Group Project#1

Assignment Template

Group Member: Mudit Prashar

Table of Contents

Problem 01: Find department name by using departmentID of people who have start date year 2009 using AdventureWorks2014?

Problem 02: Find the product names that had internet sale and had order date year 2010 using AdventureWorksDW2016?

Problem 03: Find the description of items whose last cost price is less than $5 using WideWorldImporters?

Problem 04: Find all the orders that was purchased using EFT in the first month of year 2015 using WideWorldImportersDW?

Problem 05: Find all the orders ids that had orders placed for Switzerland and find the shipping company names using Northwinds2019TSQLV5?

Problem 06: Find the orderID of the last order placed by each Company using Northwinds2019TSQLV5?

Problem 07: Find the number of orders shipped by each shipping Company using Northwinds2019TSQLV5?

Problem 08: Find the total number of orders placed of each stock item using WideWorldImportersDW?

Problem 09: Find the revenue WideWorldImporters made from each item in 2013 using WideWorldImporters?

Problem 10: Find the number of businesses in each state of US using AdventureWorks2014?

Problem 11: Find the Business ID of the first business that is registered using AdventureWorks2014?

Problem 12: Find the total number of male customers in each country using AdventureWorksDW2016?

Problem 13: Find the Total number of orders placed during internet sales that used different currencies and don't show the currencies that were never used using AdventureWorksDW2016?

Problem 14: Find the Full name of the male employees and their age when they got hired and who only do day shifts using AdventureWorks2014?

Problem 15: Find the name of the Customers, number of clothing internet sales order they placed and also count the number of years since they made their first purchase up until today using AdventureWorksDW2016?

Problem 16: Find the sales total order for each order including tax and freight cost and also find the reason for sales using AdventureWorks2014?

Problem 17: Find the fiscal year quarter for order date for each customer the first quarter beginning with july. also find the total freight, total line cost and total cost customer paid after discounts using Northwinds2019TSQLV5?

Problem 18: Find the orders total after discount and show which ones are discounted, also show the name of the product and category it belongs to using Northwinds2019TSQLV5?

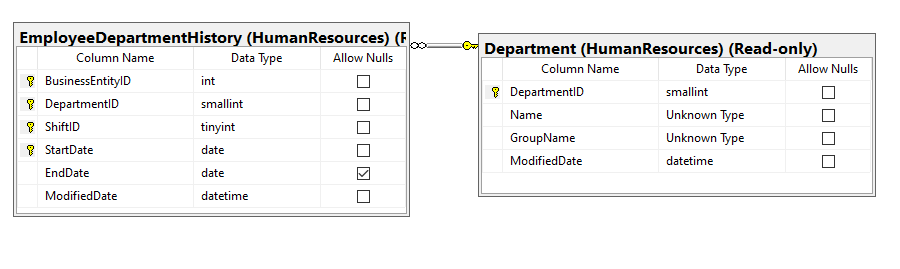
Problem 19: Find the all the orders that were placed in 2013 and show the order date in form of fiscal year quarters with January as the 1st month for first quarter and also calculate the total amount paid for each order ID including taxes using WideWorldImporters?

Problem 20: Find the total quantities of each item that was ordered and also the tax amount for one quantity using WideWorldImportersDW?

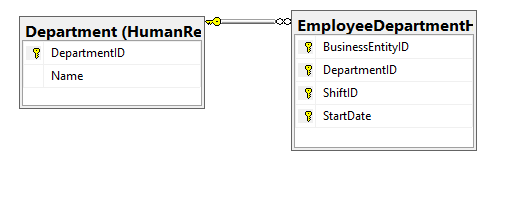
Problem 01: Find department name by using departmentID of people who have start date year 2009 using AdventureWorks2014?

Explanation: This query joins two tables named Department and Employee Department History, outputs the name of the department, start date and departmentID for the year 2009.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Department | * Name * DepartmentID |
| HumanResources.EmployeeDepartmentHistory | * StartDate |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| HumanResources.EmployeeDepartmentHistory | StartDate | ASC |

Problem Solving Query

USE AdventureWorks2014

SELECT [Name],[StartDate], a.[DepartmentID]

FROM [HumanResources].[Department] AS a

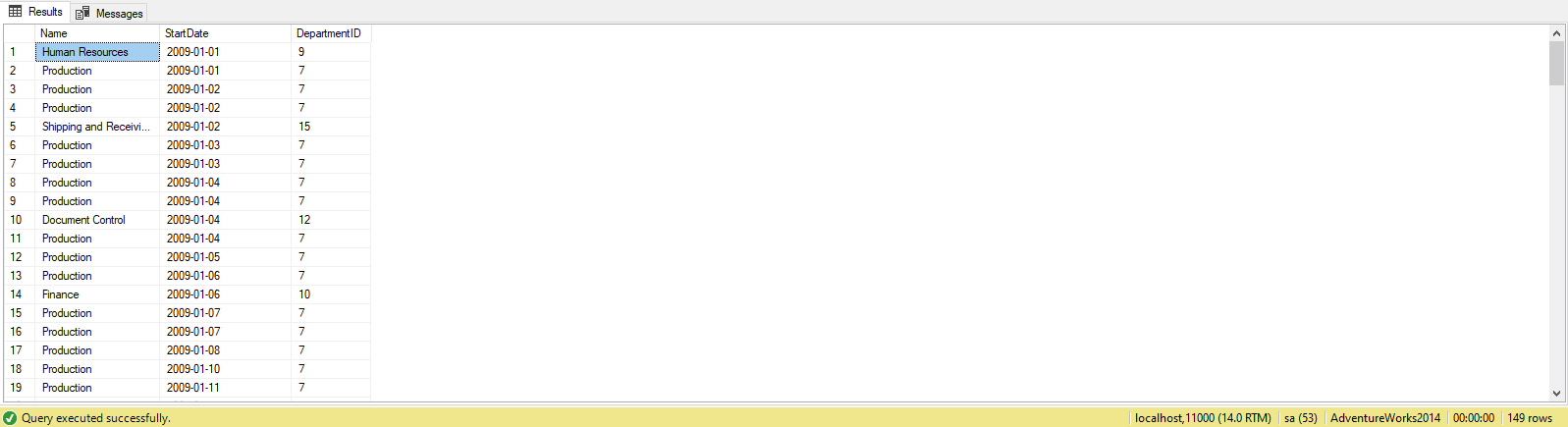
INNER JOIN [HumanResources].[EmployeeDepartmentHistory] AS b

ON a.DepartmentID = b.DepartmentID

Where StartDate BETWEEN '2009-01-01' AND '2009-12-31'

ORDER BY StartDate;

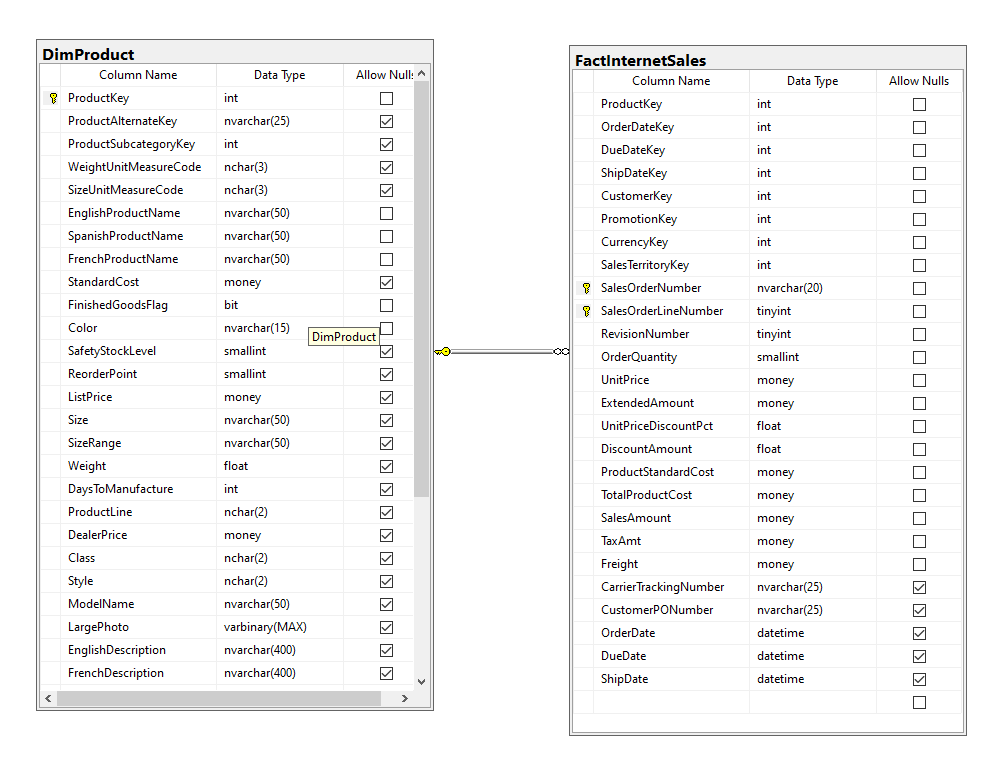
Sample Output (149 Rows Returned)



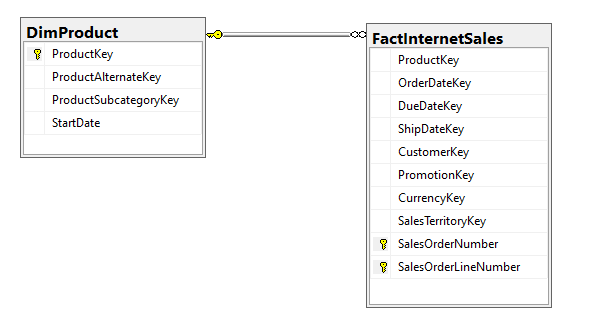
Problem 02: Find the product names that had internet sale and had order date year 2010 using AdventureWorksDW2016?

Explanation: This Query shows the order date, product key and the name of the products that was ordered during internet sales in the year 2010.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| dbo.DimProduct | * ProductKey * EnglishProductName |
| dbo.FactInternetsales | * OrderDate |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| dbo.FactInternetSales | OrderDate | ASC |

Problem Solving Query

USE AdventureWorksDW2016

SELECT b.[OrderDate]

,a.[ProductKey]

,a.[EnglishProductName]

FROM [dbo].[DimProduct] AS a

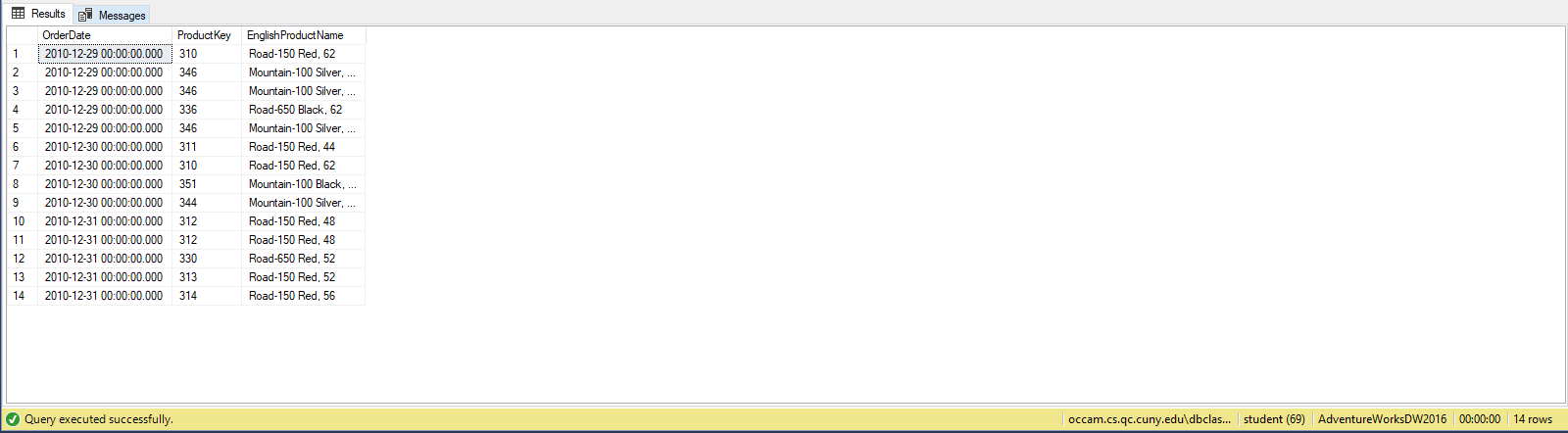
INNER JOIN [dbo].[FactInternetSales] AS b ON a.ProductKey = b.ProductKey

WHERE OrderDate BETWEEN '2010-01-01'

AND '2010-12-31'

ORDER BY OrderDate;

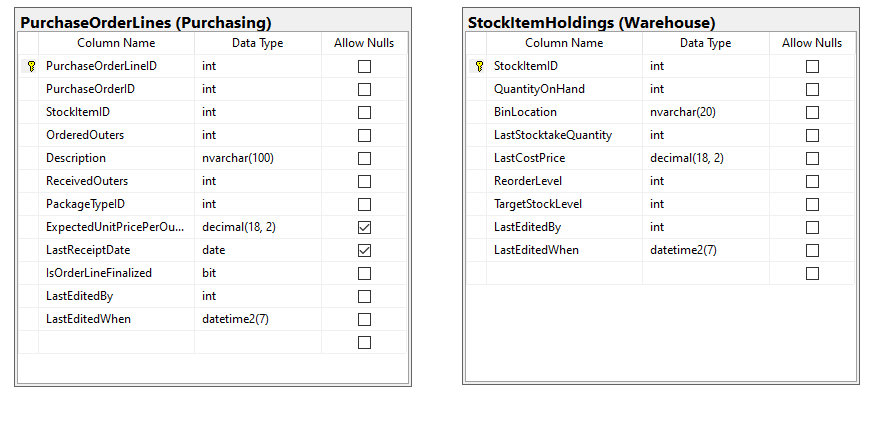
Sample Output (14 Rows Returned)



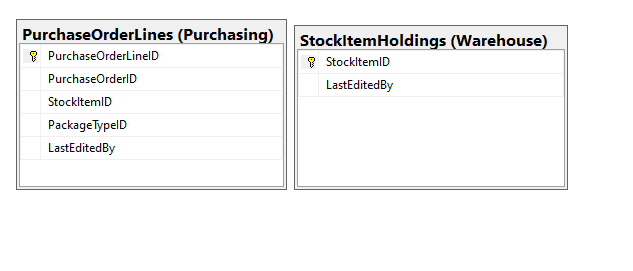
Problem 03: Find the description of items whose last cost price is less than $5 using WideWorldImporters?

