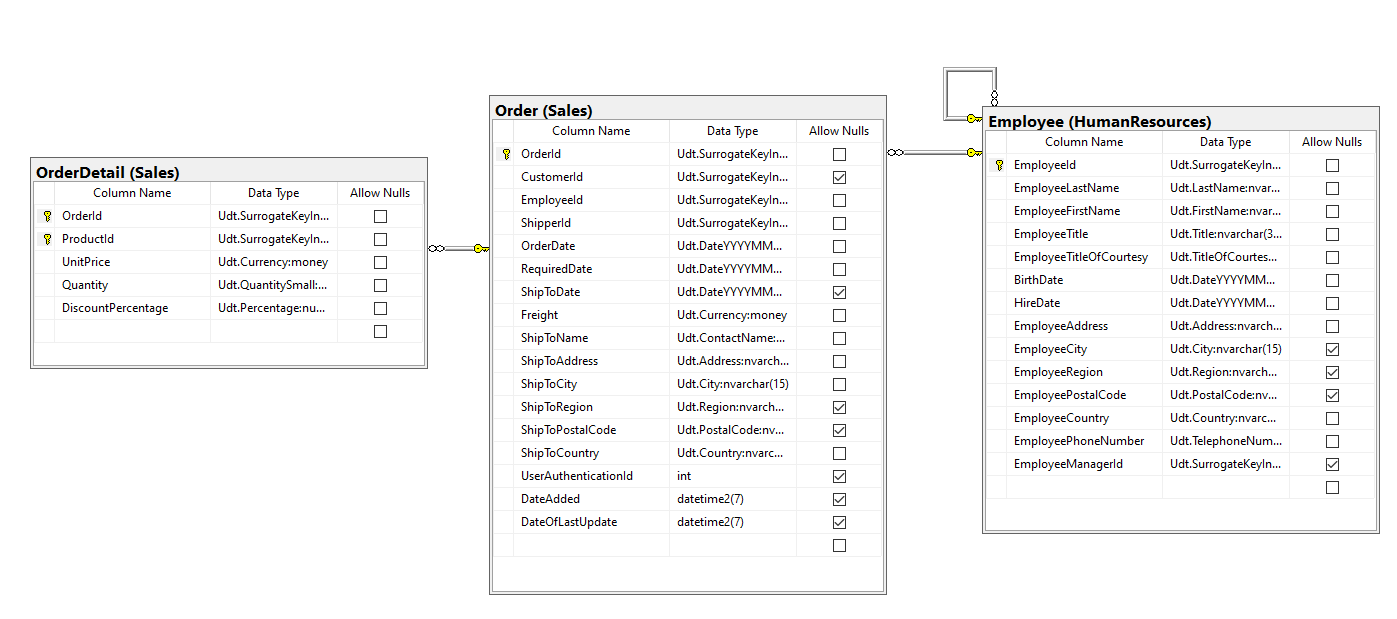


## JSON Output:

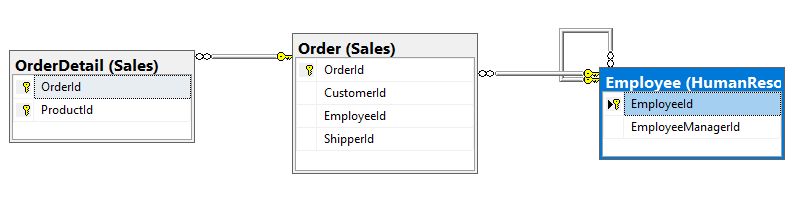


# Proposition 14: Show the top count of sales made by employees and their total sales amount using Northwinds2020TSQLV6 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderId |
| Sales.OrderDetail | Quantity, UnitPrice |
| HumanResources.Employee | E.EmployeeId, EmployeeLastName, EmployeeFirstName |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | Orders | DESC |

## Problem Solving Query:

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.EmployeeTotalFreight;

GO

SET ANSI\_NULLS ON;

GO

SET QUOTED\_IDENTIFIER ON;

GO

CREATE FUNCTION dbo.EmployeeTotalFreight

(

-- Add the parameters for the function here

@EmployeeId INT

)

RETURNS FLOAT

AS

BEGIN

-- Declare the return variable here

DECLARE @Result FLOAT;