Explanation: This Query shows the stock item id, last cost price and description of the item constraining the output to show only items that have last cost price less than $5.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Purchasing.PurchasingOrderLines | * StockItemId * Description |
| Warehouse.StockItemHoldings | * LastCostPrice |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Warehouse.StockItemHoldings | LastCostPrice | ASC |

Problem Solving Query

USE WideWorldImporters

SELECT DISTINCT a.[StockItemID]

,b.[LastCostPrice]

,a.[Description]

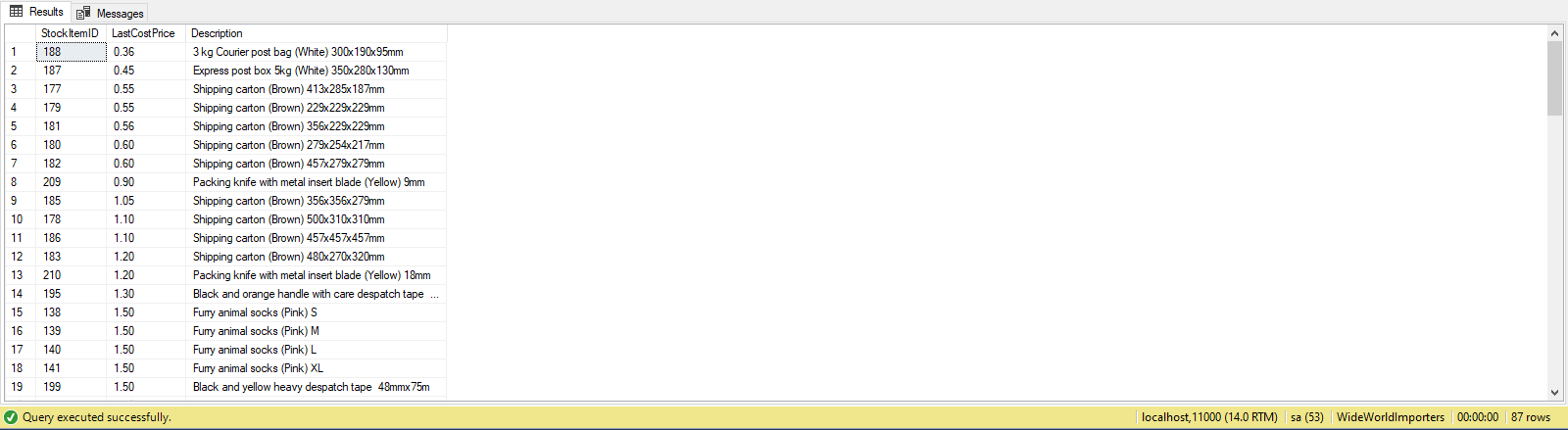
FROM [Purchasing].[PurchaseOrderLines] AS a

INNER JOIN [Warehouse].[StockItemHoldings] AS b ON a.StockItemID = b.StockItemID

WHERE b.LastCostPrice < '5'

ORDER BY b.LastCostPrice;

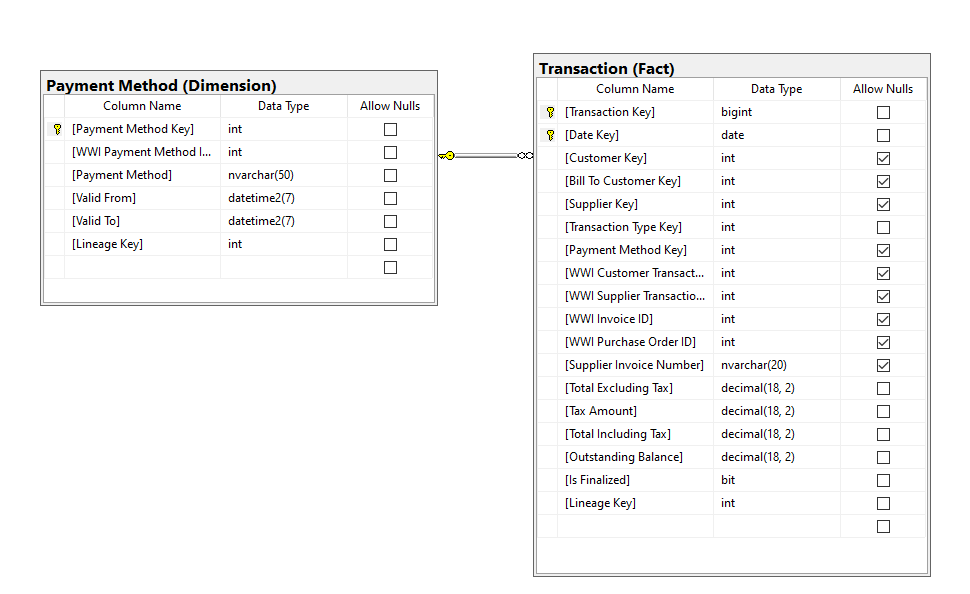
Sample Output (87 Rows Returned)



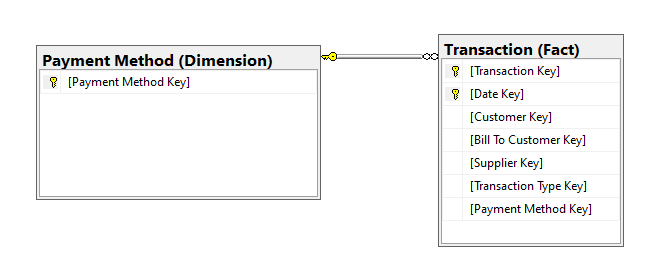
Problem 04: Find all the orders that was purchased using EFT in the first month of year 2015 using WideWorldImportersDW?

Explanation: This Query shows the payment method key, payment method used and the date when it was used constraining it to the month of January in the year 2015.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Fact.Transaction | * Payment Method Key * Date Key |
| Dimension.Payment Method | * Payment Method |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Fact.Transaction | Date Key | ASC |

Problem Solving Query

USE WideWorldImportersDW

SELECT a.[Payment Method Key]

,b.[Payment Method]

,a.[Date Key]

FROM [Fact].[Transaction] AS a

INNER JOIN [Dimension].[Payment Method] AS b ON a.[Payment Method Key] = b.[Payment Method Key]

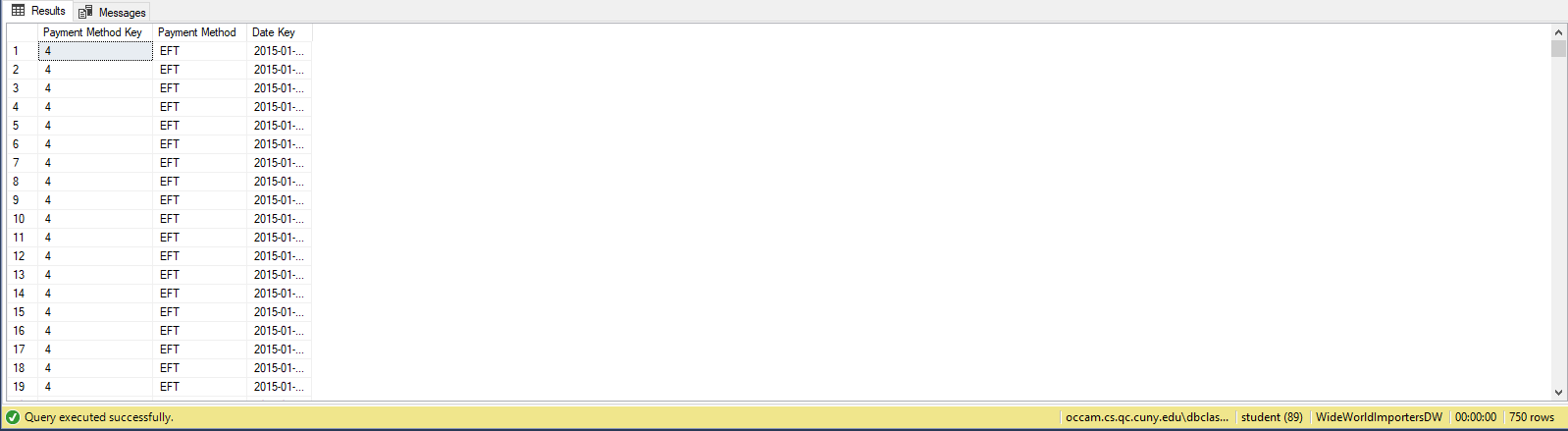
WHERE a.[Date Key] BETWEEN '2015-01-01'

AND '2015-01-31'

AND b.[Payment Method] = 'EFT'

ORDER BY a.[Date Key]

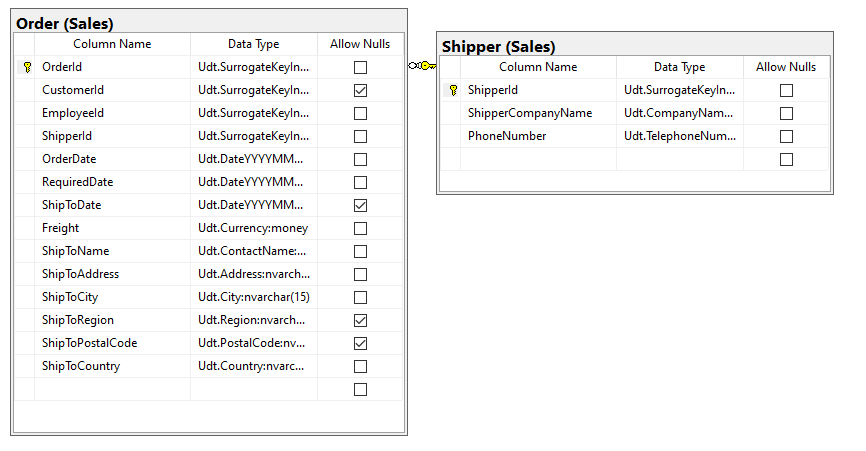
Sample Output (750 Rows Returned)



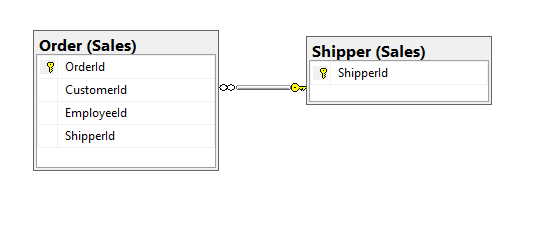
Problem 05: Find all the orders ids that had orders placed for Switzerland and find the shipping company names using Northwinds2019TSQLV5?

Explanation: This Query outputs all the orderIds and the name of shipping company for all the orders that have been sent to Switzerland.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Order | * OrderID * ShipperId * ShipToCountry |
| Sales.Shipper | * ShipperCompanyName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| N/A | N/A | N/A |

Problem Solving Query

USE Northwinds2019TSQLV5

SELECT a.OrderId

,a.ShipperId

,a.ShipToCountry

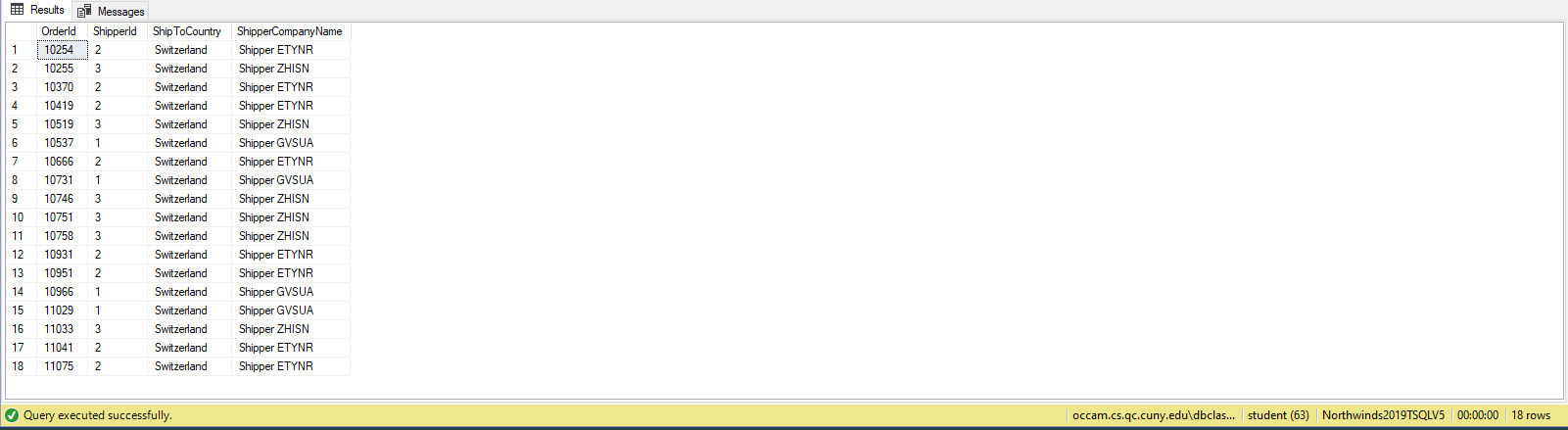
,b.ShipperCompanyName

FROM [Sales].[Order] AS a

INNER JOIN [Sales].[Shipper] AS b ON a.ShipperId = b.ShipperId

WHERE a.ShipToCountry = 'Switzerland';

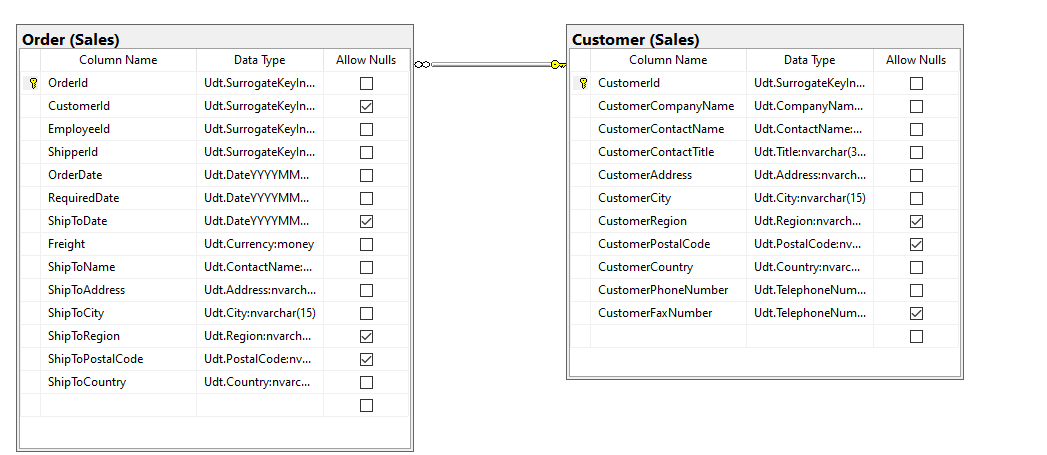
Sample Output (18 Rows Returned)



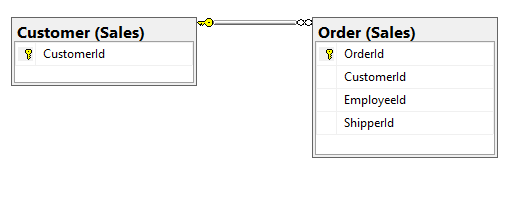
Problem 06: Find the orderID of the last order placed by each Company using Northwinds2019TSQLV5?