-- Add the T-SQL statements to compute the return value here

SELECT @Result = SUM(OD.Quantity \* OD.UnitPrice)

FROM Sales.OrderDetail AS OD

INNER JOIN Sales.[Order] AS O

ON O.OrderId = OD.OrderId

WHERE O.EmployeeId = @EmployeeId;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT E.EmployeeId,

CONCAT(E.EmployeeLastName, ', ', E.EmployeeFirstName) AS EmployeeFullName,

COUNT(DISTINCT O.OrderId) AS Orders,

dbo.EmployeeTotalFreight(E.EmployeeId) AS TotalSales

FROM HumanResources.Employee AS E

INNER JOIN Sales.[Order] AS O

ON O.EmployeeId = E.EmployeeId

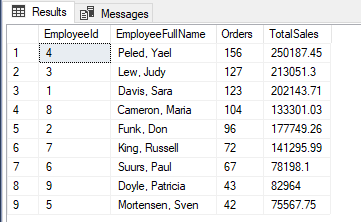
GROUP BY E.EmployeeId,

CONCAT(E.EmployeeLastName, ', ', E.EmployeeFirstName)

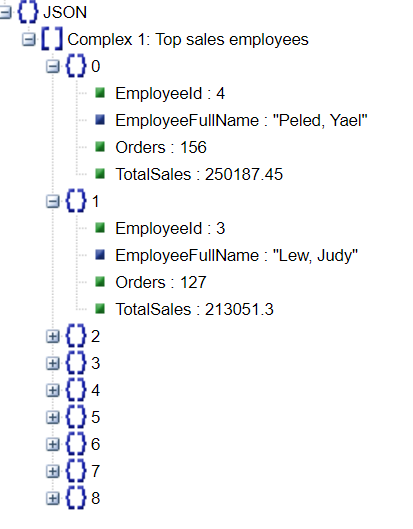
ORDER BY Orders DESC

--FOR JSON PATH, ROOT('Complex 1: Top sales employees'), INCLUDE\_NULL\_VALUES;

## Results: 9 rows

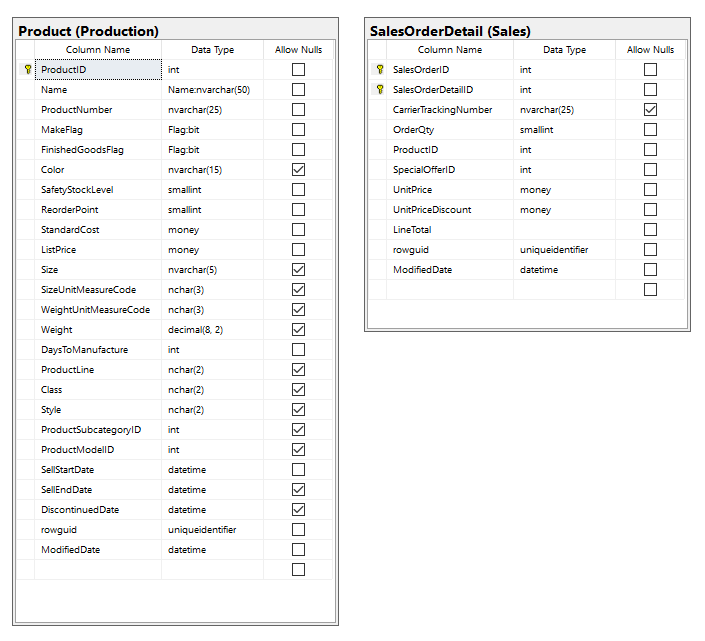


## JSON Output:

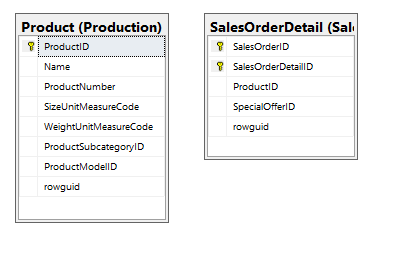


# Proposition 15: Find the top selling Products in AdventureWorks2017 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Production.Product | ProductID, Name, StandardCost |
| Sales.SalesOrderDetail | OrderQty |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Production.Product, Sales.SalesOrderDetail | QuantitySold | DESC |