Explanation: This Query uses the max function to find the last orderID for each company and outputs it next to the company name.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Order | * LastPlacedOrderId |
| Sales.Customer | * CustomerCompanyName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| N/A | N/A | N/A |

Problem Solving Query

USE Northwinds2019TSQLV5

SELECT MAX(a.OrderId) AS LastPlacedOrderId

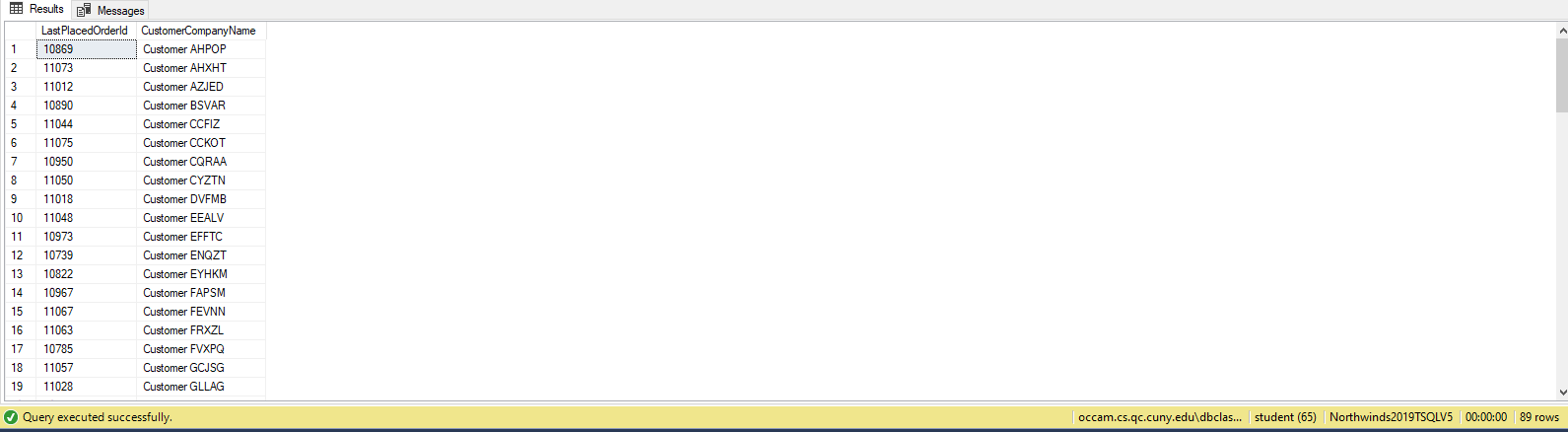
,b.CustomerCompanyName

FROM [Sales].[Order] AS a

LEFT JOIN [Sales].[Customer] AS b ON a.CustomerId = b.CustomerId

GROUP BY CustomerCompanyName

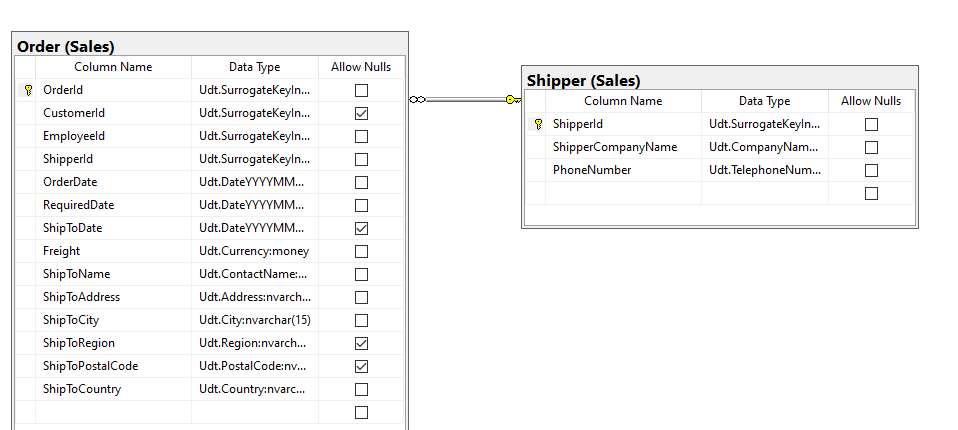
Sample Output (89 Rows Returned)



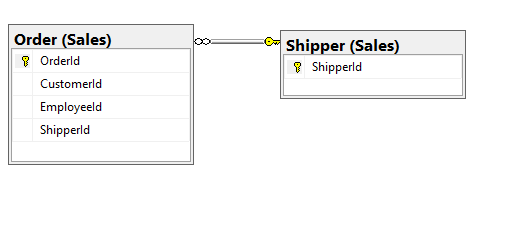
Problem 07: Find the number of orders shipped by each shipping Company using Northwinds2019TSQLV5?

Explanation: This Query uses the Count function to output the total number of orders each shipping company shipped.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Order | * NumberofOrders |
| Sales.Shipper | * ShipperComapnyName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| N/A | N/A | N/A |

Problem Solving Query

USE Northwinds2019TSQLV5

SELECT b.ShipperCompanyName

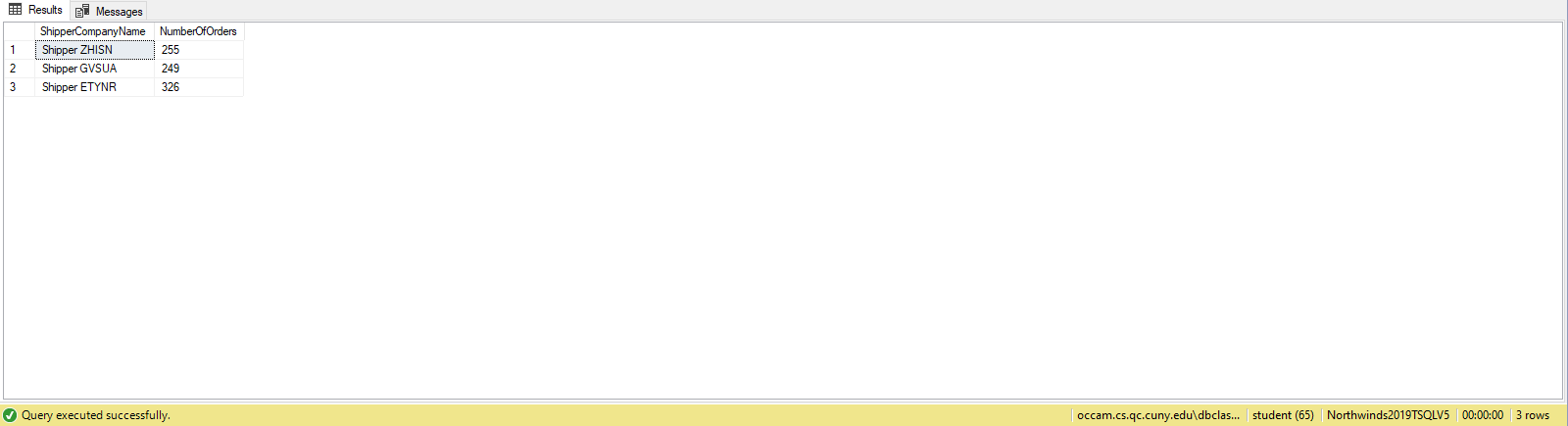
,COUNT(a.OrderID) AS NumberOfOrders

FROM [Sales].[Order] AS a

LEFT JOIN Sales.Shipper AS b ON a.ShipperID = b.ShipperID

GROUP BY ShipperCompanyName;

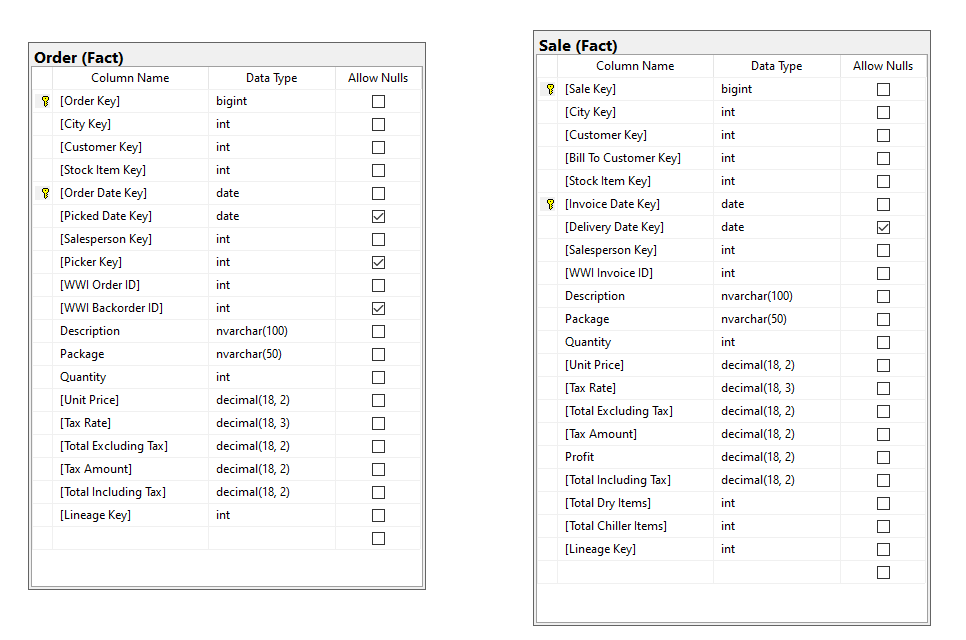
Sample Output (3 Rows Returned)



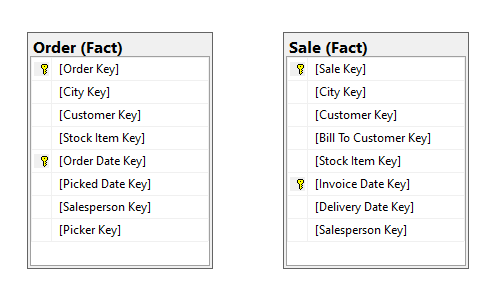
Problem 08: Find the total number of orders placed of each stock item using WideWorldImportersDW?

Explanation: This Query uses the built in function to count all the orders placed for each stock item in the year 2016.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Fact.Order | * Total Orders |
| Fact.Sale | * Stock Item Key |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Fact.Sale | StockItemKey | ASC |

Problem Solving Query

USE WideWorldImportersDW

SELECT SA.[Stock Item Key]

,COUNT(OD.[Order Key]) AS TotalOrders

FROM [Fact].[Order] AS OD

INNER JOIN [Fact].[Sale] AS SA ON OD.[Stock Item Key] = SA.[Stock Item Key]

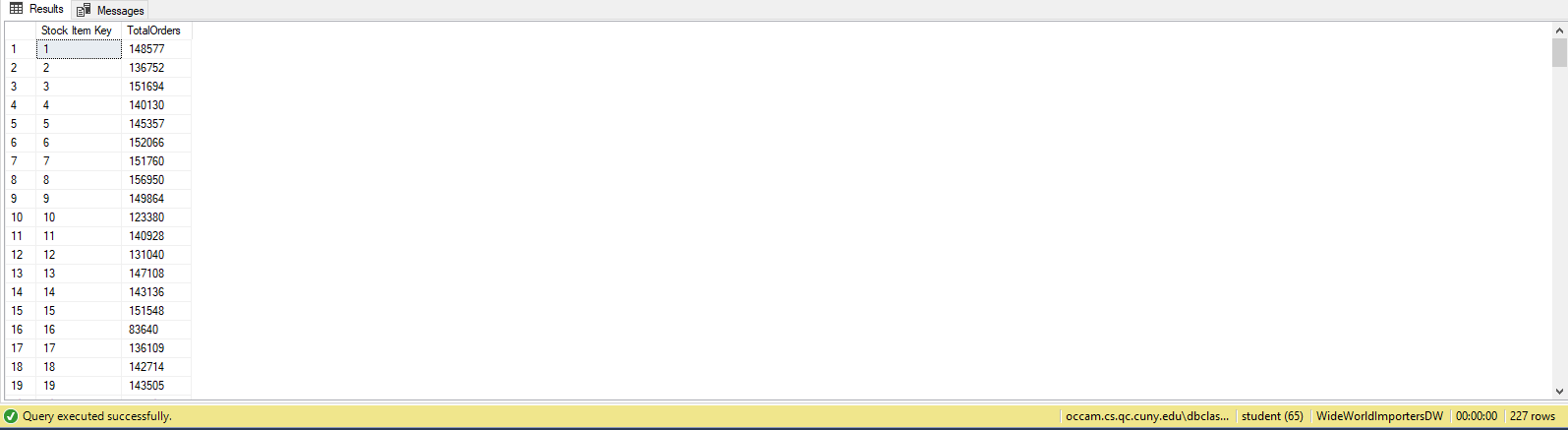
WHERE OD.[Order Date Key] >= '2016-01-01'

AND OD.[Order Date Key] < '2017-01-01'

GROUP BY SA.[Stock Item Key]

ORDER BY SA.[Stock Item Key]

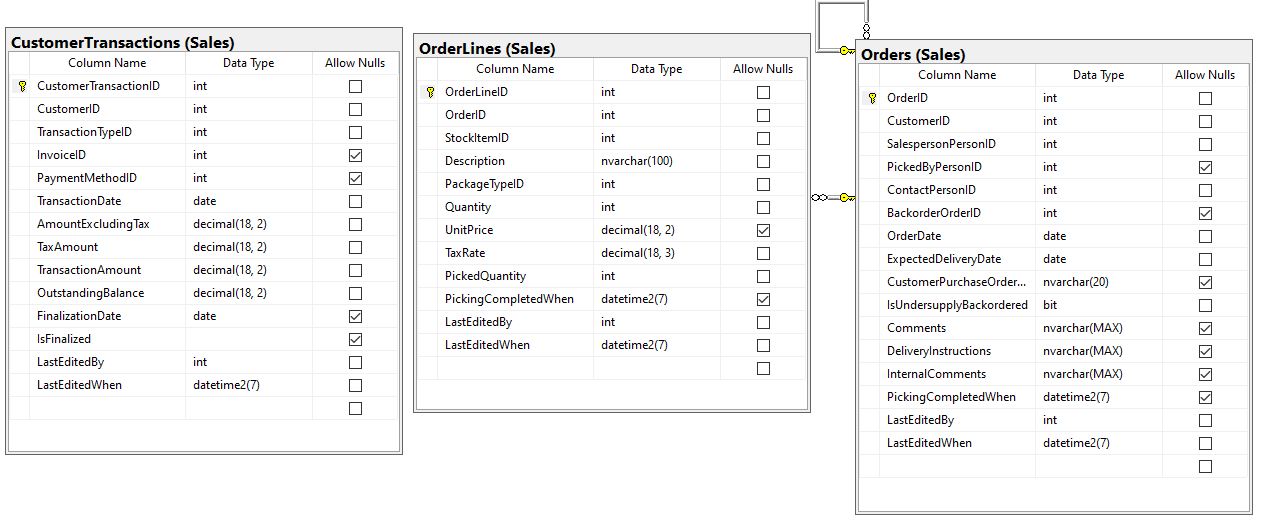
Sample Output (227 Rows Returned)



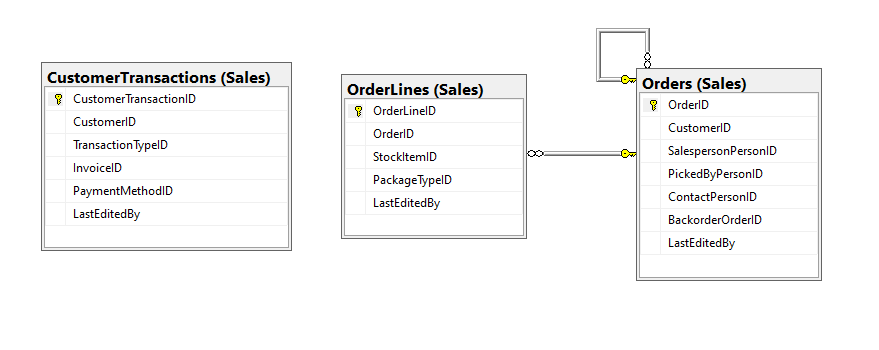
Problem 09: Find the revenue WideWorldImporters made from each item in 2013 using WideWorldImporters?

Explanation: This Query shows all the revenue meaning the sum of the transaction amount made from each stock item in the year 2013. We use the built in function called SUM for this.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.CustomerTransactions | * Revenue |
| Sales.Orders |  |
|  |  |
| Sales.OrderLines | * StockItemID * Description |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.OrderLines | StockItemID | ASC |

Problem Solving Query

USE WideWorldImporters

SELECT ODL.StockItemID

,ODL.Description

,SUM(T.TransactionAmount) AS Revenue

FROM [Sales].[CustomerTransactions] AS T

INNER JOIN [Sales].[Orders] AS OD ON T.CustomerID = OD.CustomerID

INNER JOIN [Sales].[OrderLines] AS ODL ON OD.OrderID = ODL.OrderID

WHERE OD.OrderDate >= '2013-01-01'

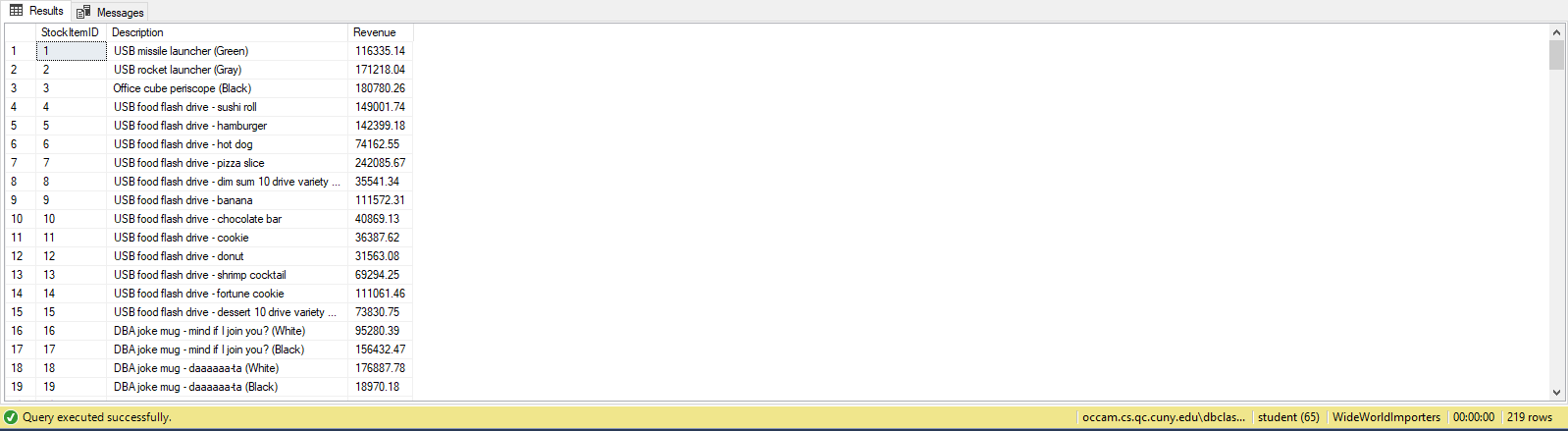
AND OD.OrderDate < '2014-01-01'

GROUP BY ODL.StockItemID

,ODL.Description

ORDER BY ODL.StockItemID

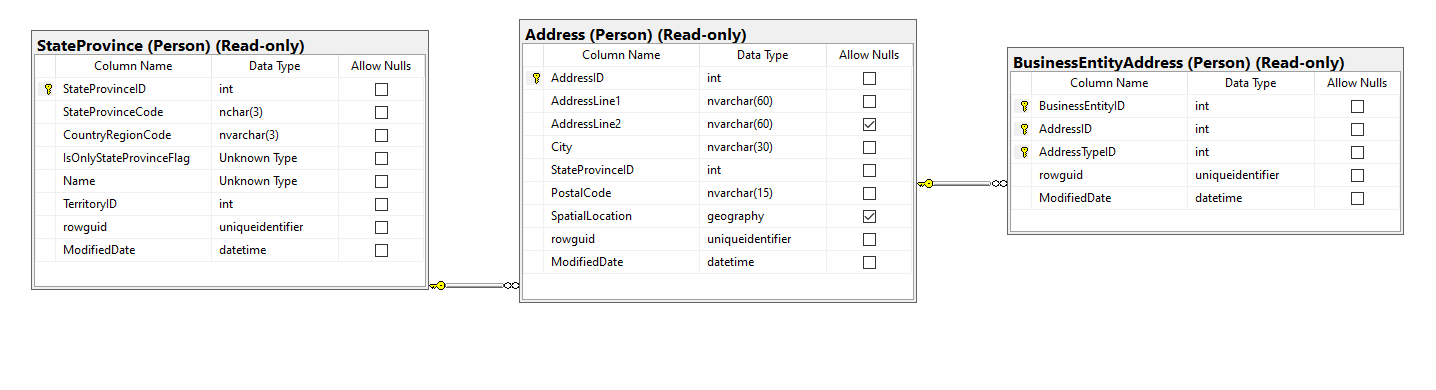
Sample Output (219 Rows Returned)



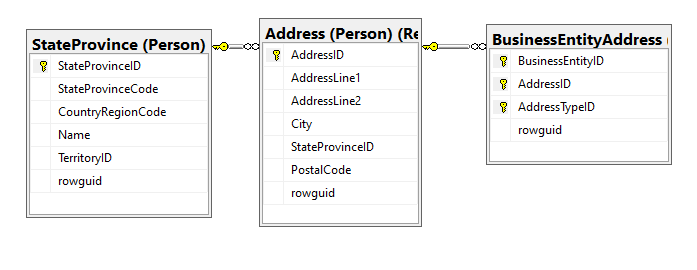
Problem 10: Find the number of businesses in each state of US using AdventureWorks2014?

Explanation: This Query uses Count built in function and outputs the total number of businesses that are open in each state in the USA.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Person.BusinessEntityAddress | * TotalNumberofBusinessEntityIDs |
| Person.Address |  |
|  |  |
| Person.StateProvince | * StateProvinceCode * CountryRegionCode |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Person.StateProvince | StateProvinceCode | ASC |

Problem Solving Query

USE AdventureWorks2014

SELECT c.StateProvinceCode

,c.CountryRegionCode

,COUNT(a.BusinessEntityID) AS TotalNumberOfBusinessEntityIDs

FROM [Person].[BusinessEntityAddress] AS a

FULL OUTER JOIN [Person].[Address] AS b ON a.AddressID = b.AddressID

FULL OUTER JOIN [Person].[StateProvince] AS c ON c.StateProvinceID = b.StateProvinceID

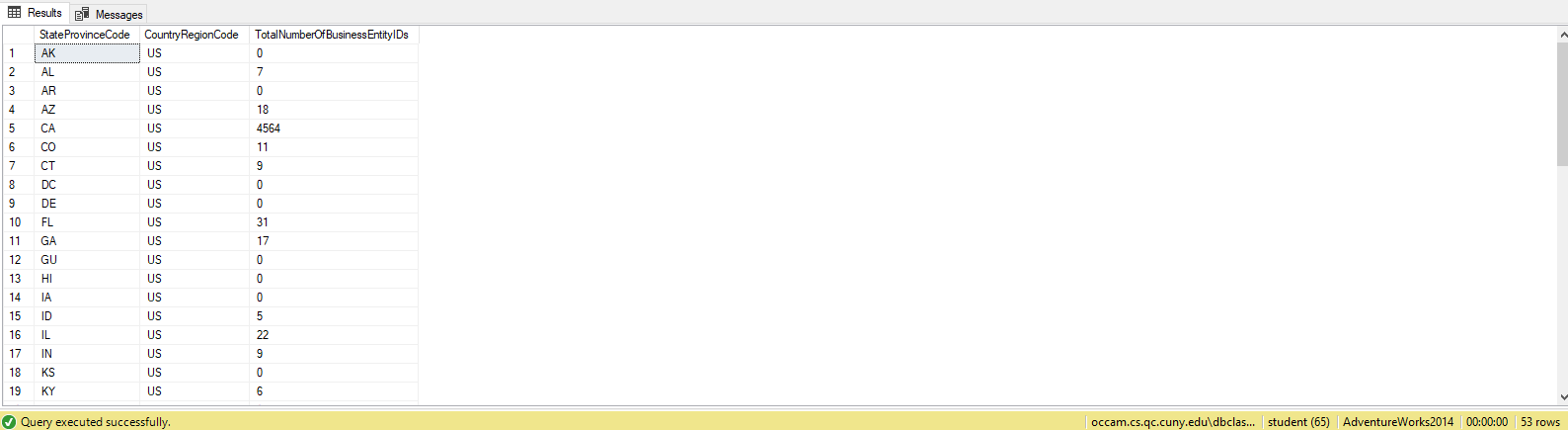
WHERE c.CountryRegionCode = 'US'

GROUP BY c.StateProvinceCode

,c.CountryRegionCode

ORDER BY c.StateProvinceCode

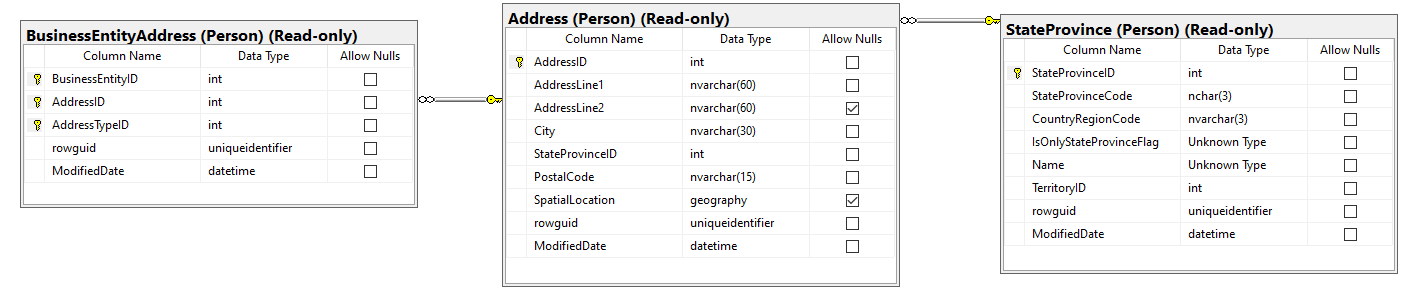
Sample Output (53 Rows Returned)



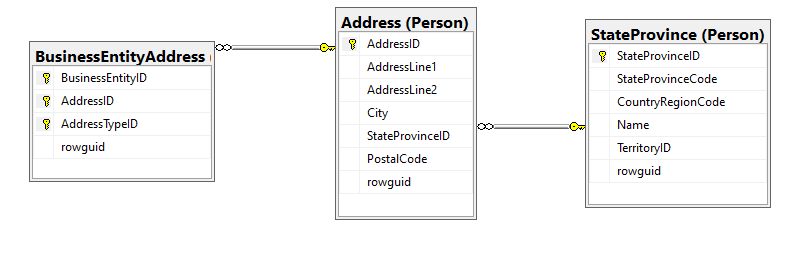
Problem 11: Find the Business ID of the first business that is registered using AdventureWorks2014?

Explanation: This Queries uses MIN function and output the business entity ID of the first business to register in each state ordered by the Country Region Code.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Person.BusinessEntityAddress | * FirstBusinessToRegisterID |
| Person.Address |  |
|  |  |
| Person.StateProvince | * StateProvinceCode * CountryRegionCode |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Person.StateProvince | CountryRegionCode | ASC |

Problem Solving Query

USE AdventureWorks2014

SELECT c.StateProvinceCode

,c.CountryRegionCode

,MIN(a.BusinessEntityID) AS FirstBusinessToRegisterID

FROM [Person].[BusinessEntityAddress] AS a

FULL OUTER JOIN [Person].[Address] AS b ON a.AddressID = b.AddressID

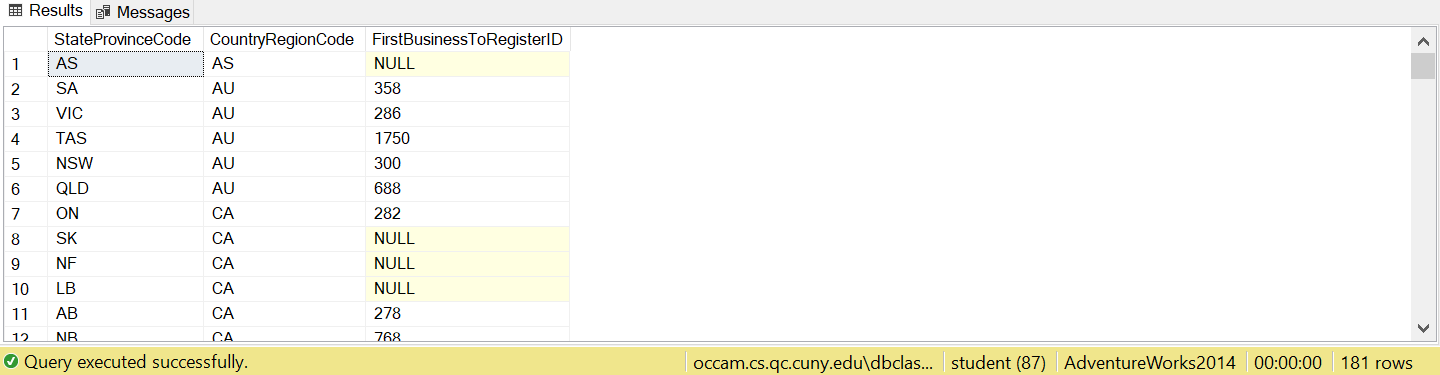
FULL OUTER JOIN [Person].[StateProvince] AS c ON c.StateProvinceID = b.StateProvinceID

GROUP BY c.StateProvinceCode

,c.CountryRegionCode

ORDER BY c.CountryRegionCode

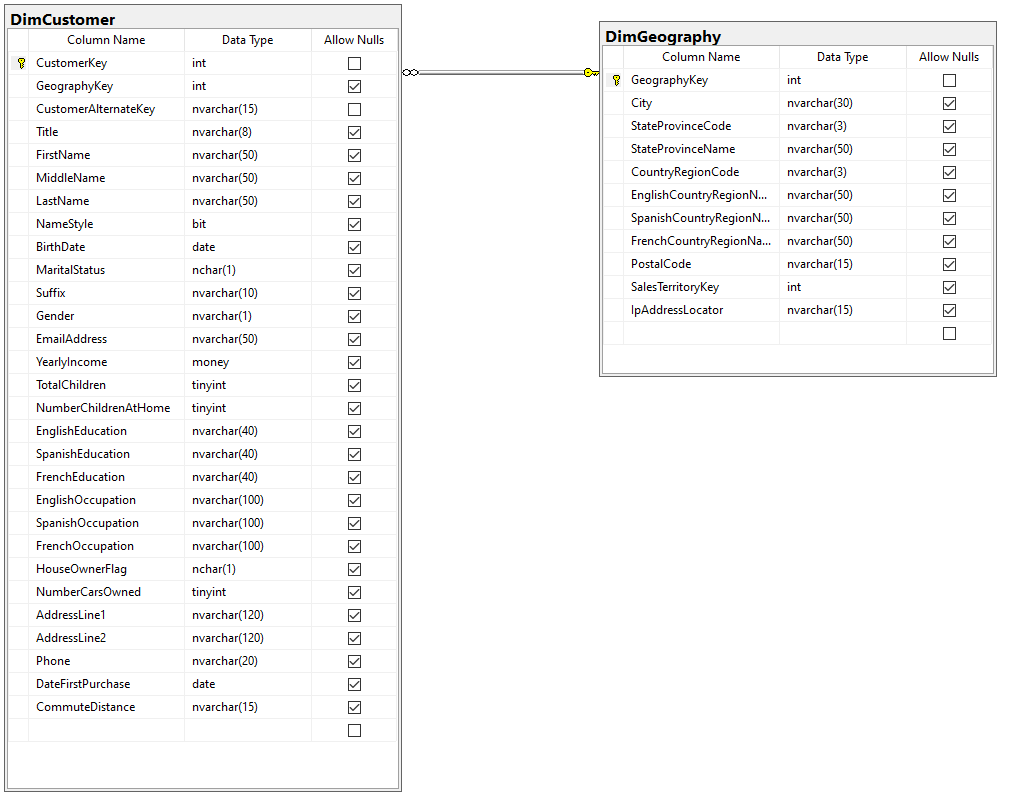
Sample Output (181 Rows Returned)



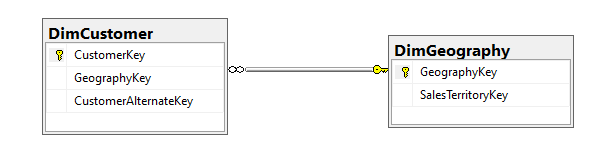
Problem 12: Find the total number of male customers in each country using AdventureWorksDW2016?