## Problem Solving Query:

USE [AdventureWorks2017];

GO

DROP FUNCTION IF EXISTS dbo.QuantitySold;

GO

CREATE FUNCTION dbo.QuantitySold

(

-- Add the parameters for the function here

@ProductId INT

)

RETURNS INT

AS

BEGIN

-- Declare the return variable here

DECLARE @Result INT;

-- Add the T-SQL statements to compute the return value here

SELECT @Result = SUM(OrderQty)

FROM Sales.SalesOrderDetail

WHERE ProductID = @ProductId;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT P.ProductID,

P.[Name],

dbo.QuantitySold(P.ProductID) AS QuantitySold,

P.StandardCost,

(P.StandardCost \* dbo.QuantitySold(P.ProductID)) AS TotalEarnings

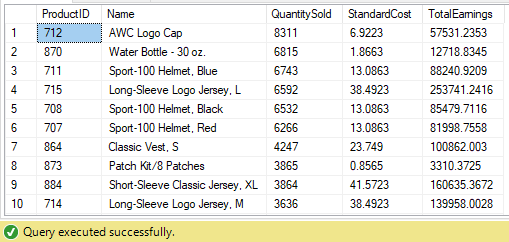
FROM Production.Product AS P

WHERE dbo.QuantitySold(P.ProductID) IS NOT NULL

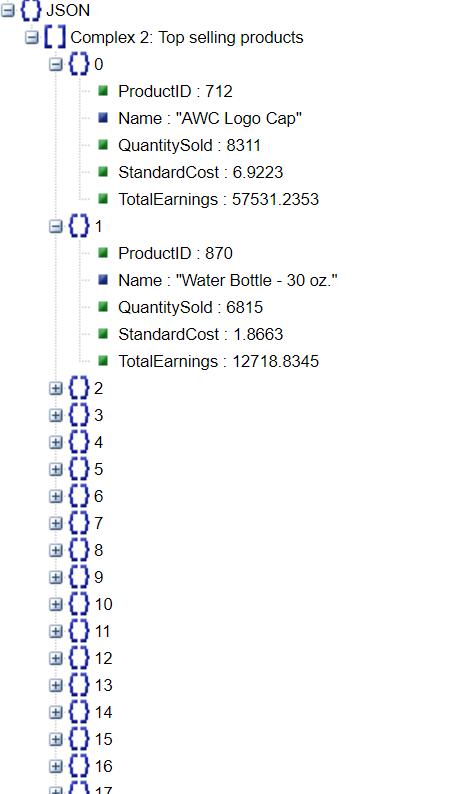
ORDER BY QuantitySold DESC

--FOR JSON PATH, ROOT('Complex 2: Top selling products'), INCLUDE\_NULL\_VALUES;

## Results: 266 rows

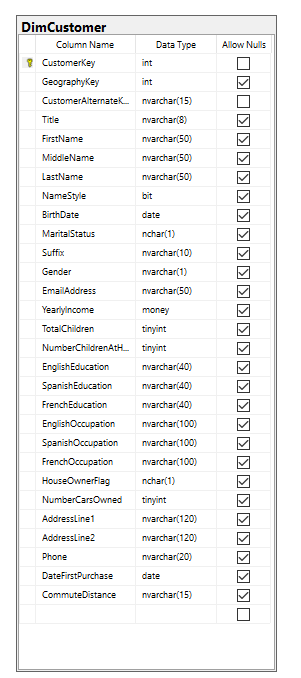


## JSON Output:

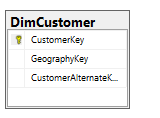


# Proposition 16: Find out how many customers have children but do not have them at home anymore using AdventureWorksDW2017 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| DimCustomer | CustomerKey, FirstName, LastName, TotalChildren, NumberChildrenAtHome |

## Order By:

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

## Problem Solving Query:

USE AdventureWorksDW2017;

GO

DROP FUNCTION IF EXISTS dbo.emptyNest;

GO

CREATE FUNCTION dbo.emptyNest

(

-- Add the parameters for the function here

@CustomerKey INT

)

RETURNS CHAR(5)