Explanation: This Query uses Count built in function to count the number of male customers in each country excluding the null values.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| dbo.DimCustomer | * TotalMaleCustomers |
| dbo.DimGeograpy | * CountryRegionCode * EnglishCountryRegionName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| dbo.DimGeography | EnglishCountryRegionName | ASC |

Problem Solving Query

USE AdventureWorksDW2016

SELECT b.CountryRegionCode

,b.EnglishCountryRegionName

,COUNT(a.GeographyKey) AS TotalMaleCustomers

FROM [dbo].[DimCustomer] AS a

INNER JOIN [dbo].[DimGeography] AS b ON a.GeographyKey = b.GeographyKey

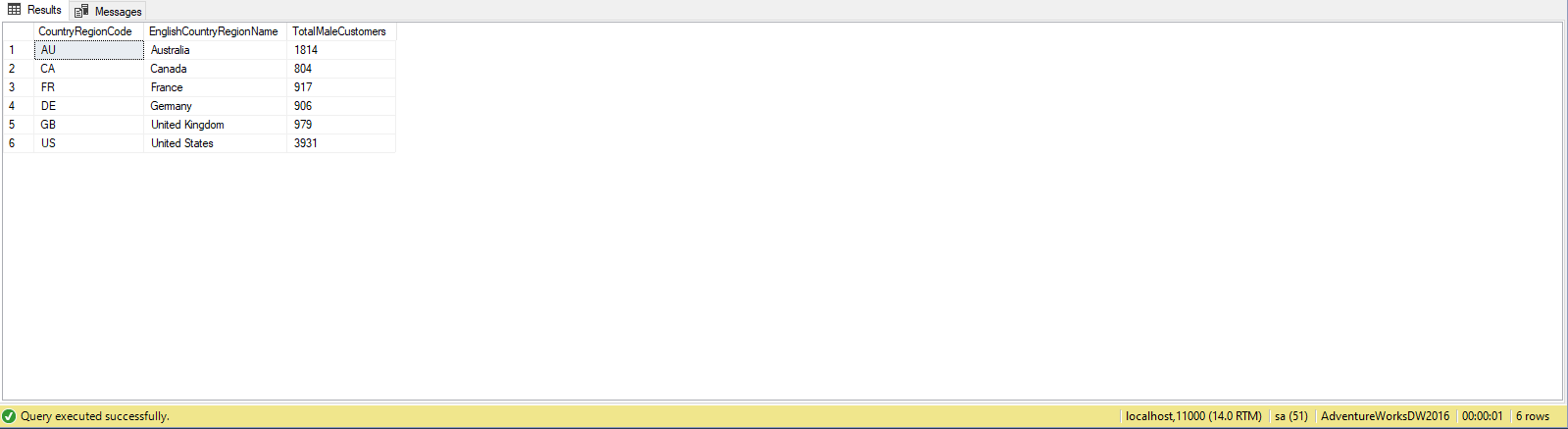
WHERE a.Gender = 'M'

GROUP BY b.CountryRegionCode

,b.EnglishCountryRegionName

ORDER BY b.EnglishCountryRegionName

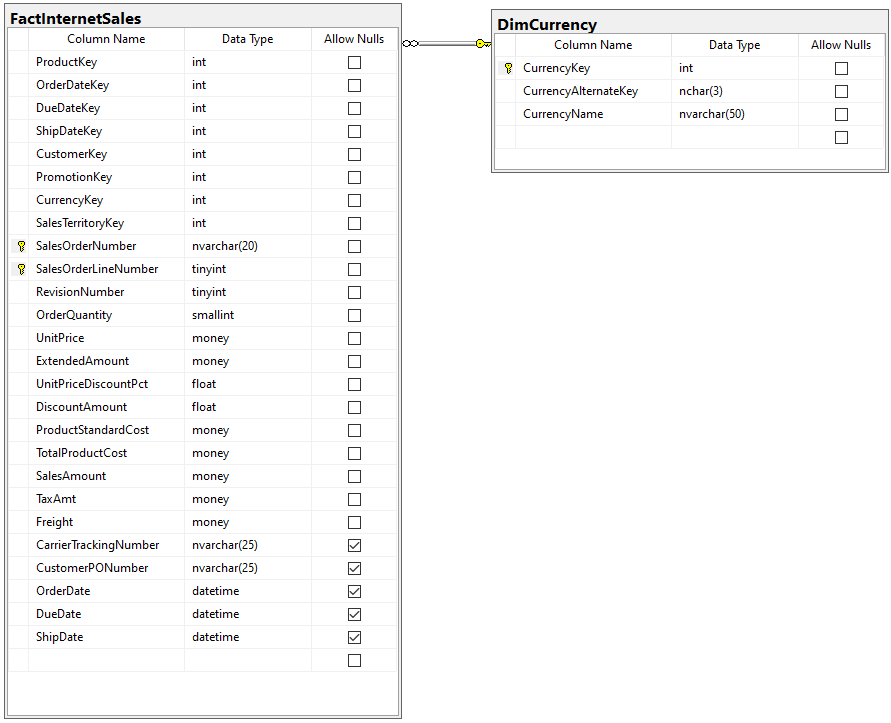
Sample Output (6 Rows Returned)



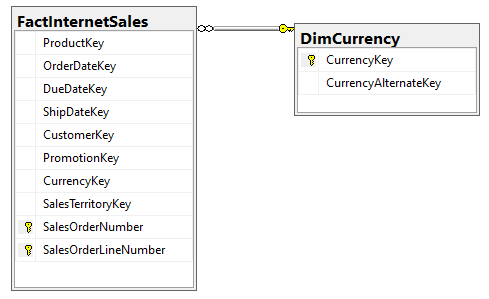
Problem 13: Find the Total number of orders placed during internet sales that used different currencies and don't show the currencies that were never used using AdventureWorksDW2016?

Explanation: This Query Counts the total number of orders placed during the internet sales using different currencies excluding the 0 or null value rows.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| dbo.FactInternetSales | * ToatlInternetSales |
| dbo.DimCurrency | * CurrencyKey * CurrencyName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| dbo.DimCurrency | CurrencyName | ASC |

Problem Solving Query

USE AdventureWorksDW2016

SELECT b.CurrencyKey

,b.CurrencyName

,COUNT(a.ProductKey) AS TotalInternetSales

FROM [dbo].[FactInternetSales] AS a

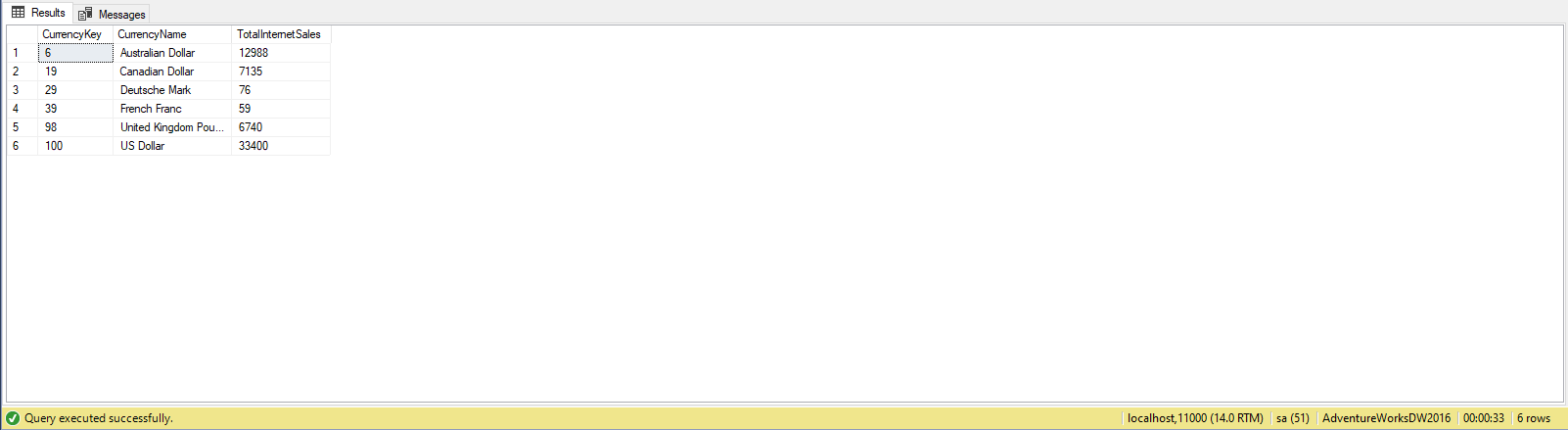
INNER JOIN [dbo].[DimCurrency] AS b ON a.CurrencyKey = b.CurrencyKey

GROUP BY b.CurrencyKey

,b.CurrencyName

ORDER BY b.CurrencyName

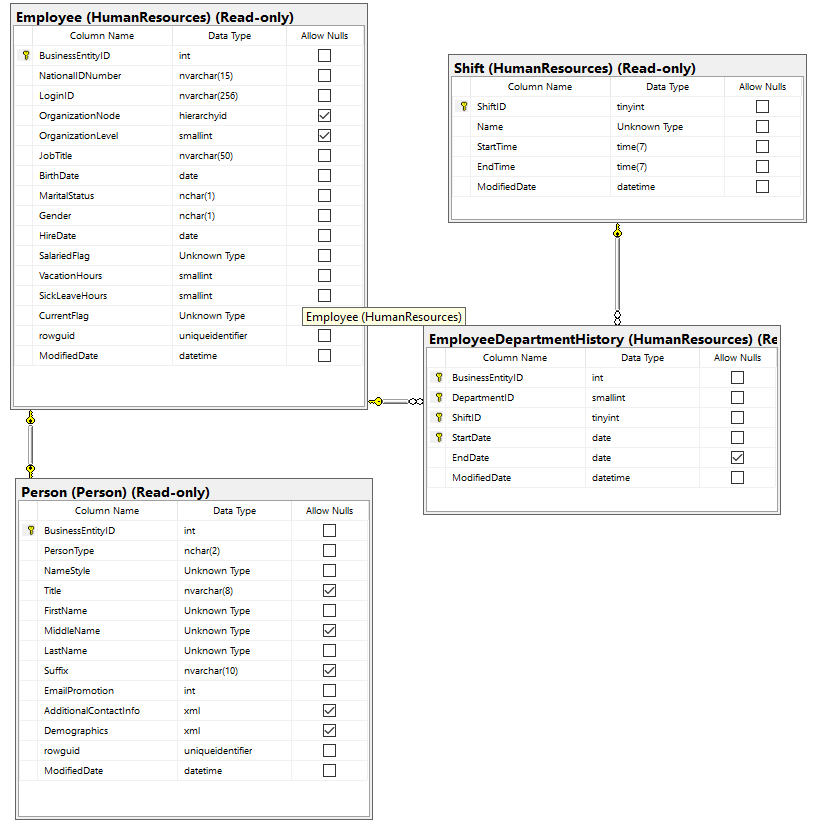
Sample Output 6 Rows Returned)



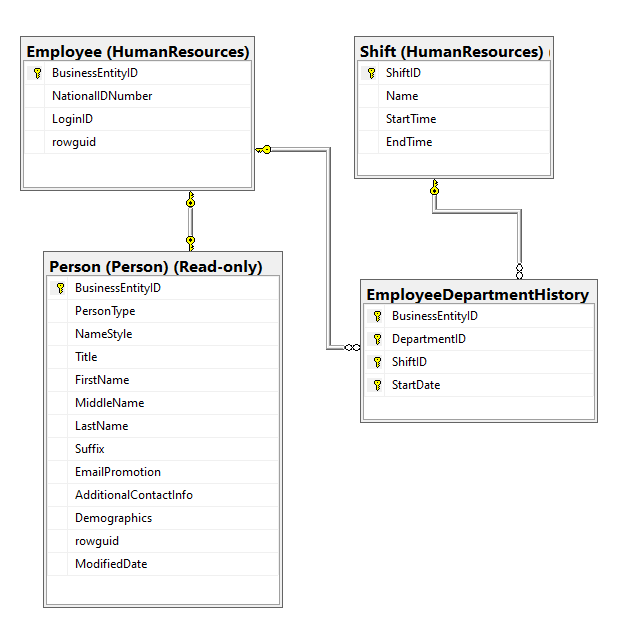
Problem 14: Find the Full name of the male employees and their age when they got hired and who only do day shifts using AdventureWorks2014?

Explanation: This Query uses the scalar function called calculate years which finds the difference in years between two dates. The query itself output the full name of the customer, show how old they were when they got hired and show their shift type.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| HumanResources.Employee | * AgeWhenTheyGotHired |
| Person.Person | * FullName |
| HumanResources.EmployeeDepartmentHistory |  |
| HumanResources.Shift | * ShiftType |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| N/A | N/A | N/A |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.CalculateYears') IS NOT NULL

DROP FUNCTION dbo.CalculateYears;

GO

CREATE FUNCTION dbo.CalculateYears (

@DOS DATE

,--dateofstart

@DOF DATE --dateoffinish

)

RETURNS INT

AS

BEGIN

DECLARE @AGE INT

SET @AGE = DATEDIFF(YEAR, @DOS, @DOF) - CASE

WHEN (MONTH(@DOS) > MONTH(@DOF))

OR (

MONTH(@DOS) = MONTH(@DOF)

AND DAY(@DOS) > DAY(@DOF)

)

THEN 1

ELSE 0

END

RETURN @AGE

END;

GO

--FUNCTION CODE ENDS HERE

USE AdventureWorks2014

SELECT CONCAT (

B.FirstName

,' '

,B.LastName

) AS FullName

,dbo.CalculateYears(convert(NVARCHAR, A.BirthDate, 101), convert(NVARCHAR, A.HireDate, 101)) AS AgeWhenTheyGotHired

,D.[Name] AS ShiftType

FROM [HumanResources].[Employee] AS A

INNER JOIN [Person].[Person] AS B ON A.BusinessEntityID = B.BusinessEntityID

INNER JOIN [HumanResources].[EmployeeDepartmentHistory] AS C ON B.BusinessEntityID = C.BusinessEntityID

INNER JOIN [HumanResources].[Shift] AS D ON C.ShiftID = D.ShiftID

WHERE D.[Name] = 'Day'

AND A.Gender = 'M'

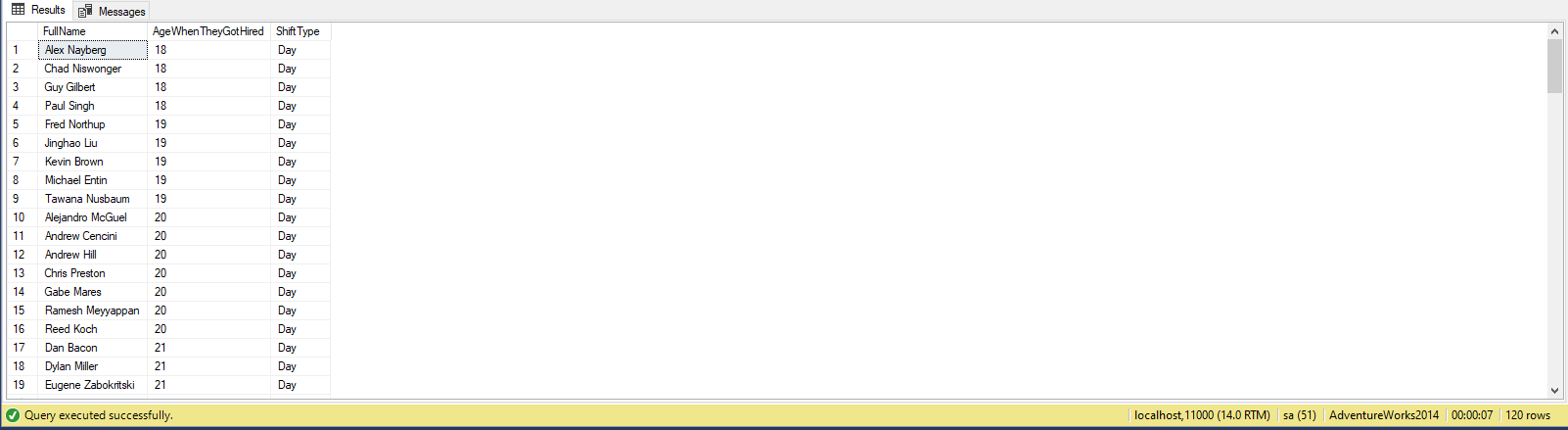
GROUP BY dbo.CalculateYears(convert(NVARCHAR, A.BirthDate, 101), convert(NVARCHAR, A.HireDate, 101))

,B.FirstName

,B.LastName

,D.[Name]

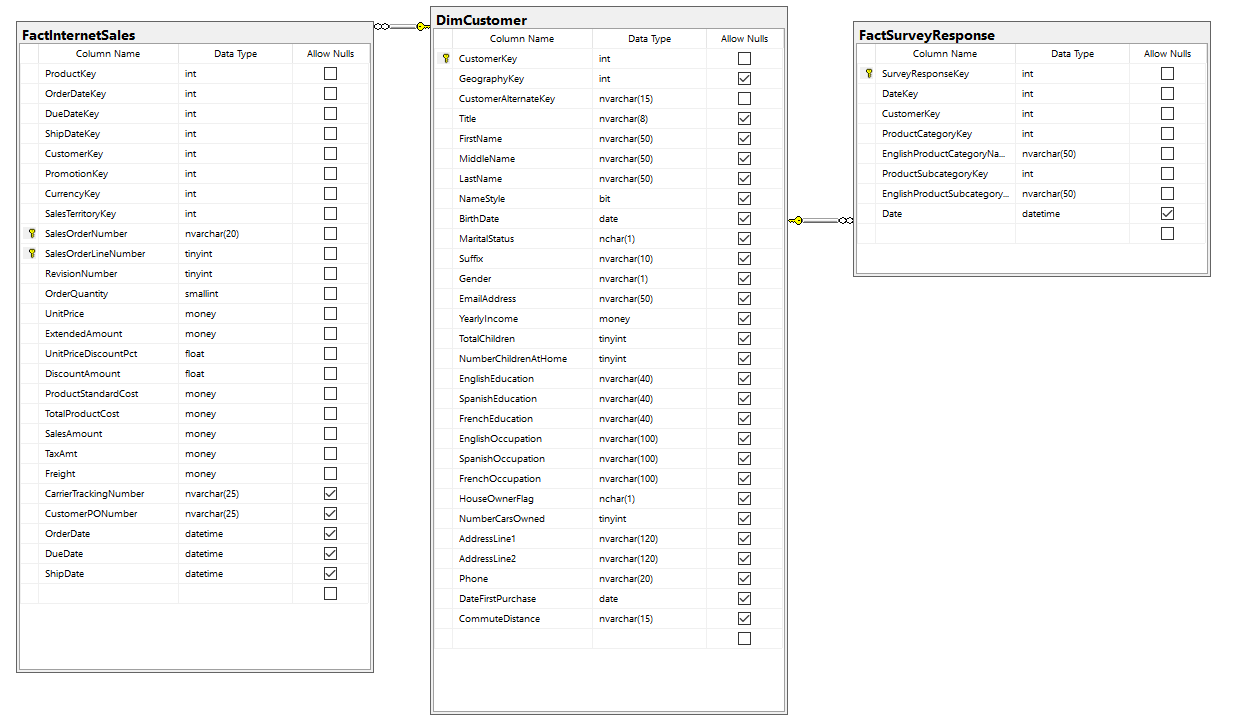
Sample Output (120 Rows Returned)



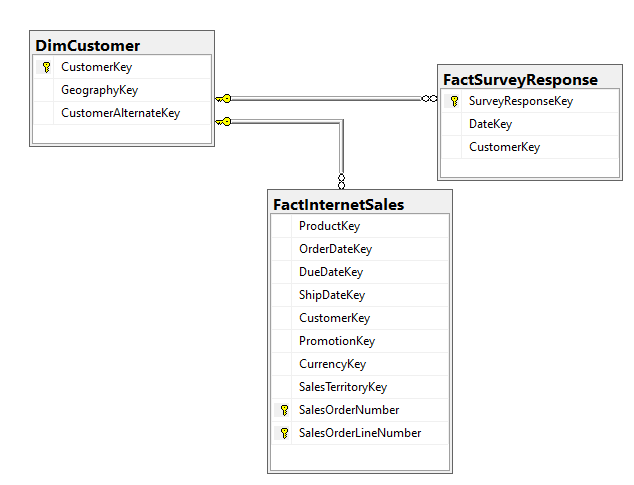
Problem 15: Find the name of the Customers, number of clothing internet sales order they placed and also count the number of years since they made their first purchase up until today using AdventureWorksDW2016?

Explanation: This Query shows the full name by using Concat function, show how many years its been since they made their first purchase, shows the total number of orders placed during internet sales and products they ordered.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| dbo.DimCustomer | * CustomerKey * FullName * YearsSinceTheirFirstPurchase |
| dbo.FactInternetSales | * TotalInternetSalesOrder |
| dbo.FactSurveyResponse | * EnglishProductSubcategoryName |

Query “ORDER BY” Projections

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

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.CalculateYears') IS NOT NULL

DROP FUNCTION dbo.CalculateYears;

GO

CREATE FUNCTION dbo.CalculateYears (

@DOS DATE

,--dateofstart

@DOF DATE --dateoffinish

)

RETURNS INT

AS

BEGIN

DECLARE @AGE INT

SET @AGE = DATEDIFF(YEAR, @DOS, @DOF) - CASE

WHEN (MONTH(@DOS) > MONTH(@DOF))

OR (

MONTH(@DOS) = MONTH(@DOF)

AND DAY(@DOS) > DAY(@DOF)

)

THEN 1

ELSE 0

END

RETURN @AGE

END;

GO

--FUNCTION CODE ENDS HERE

USE AdventureWorksDW2016

SELECT A.CustomerKey

,CONCAT (

A.FirstName

,' '

,A.LastName

) AS FullName

,dbo.CalculateYears(A.DateFirstPurchase, getDate()) AS YearsSinceTheirFirstPurchase

,COUNT(B.OrderQuantity) TotalInternetSalesOrder

,C.EnglishProductSubcategoryName

FROM [dbo].[DimCustomer] AS A

INNER JOIN [dbo].[FactInternetSales] AS B ON A.CustomerKey = B.CustomerKey

INNER JOIN [dbo].[FactSurveyResponse] AS C ON B.CustomerKey = C.CustomerKey

WHERE C.EnglishProductCategoryName = 'Clothing'

GROUP BY A.CustomerKey

,dbo.CalculateYears(A.DateFirstPurchase, getDate())

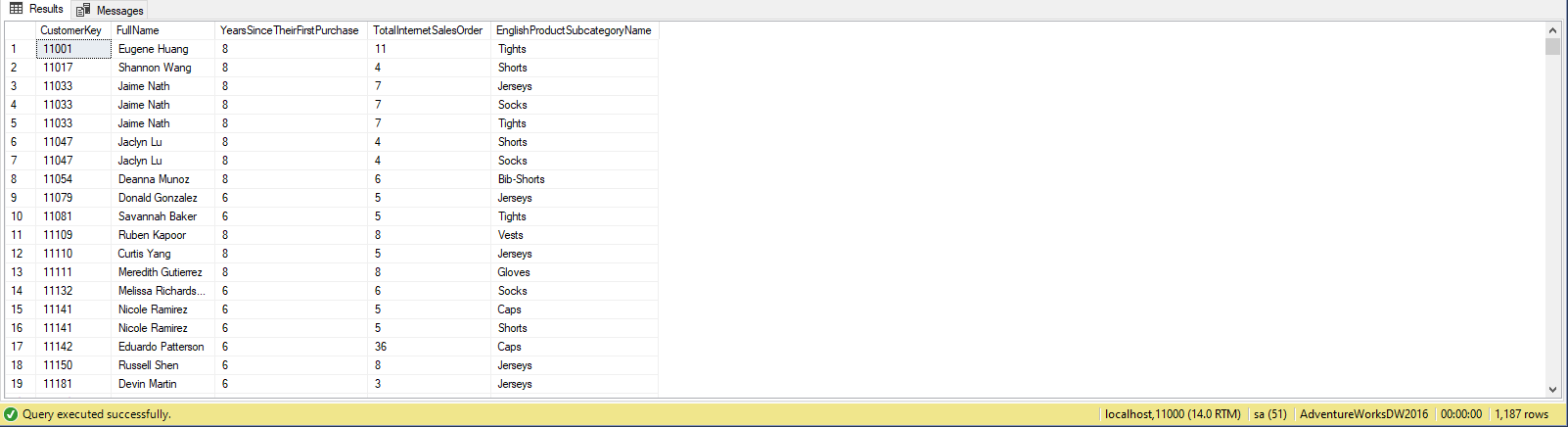
,A.FirstName

,A.LastName

,C.EnglishProductSubcategoryName

ORDER BY A.CustomerKey

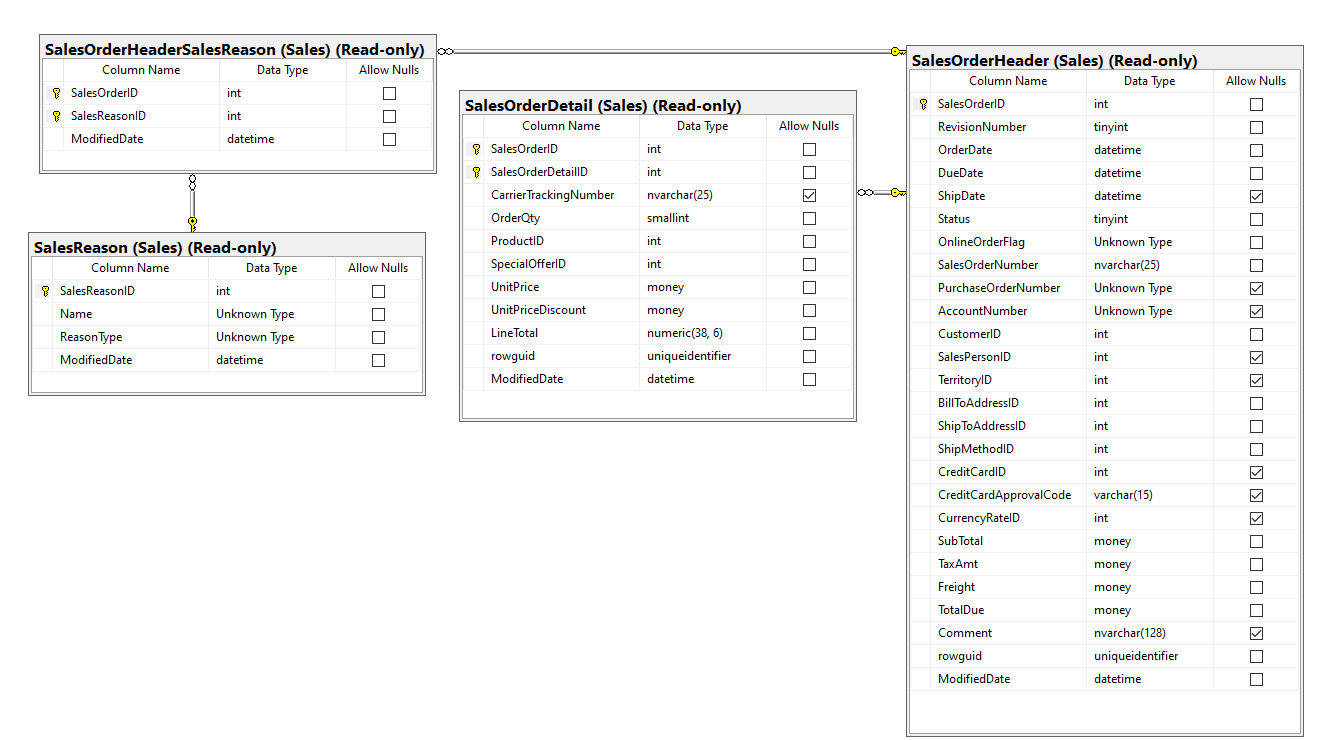
Sample Output (1187 Rows Returned)



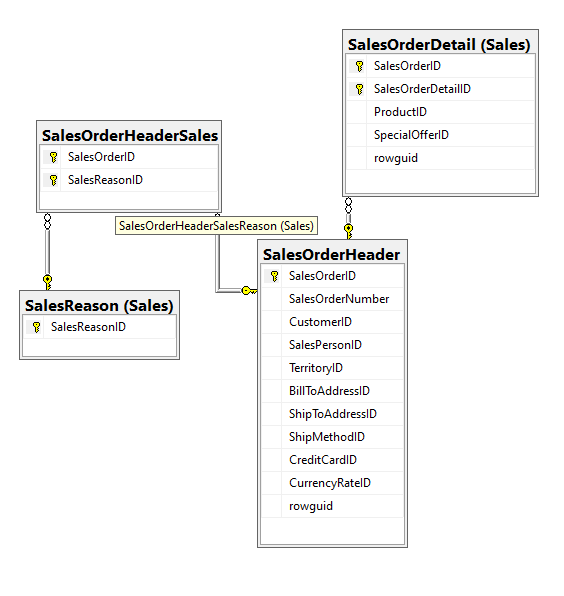
Problem 16: Find the sales total order for each order including tax and freight cost and also find the reason for sales using AdventureWorks2014?