AS

BEGIN

-- Declare the return variable here

DECLARE @Result CHAR(5);

-- Add the T-SQL statements to compute the return value here

SELECT @Result = CASE

WHEN (C.TotalChildren) > 0

AND (C.NumberChildrenAtHome < C.TotalChildren) THEN

'TRUE'

ELSE

'FALSE'

END

FROM dbo.DimCustomer AS C

WHERE C.CustomerKey = @CustomerKey;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT C.CustomerKey,

CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,

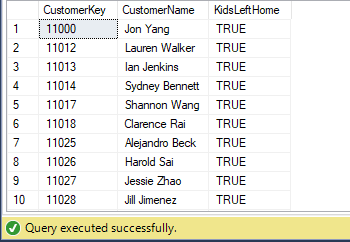
dbo.emptyNest(C.CustomerKey) AS KidsLeftHome

FROM dbo.DimCustomer AS C

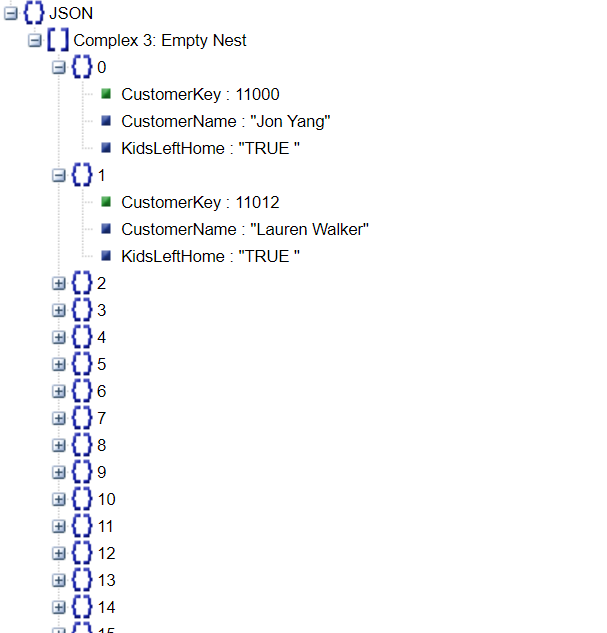
WHERE dbo.emptyNest(C.CustomerKey) LIKE 'TRUE'

--FOR JSON PATH, ROOT('Complex 3: Empty Nest'), INCLUDE\_NULL\_VALUES;

## Results: 9,074 rows

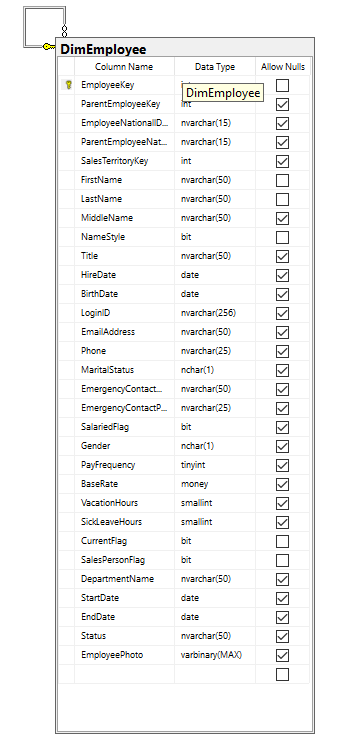


## JSON Output:

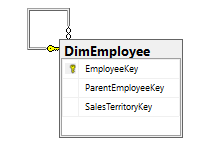


# Proposition 17: Find the AVG yearly income of employees grouped by department using AdventureWorksDW2017 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| DimEmployee | DepartmentName, EmployeeKey, BaseRate |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DimEmployee | AvgIncome | DESC |

## Problem Solving Query:

USE AdventureWorksDW2017;

GO

DROP FUNCTION IF EXISTS dbo.CalculateYearlyIncome;

GO

CREATE FUNCTION dbo.CalculateYearlyIncome

(

-- Add the parameters for the function here

@EmployeeKey INT

)

RETURNS MONEY

AS

BEGIN

-- Declare the return variable here

DECLARE @Result MONEY;

-- Add the T-SQL statements to compute the return value here

SELECT @Result = (BaseRate \* 160 \* 12) + (BaseRate \* SickLeaveHours)