Explanation: This Query uses the scalar function called CalculateSalesOrderTotal which takes the LineTotal, TaxAmt and Freight Amount. The query itself shows salesorderID and the reason why the product was on sale excluding the rows which has reason type “other”.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.SalesOrderDetail | * SalesOrderID * Uses LineTotal |
| Sales.SalesOrderHeader | * Uses TaxAmt, Freight |
| Sales.SalesOrderHeaderSalesReason |  |
| Sales.SalesReason | * ReasonType |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.SalesOrderDetail | SalesOrderID | ASC |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.CalculateSalesOrderTotal') IS NOT NULL

DROP FUNCTION dbo.CalculateSalesOrderTotal;

GO

CREATE FUNCTION dbo.CalculateSalesOrderTotal (

@LineTotal INT

,@TaxAmt INT

,@Freight INT

)

RETURNS MONEY

BEGIN

DECLARE @SalesOrderTotal AS MONEY

SELECT @SalesOrderTotal = @LineTotal + @TaxAmt + @Freight

RETURN @SalesOrderTotal;

END;

GO

--FUNCTION CODE ENDS HERE

USE AdventureWorks2014

SELECT OD.SalesOrderID

,dbo.CalculateSalesOrderTotal(OD.LineTotal, OH.TaxAmt, OH.Freight) AS SalesOrderTotal

,SR.ReasonType

FROM [Sales].[SalesOrderDetail] AS OD

FULL OUTER JOIN [Sales].[SalesOrderHeader] AS OH ON OD.SalesOrderID = OH.SalesOrderID

INNER JOIN [Sales].[SalesOrderHeaderSalesReason] AS OHSR ON OD.SalesOrderID = OHSR.SalesOrderID

INNER JOIN [Sales].[SalesReason] AS SR ON OHSR.SalesReasonID = SR.SalesReasonID

WHERE SR.ReasonType != 'OTHER'

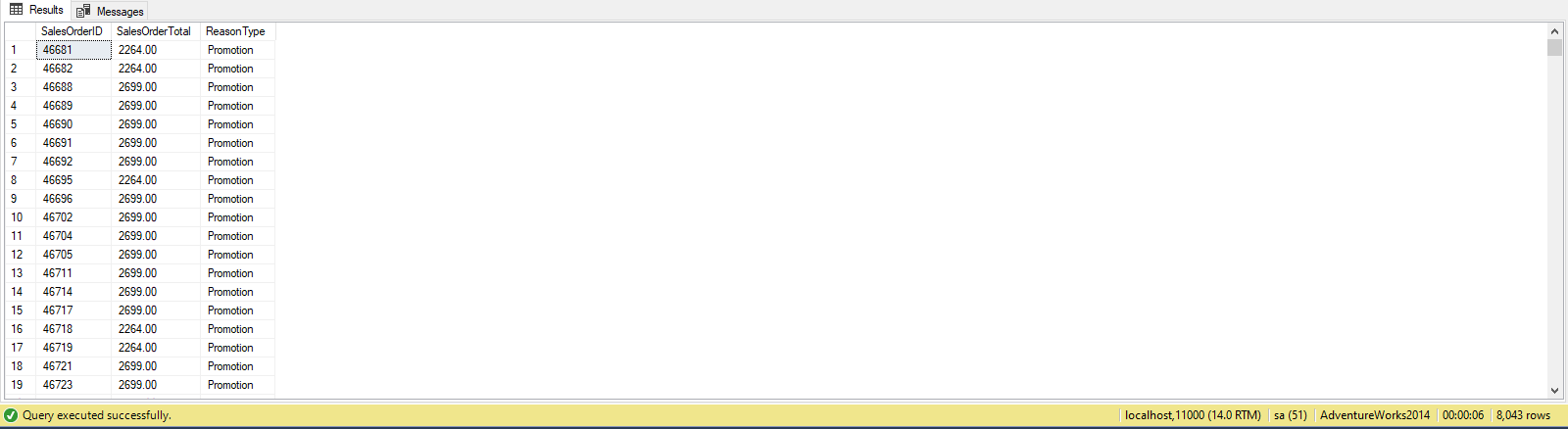
GROUP BY OD.SalesOrderID

,dbo.CalculateSalesOrderTotal(OD.LineTotal, OH.TaxAmt, OH.Freight)

,SR.ReasonType

ORDER BY OD.SalesOrderID

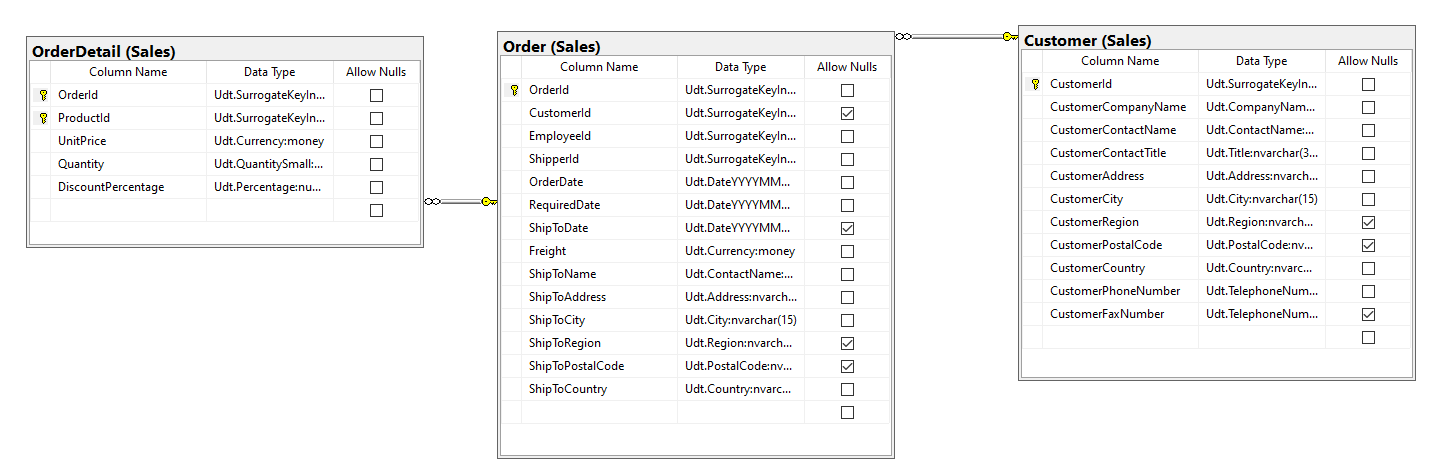
Sample Output (8043 Rows Returned)



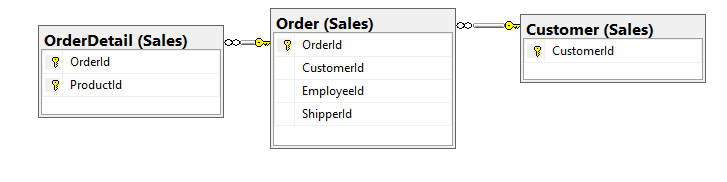
Problem 17: Find the fiscal year quarter for order date for each customer the first quarter beginning with july. also find the total freight, total line cost and total cost customer paid after discounts using Northwinds2019TSQLV5?

Explanation: This Query shows the total number of order a company placed, displays the order date in fiscal calendar year format and calculates the total line cost, total freight amt, and total after discount.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Customer | * CustomerCompanyName |
| Sales.Order | * FiscalYearQuarter * Uses Freight column |
| Sales.OrderDetail | * Uses UnitPrice, Quantity and DiscountPercentage column |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Customer | CustomerCompanyName | ASC |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.fiscalCalendarYear') IS NOT NULL

DROP FUNCTION dbo.fiscalCalendarYear;

GO

CREATE FUNCTION dbo.fiscalCalendarYear (@date DATE)

RETURNS NVARCHAR(25)

AS

BEGIN

DECLARE @getYear INT

,@getMonth INT

,@getDay INT

,@output1 NVARCHAR(20)

,@output2 NVARCHAR(20)

SET @getYear = YEAR(@date)

SET @getMonth = MONTH(@date)

SET @getDay = DAY(@date)

SELECT @output1 = CASE

WHEN (

@getMonth >= 7

AND @getMonth <= 9

)

THEN '-Quarter I'

WHEN (

@getMonth >= 10

AND @getMonth <= 12

)

THEN '-Quarter II'

WHEN (

@getMonth >= 1

AND @getMonth <= 3

)

THEN '-Quarter III'

WHEN (

@getMonth >= 4

AND @getMonth <= 6

)

THEN '-Quarter IV'

ELSE '0'

END

SET @output2 = @output1

SET @output1 = CAST(@getYear AS NVARCHAR(20)) + @output2

RETURN @output1

END;

GO

--FUNCTION CODE ENDS HERE

USE Northwinds2019TSQLV5

SELECT C.CustomerCompanyName

,dbo.fiscalCalendarYear(O.OrderDate) AS FiscalYearQuarter

,COUNT(DISTINCT O.OrderId) AS TotalNumberOfOrders

,'$ ' + CONVERT(NVARCHAR(20), SUM(O.Freight)) AS [TotalFreight]

,'$ ' + CONVERT(NVARCHAR(20), SUM(OD.UnitPrice \* OD.Quantity)) AS TotalLineCost

,'$ ' + CONVERT(NVARCHAR(20), SUM((OD.UnitPrice \* OD.Quantity) \* (1. - OD.DiscountPercentage))) AS TotalDicountedLineCost

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O ON O.CustomerId = C.CustomerId

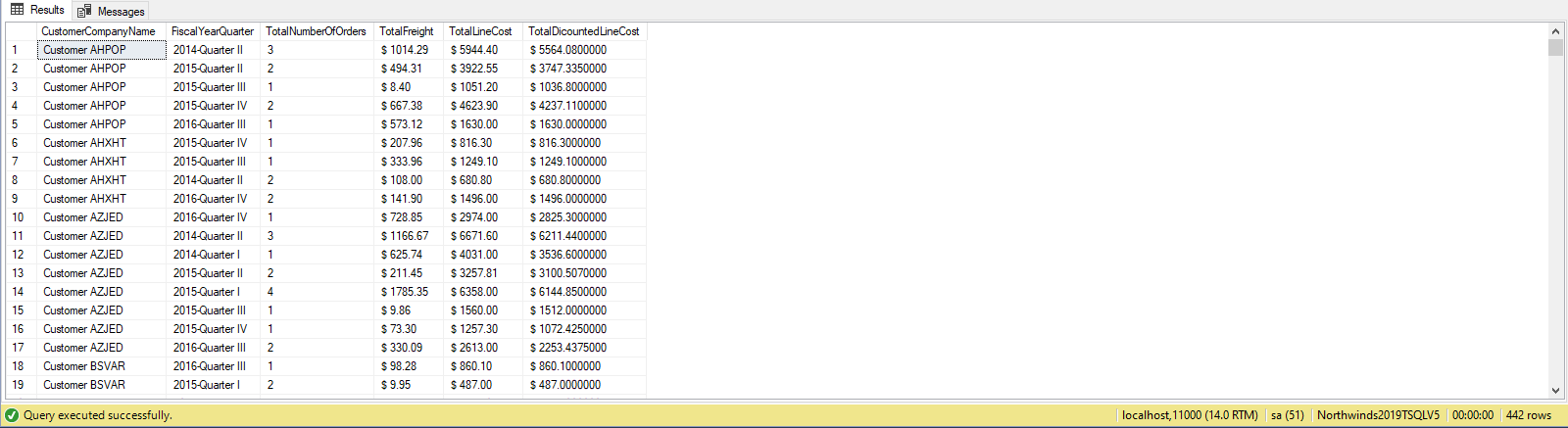
INNER JOIN Sales.OrderDetail AS OD ON OD.OrderId = O.OrderId

GROUP BY C.CustomerCompanyName

,dbo.fiscalCalendarYear(O.orderdate)

ORDER BY C.CustomerCompanyName

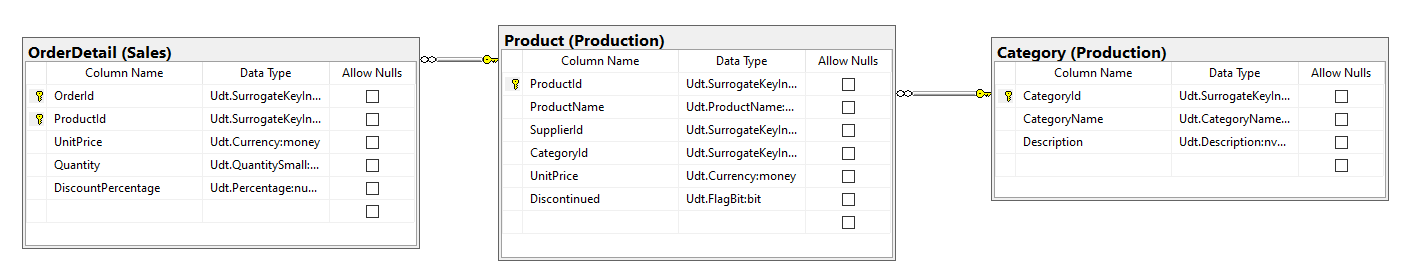
Sample Output (442 Rows Returned)



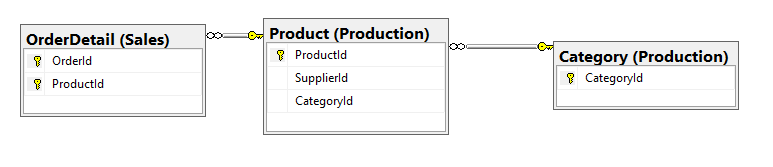
Problem 18: Find the orders total after discount and show which ones are discounted, also show the name of the product and category it belongs to using Northwinds2019TSQLV5?

Explanation: This Query uses the scalar function called “IsSomethingTrueOrFalse”, it basically return two values yes or no. the output consists of order id, product id, product name, category name, total after discount and the last column shows if there was even any discount for that product.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.OrderDetail | * OrderID * TotalDiscountedLineCost * IsDiscounted |
| Production.Product | * ProductId * ProductName |
| Production.Category | * CategoryName |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Orderdetail | OrderID | ASC |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.isSomethingTrueOrFalse') IS NOT NULL

DROP FUNCTION dbo.isSomethingTrueOrFalse;

GO

CREATE FUNCTION dbo.isSomethingTrueOrFalse (@input NVARCHAR(255))

RETURNS NVARCHAR(255)

AS

BEGIN

DECLARE @output NVARCHAR(255)

SELECT @output = CASE

WHEN (@input = '0.000')

THEN 'No'

ELSE 'Yes'

END

RETURN @output

END;

GO

--FUNCTION CODE ENDS HERE

USE Northwinds2019TSQLV5

SELECT OD.OrderId

,P.ProductId

,P.ProductName

,C.CategoryName

,'$ ' + CONVERT(NVARCHAR(20), SUM((OD.UnitPrice \* OD.Quantity) \* (1. - OD.DiscountPercentage))) AS TotalDicountedLineCost

,dbo.isSomethingTrueOrFalse(OD.DiscountPercentage) AS IsDiscounted

FROM [Sales].[OrderDetail] AS OD

INNER JOIN [Production].[Product] AS P ON OD.ProductId = P.ProductId

INNER JOIN [Production].[Category] AS C ON P.CategoryId = C.CategoryId

GROUP BY OD.OrderId

,P.ProductId

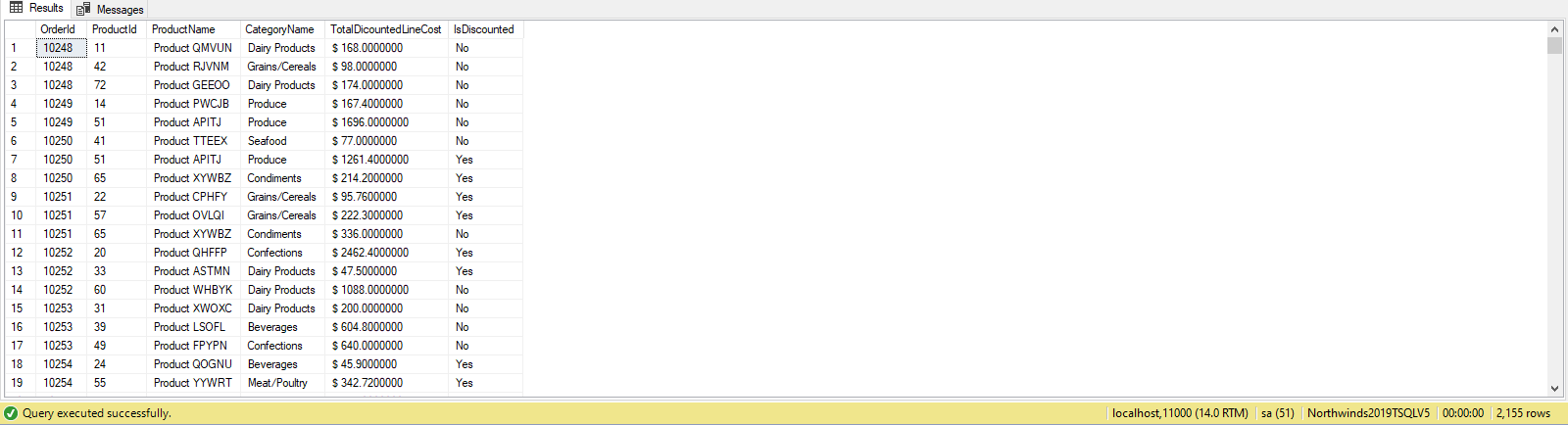
,OD.DiscountPercentage

,C.CategoryName

,P.ProductName

ORDER BY OD.OrderId

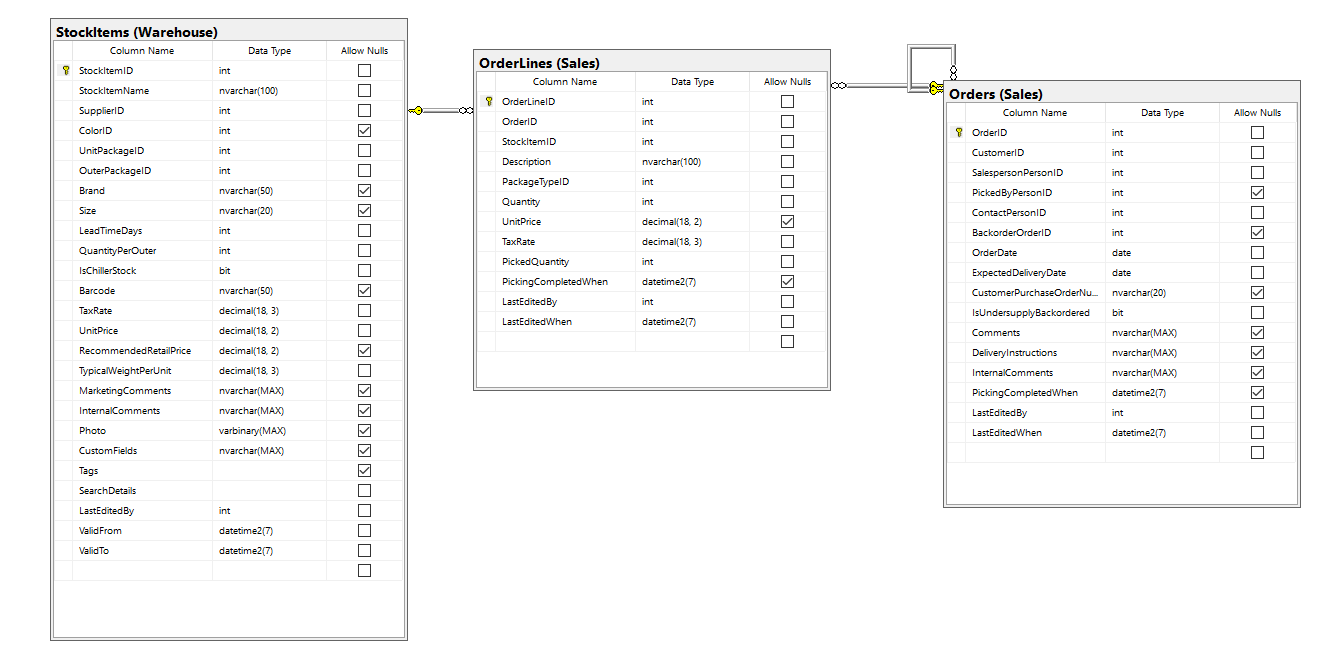
Sample Output (2155 Rows Returned)



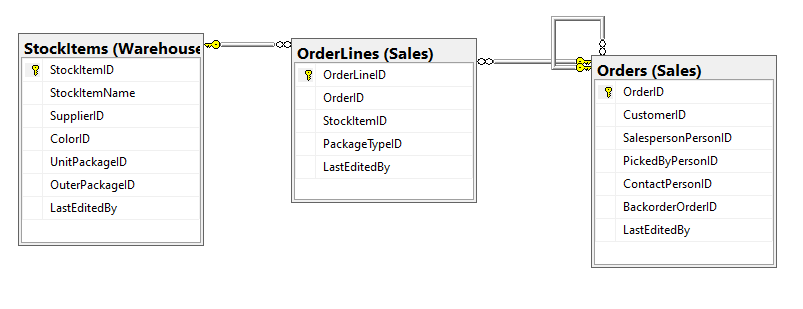
Problem 19: Find all the orders that were placed in 2013 and show the order date in form of fiscal year quarters with January as the 1st month for first quarter and also calculate the total amount paid for each order ID including taxes using WideWorldImporters?

Explanation: This Query outputs the order id, order date in fiscal quarter year format and the total amount for each order including taxes in the year 2013.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Sales.Orders | * OrderID * FiscalOrderDate |
| Sales.OrderLines | * AmountPaindIncludingtax |
| Warehouse.StockItems |  |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Sales.Orders | OrderID | ASC |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.fiscalCalendarYear') IS NOT NULL

DROP FUNCTION dbo.fiscalCalendarYear;

GO

CREATE FUNCTION dbo.fiscalCalendarYear (@date DATE)

RETURNS NVARCHAR(25)

AS

BEGIN

DECLARE @getYear INT

,@getMonth INT

,@getDay INT

,@output1 NVARCHAR(20)

,@output2 NVARCHAR(20)

SET @getYear = YEAR(@date)

SET @getMonth = MONTH(@date)

SET @getDay = DAY(@date)

SELECT @output1 = CASE

WHEN (

@getMonth >= 1

AND @getMonth <= 3

)

THEN '-Quarter I'

WHEN (

@getMonth >= 4

AND @getMonth <= 6

)

THEN '-Quarter II'

WHEN (

@getMonth >= 7

AND @getMonth <= 9

)

THEN '-Quarter III'

WHEN (

@getMonth >= 10

AND @getMonth <= 12

)

THEN '-Quarter IV'

ELSE '0'

END

SET @output2 = @output1

SET @output1 = CAST(@getYear AS NVARCHAR(20)) + @output2

RETURN @output1

END;

GO

--FUNCTION CODE ENDS HERE

USE WideWorldImporters

SELECT DISTINCT O.OrderID

,dbo.fiscalCalendarYear(O.OrderDate) AS FiscalOrderDate

,'$ ' + CONVERT(NVARCHAR(255), SUM((OL.UnitPrice) \* (1 + (OL.TaxRate / 100)))) AS AmountPaidIncludingTax

FROM [Sales].[Orders] AS O

INNER JOIN [Sales].[OrderLines] AS OL ON OL.OrderID = O.OrderID

INNER JOIN [Warehouse].[StockItems] AS S ON OL.StockItemID = S.StockItemID

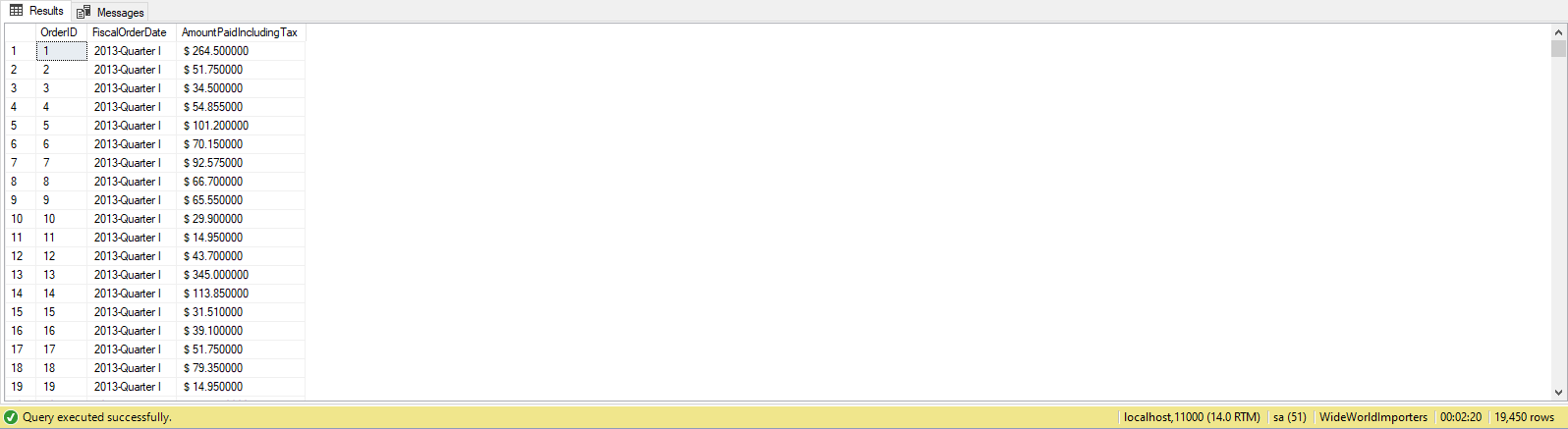
WHERE YEAR(O.OrderDate) = '2013'

GROUP BY O.OrderID

,dbo.fiscalCalendarYear(O.OrderDate)

ORDER BY OrderID

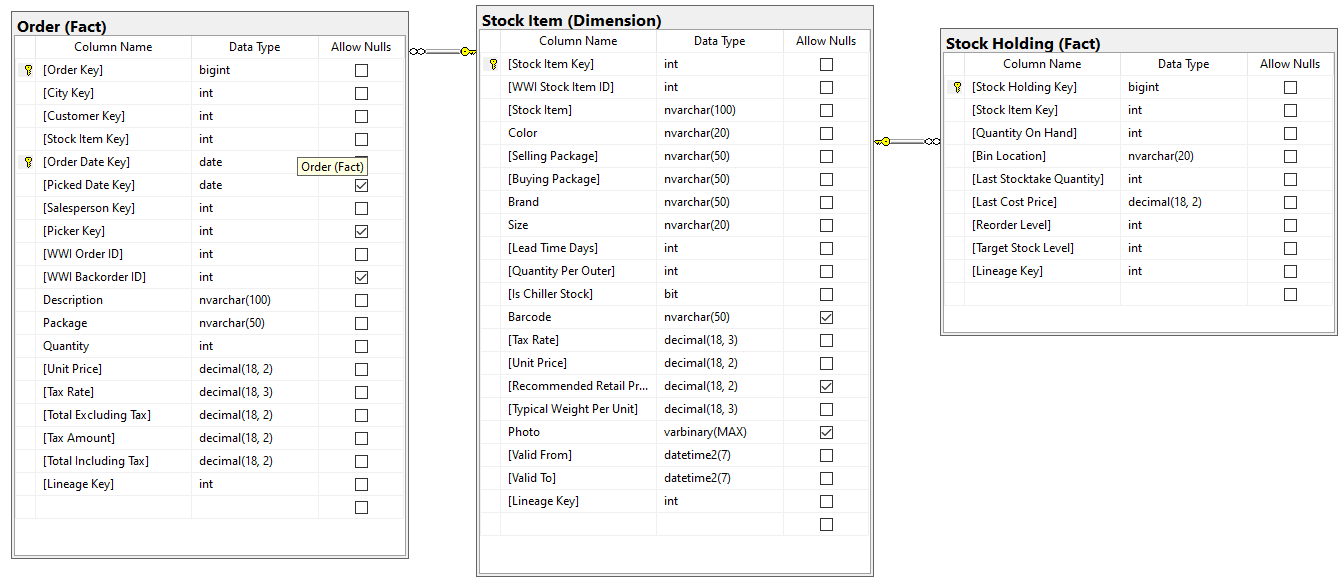
Sample Output (19450 Rows Returned)



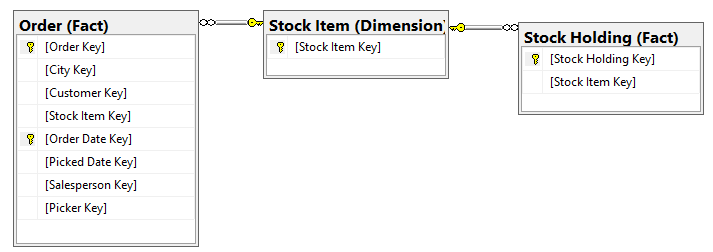
Problem 20: Find the total quantities of each item that was ordered and also the tax amount for one quantity using WideWorldImportersDW?

Explanation: This Query uses the scalar function called taxonONE which the counts the tax amount charged on each quantity. The output shows us all the purchases made by every customer except customer with key 0.

Standard View:



Key View:



Query “SELECT” Projections

|  |  |
| --- | --- |
| Table Name | Column Name |
| Fact.Order | * TaxOneachQuantity |
| Fact.Stock holding |  |
| Dimension.Stock Item | * Stock Item Key * Stock Item * TotalNumberOrdered |

Query “ORDER BY” Projections

|  |  |  |
| --- | --- | --- |
| Table Name | Column Name | Sort Order |
| Dimension.Stock Item | Stock Item Key | ASC |

Problem Solving Query

--FUNCTION CODE BEGINS

IF OBJECT\_ID('dbo.taxonONE') IS NOT NULL

DROP FUNCTION dbo.taxonONE;

GO

CREATE FUNCTION dbo.taxonONE (

@unitprice NUMERIC(38, 2)

,@taxrate NUMERIC(38, 2)

)

RETURNS NUMERIC(38, 2)

AS

BEGIN

DECLARE @output NUMERIC(38, 2);

SET @output = (@unitprice \* (@taxrate / 100));

RETURN @output

END;

GO

--FUNCTION CODE ENDS HERE

USE WideWorldImportersDW

SELECT SI.[Stock Item Key]

,SI.[Stock Item]

,COUNT(SI.[Stock Item Key]) AS TotalNumberOrdered

,dbo.taxonONE(OD.[Unit Price], OD.[Tax Rate]) AS TaxOnEachQuantity

FROM [Fact].[Order] AS OD

FULL OUTER JOIN [Fact].[Stock Holding] AS SH ON OD.[Stock Item Key] = SH.[Stock Item Key]

FULL OUTER JOIN [Dimension].[Stock Item] AS SI ON OD.[Stock Item Key] = SI.[Stock Item Key]

WHERE OD.[Customer Key] != '0'

GROUP BY SI.[Stock Item Key]

,SI.[Stock Item]

,od.[Tax Rate]

,OD.[Unit Price]

ORDER BY SI.[Stock Item Key]

Sample Output (255 Rows Returned)