FROM dbo.DimEmployee

WHERE EmployeeKey = @EmployeeKey;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT DepartmentName,

AVG(dbo.CalculateYearlyIncome(EmployeeKey)) AS AvgIncome

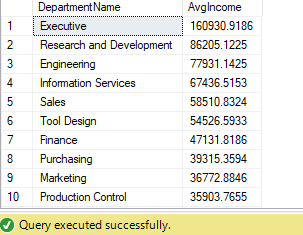
FROM dbo.DimEmployee

GROUP BY DepartmentName

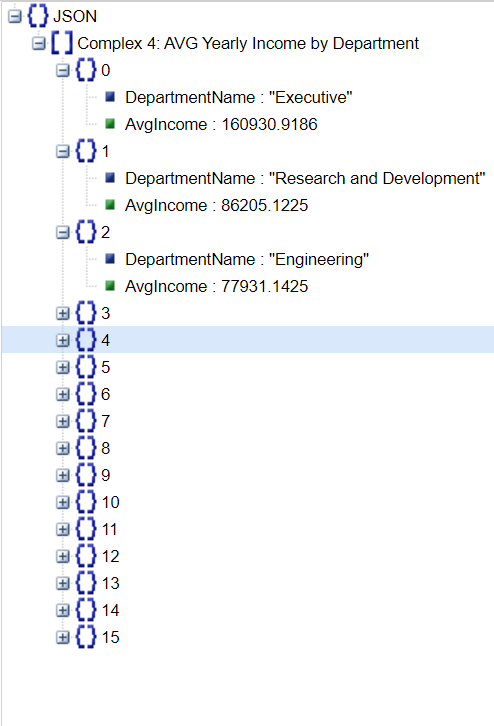
ORDER BY AvgIncome DESC

--FOR JSON PATH, ROOT('Complex 4: AVG Yearly Income by Department '), INCLUDE\_NULL\_VALUES;

## Results: 16 rows

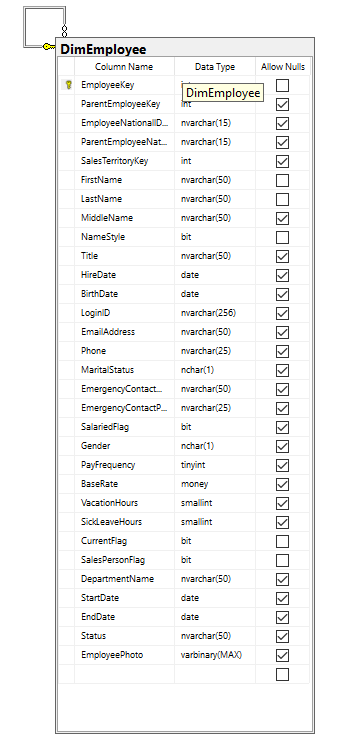


## JSON Output:

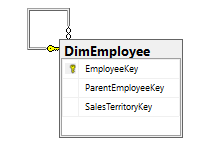


# Proposition 18: Calculate expected years until retirement for employees using AdventureWorksDW2017 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| DimEmployee | BirthDate, EmployeeKey |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| DimEmployee | YearsUntilRetirement | DESC |

## Problem Solving Query:

USE AdventureWorksDW2017;

GO

DROP FUNCTION IF EXISTS dbo.ExpectedRetirementYear;

GO

CREATE FUNCTION dbo.ExpectedRetirementYear

(

-- Add the parameters for the function here

@CurrentDate DATE,

@EmployeeKey INT

)

RETURNS INT

AS

BEGIN

-- Declare the return variable here

DECLARE @Result INT;

-- Add the T-SQL statements to compute the return value here

SELECT @Result = CASE

WHEN DATEDIFF(YEAR, BirthDate, @CurrentDate) > 65 THEN

-1

ELSE

65 - DATEDIFF(YEAR, BirthDate, @CurrentDate)

END

FROM dbo.DimEmployee

WHERE EmployeeKey = @EmployeeKey;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT EmployeeKey,

CONCAT(FirstName, ' ', LastName) AS EmployeeName,

DATEDIFF(YEAR, BirthDate, SYSDATETIME()) EmployeeAge,

dbo.ExpectedRetirementYear(SYSDATETIME(), EmployeeKey) AS YearsUntilRetirement

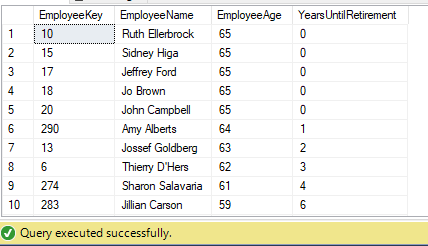
FROM dbo.DimEmployee

WHERE dbo.ExpectedRetirementYear(SYSDATETIME(), EmployeeKey) >= 0

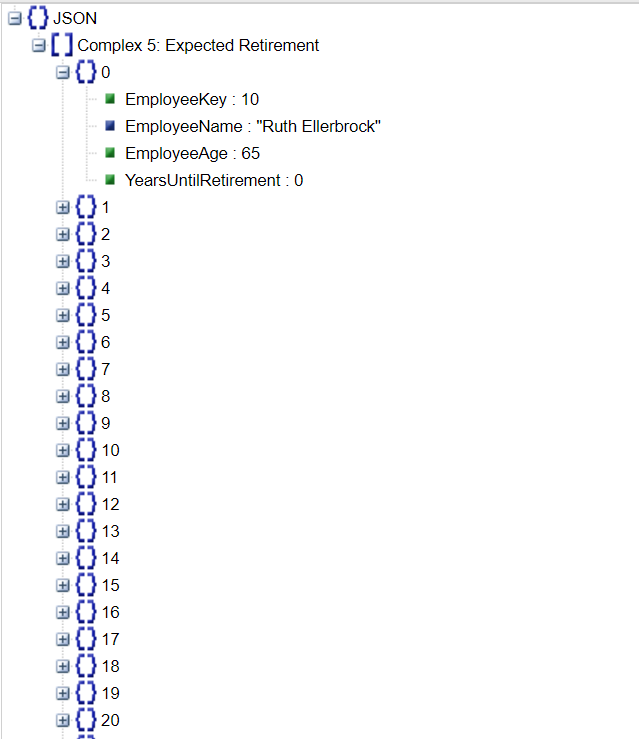
ORDER BY YearsUntilRetirement

--FOR JSON PATH, ROOT('Complex 5: Expected Retirement'), INCLUDE\_NULL\_VALUES;

## Results: 277 rows

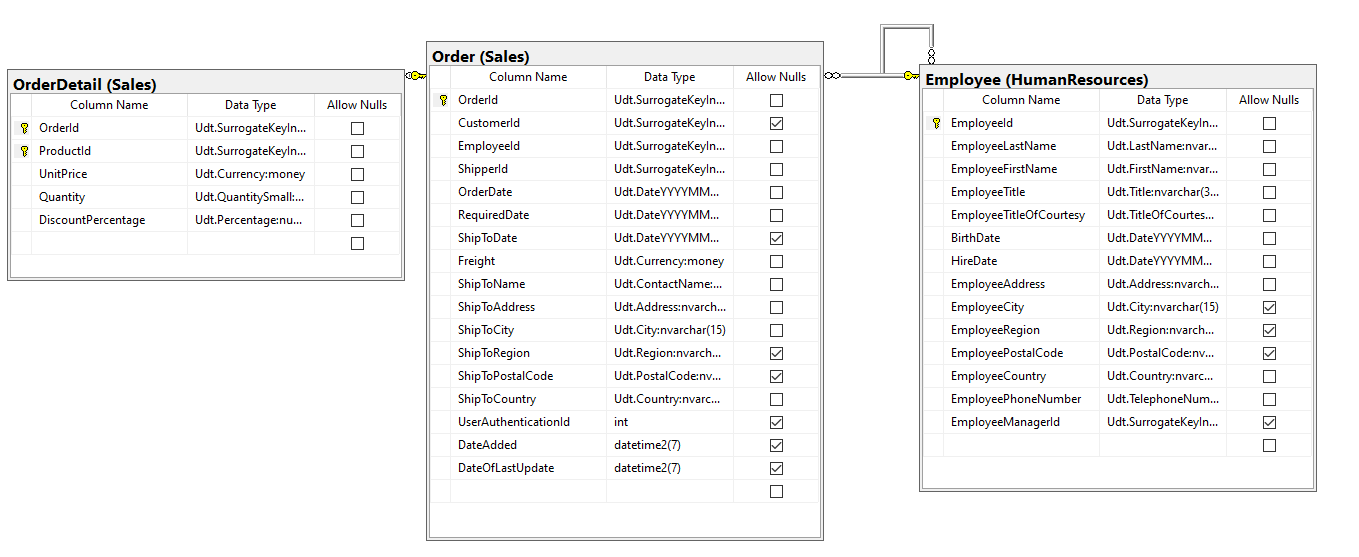


## JSON Output:

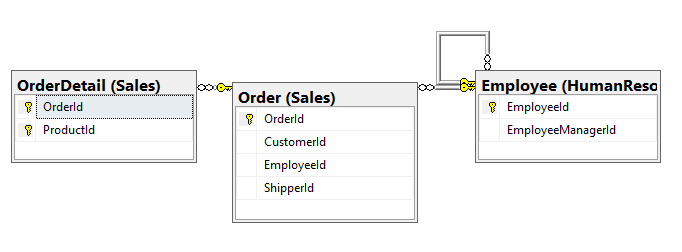


# Proposition 19: Employee of the year based on total sales using Northwinds2020TSQLV6 [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| Sales.Order | OrderDate, EmployeeId |
| HumanResources.Employee | EmployeeId |
| Sales.OrderDetail | Quantity, UnitPrice |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Order | TotalEarnings  OrderDate | DESC  ASC |

## Problem Solving Query:

USE Northwinds2020TSQLV6;

GO

DROP FUNCTION IF EXISTS dbo.EmployeeOfTheMonth;

GO

SET ANSI\_NULLS ON;

GO

SET QUOTED\_IDENTIFIER ON;

GO

CREATE FUNCTION dbo.EmployeeOfTheMonth

(

-- Add the parameters for the function here

@EmployeeId INT,

@Month INT,

@Year INT

)

RETURNS FLOAT

AS

BEGIN

-- Declare the return variable here

DECLARE @Result FLOAT;

-- Add the T-SQL statements to compute the return value here

SELECT @Result = SUM(OD.Quantity \* OD.UnitPrice)

FROM Sales.OrderDetail AS OD

INNER JOIN Sales.[Order] AS O

ON O.OrderId = OD.OrderId

WHERE O.EmployeeId = @EmployeeId

AND MONTH(O.OrderDate) = @Month

AND YEAR(O.OrderDate) = @Year;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT MONTH(O.OrderDate) AS 'Month',

E.EmployeeId,

(

SELECT TOP (1)

dbo.EmployeeOfTheMonth(E.EmployeeId, MONTH(O.OrderDate), YEAR(O.OrderDate)) AS TotalEarnings

FROM Sales.[Order] AS O

ORDER BY TotalEarnings DESC

) AS TotalEarnings

FROM HumanResources.Employee AS E

INNER JOIN Sales.[Order] AS O

ON O.EmployeeId = E.EmployeeId

WHERE YEAR(O.OrderDate) = 2015

GROUP BY MONTH(O.OrderDate),

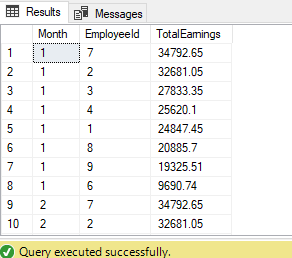
E.EmployeeId

ORDER BY MONTH(O.OrderDate),

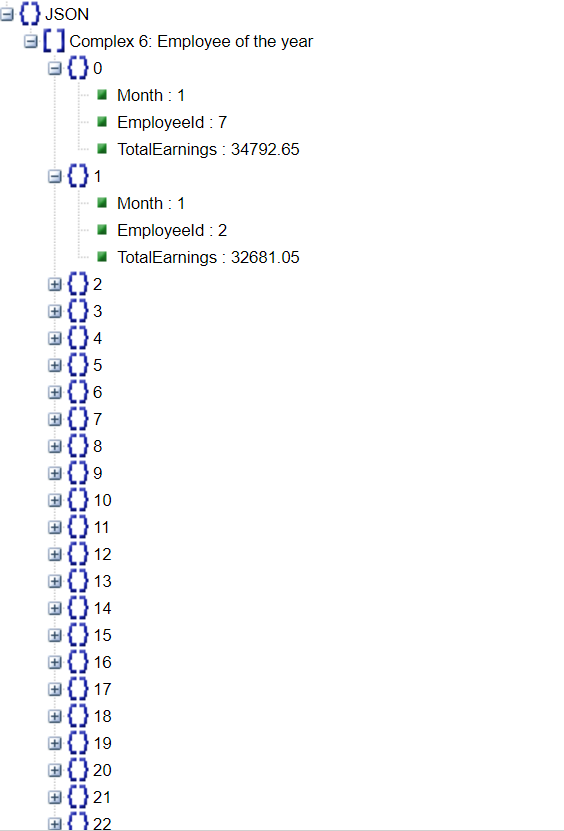
TotalEarnings DESC;

--FOR JSON PATH, ROOT('Complex 6: Employee of the year'), INCLUDE\_NULL\_VALUES;

## Results: 104 rows

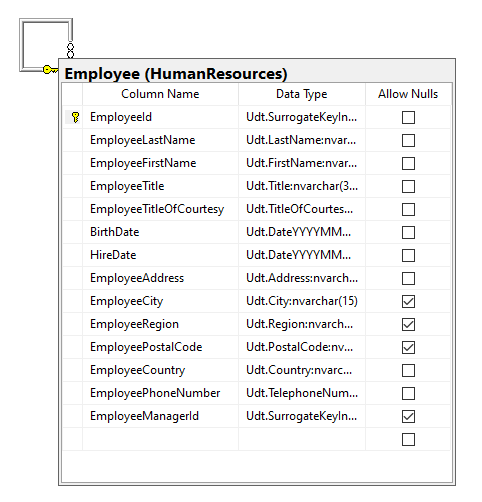


## JSON Output:



# Proposition 20: Create a location profile for each employee using Northwinds2020TSQLV6. [Complex]

## Standard View:



## Key View:



## Select By Columns:

|  |  |
| --- | --- |
| Table Name | Columns |
| HumanResources.Employee | EmployeeId, EmployeeTitleOfCourtesy, EmployeeLastName, EmployeeFirstName, EmployeeRegion, EmployeeCity, EmployeePostalCode |

## Order By:

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| HumanResources. Employee | EmployeeId | ASC |

## Problem Solving Query:

USE Northwinds2020TSQLV6;

GO

DROP FUNCTION IF EXISTS dbo.EmployeeLocation;

GO

CREATE FUNCTION dbo.EmployeeLocation

(

-- Add the parameters for the function here

@EmployeeId INT

)

RETURNS NVARCHAR(MAX)

AS

BEGIN

-- Declare the return variable here

DECLARE @Result NVARCHAR(MAX);

-- Add the T-SQL statements to compute the return value here

SELECT @Result

= CONCAT(

E.EmployeeAddress,

' ',

E.EmployeePostalCode,

' ',

E.EmployeeCity,

' ',

E.EmployeeRegion,

' ',

E.EmployeeCountry

)

FROM HumanResources.Employee AS E

WHERE E.EmployeeId = @EmployeeId;

-- Return the result of the function

RETURN @Result;

END;

GO

SELECT EmployeeId,

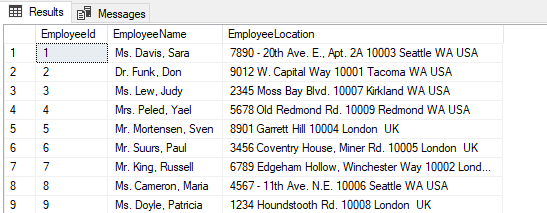
CONCAT(EmployeeTitleOfCourtesy, ' ', EmployeeLastName, ', ', EmployeeFirstName) AS EmployeeName,

dbo.EmployeeLocation(EmployeeId) AS EmployeeLocation

FROM HumanResources.Employee

--FOR JSON PATH, ROOT('Complex 7: Location Profiles'), INCLUDE\_NULL\_VALUES;

## Results: 9 rows



## JSON Output:

