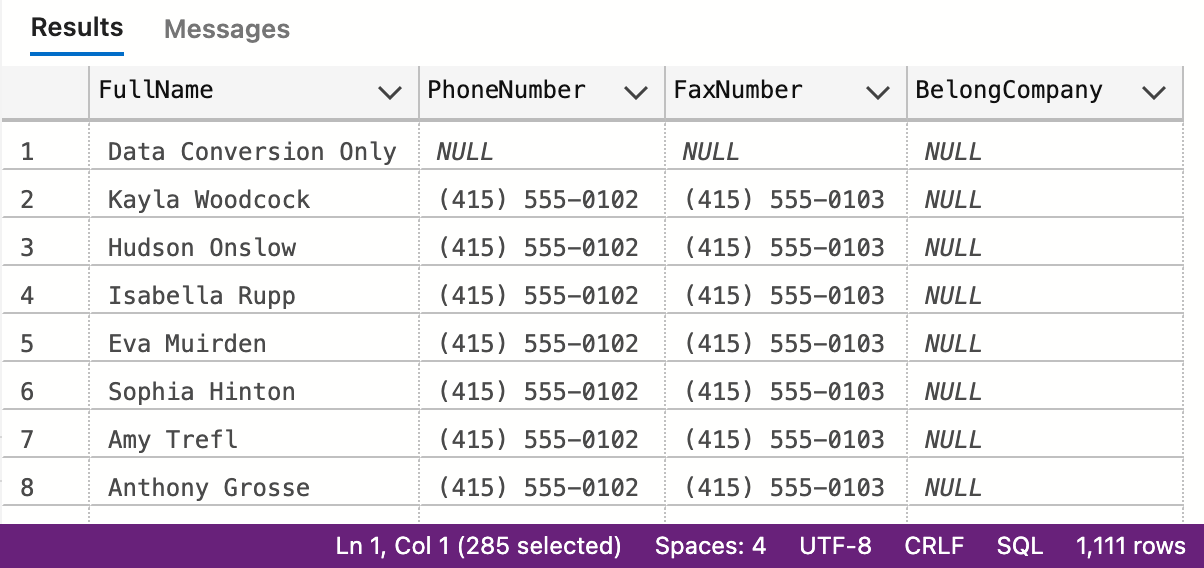
1. **List of Persons’ full name, all their fax and phone numbers, as well as the phone number and fax of the company they are working for (if any).**

SELECT p.[FullName], p.[PhoneNumber], p.[FaxNumber], c.[CustomerName] AS BelongCompany

FROM [WideWorldImporters].[Application].[People] AS p

LEFT JOIN [WideWorldImporters].[Sales].[Customers] As c

ON p.PersonID = c.PrimaryContactPersonID OR p.PersonID = c.AlternateContactPersonID



1. **If the customer's primary contact person has the same phone number as the customer’s phone number, list the customer companies.**

SELECT [CustomerName]

FROM(

SELECT C.[CustomerName], C.[PrimaryContactPersonID], C.[PhoneNumber] AS C\_Number, P.[PhoneNumber] AS P\_Number

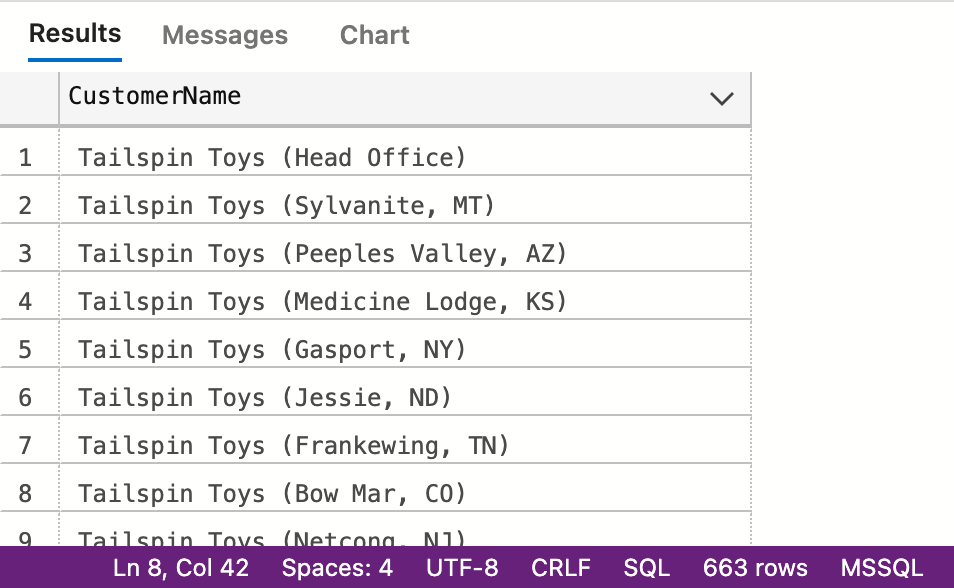
FROM [WideWorldImporters].[Sales].[Customers] AS C

INNER JOIN [WideWorldImporters].[Application].[People] AS P

ON c.[PrimaryContactPersonID] = P.[PersonID]

) AS sub\_q

WHERE sub\_q.[C\_Number] = sub\_q.[P\_Number]



1. **List of customers to whom we made a sale prior to 2016 but no sale since 2016-01-01.**

SELECT \*

FROM(

SELECT DISTINCT[CustomerID]

FROM [WideWorldImporters].[Sales].[CustomerTransactions]

WHERE [TransactionDate] < '2016-01-01'

)AS before\_2016

LEFT JOIN(

SELECT DISTINCT[CustomerID]

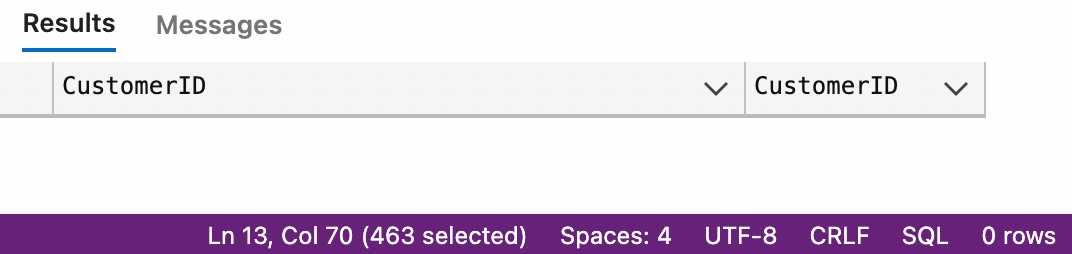
FROM [WideWorldImporters].[Sales].[CustomerTransactions]

WHERE [TransactionDate] >= '2016-01-01'

)AS after\_2016

ON before\_2016.CustomerID = after\_2016.CustomerID

WHERE before\_2016.CustomerID != NULL AND after\_2016.CustomerID = NULL

****

1. **List of Stock Items and total quantity for each stock item in Purchase Orders in Year 2013. 219 rows, Inner Join**

SELECT t3.StockItemName, sum(t3.QuantityPerOuter \* t2.ReceivedOuters) AS total\_quantity

FROM [Purchasing].[PurchaseOrders] AS t1

JOIN [Purchasing].[PurchaseOrderLines] AS t2

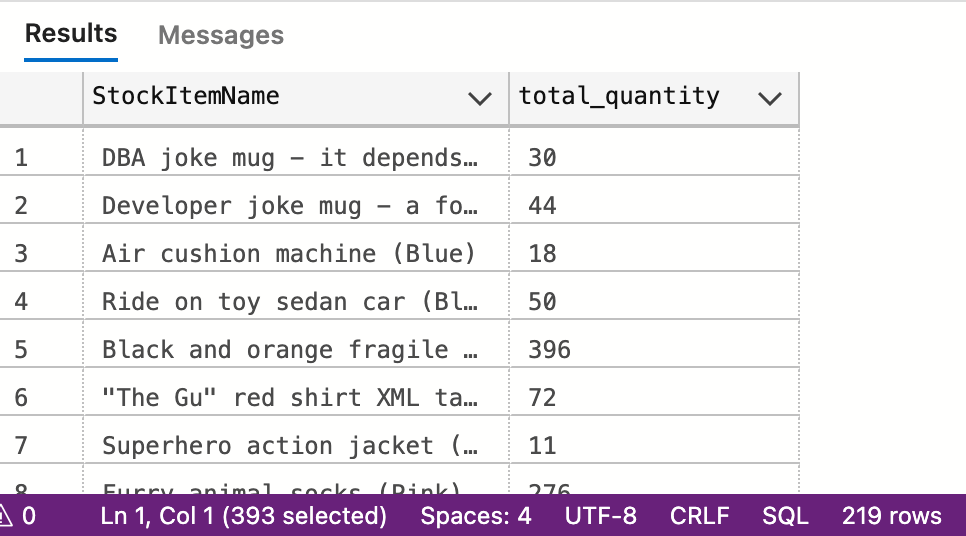
ON t1.PurchaseOrderID = t2.PurchaseOrderID

JOIN [Warehouse].[StockItems] AS t3

ON t2.StockItemID = t3.StockItemID

WHERE t1.OrderDate between '2013-01-01' and '2013-12-31'

GROUP BY t3.StockItemID, t3.StockItemName

****

1. **List of stock items that have at least 10 characters in description.**

SELECT i.[StockItemName], items.[Description]

FROM(

SELECT DISTINCT[StockItemID],[Description]

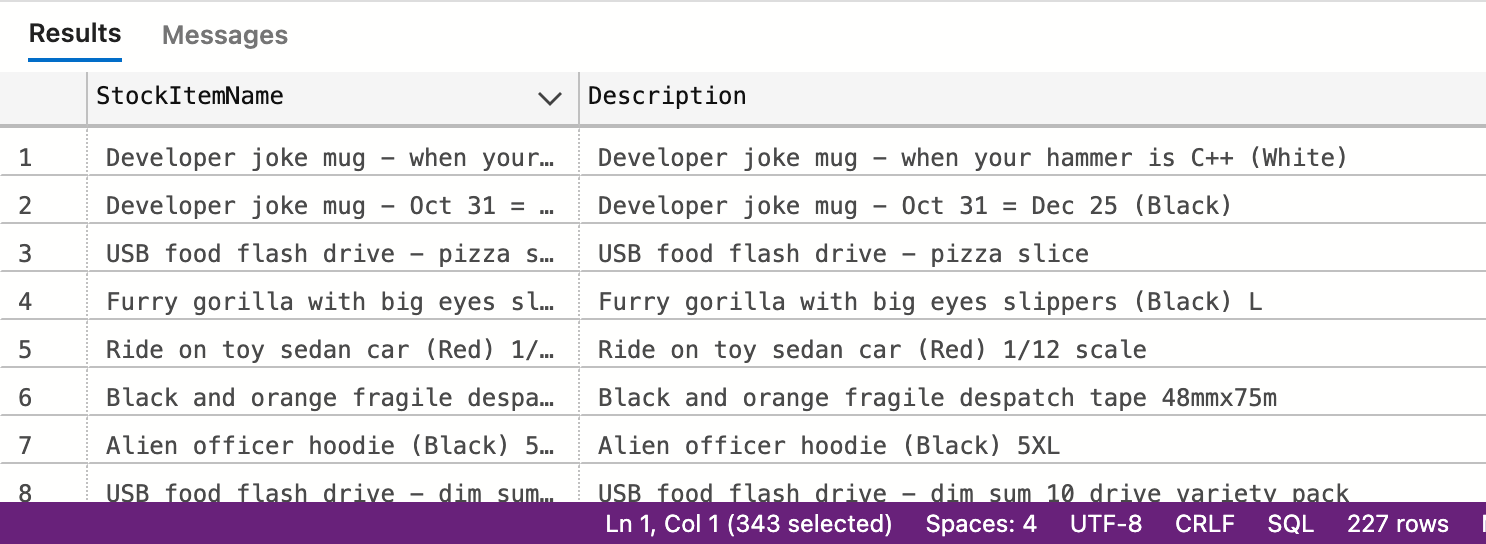
FROM [WideWorldImporters].[Sales].[OrderLines]

WHERE len([WideWorldImporters].[Sales].[OrderLines].[Description]) >= 10

)AS items

LEFT JOIN [WideWorldImporters].[Warehouse].[StockItems] AS i

ON items.StockItemID = i.StockItemID



1. **List of stock items that are not sold to the state of Alabama and Georgia in 2014.**

SELECT DISTINCT a2.StockItemName

FROM(

SELECT\*

FROM(

SELECT s2.[DeliveryCityID], s1.OrderID

FROM(

SELECT [CustomerID], [OrderID]

FROM [WideWorldImporters].[Sales].[Orders]

WHERE [OrderDate] BETWEEN '2014-01-01' AND '2015-01-01'

)AS s1

LEFT JOIN [WideWorldImporters].[Sales].[Customers] AS s2

ON s1.CustomerID = s2.CustomerID

) AS t1

LEFT JOIN (

SELECT c.[CityID], s.[StateProvinceName]

FROM [WideWorldImporters].[Application].[Cities] AS c

LEFT JOIN [WideWorldImporters].[Application].[StateProvinces] As s

ON c.StateProvinceID = s.StateProvinceID

) AS t2

ON t1.DeliveryCityID = t2.CityID

WHERE t2.StateProvinceName != 'Alabama' AND t2.StateProvinceName != 'Georgia'

)AS a1

LEFT JOIN(

SELECT t1.[OrderID],t2.StockItemID, t2.[StockItemName]

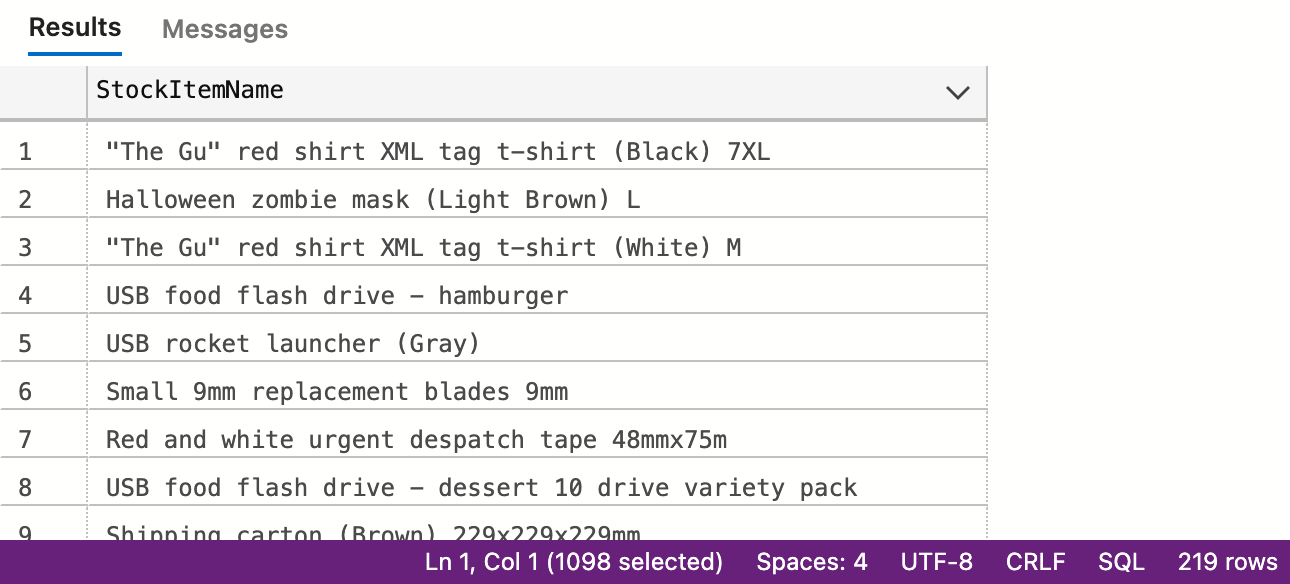
FROM [WideWorldImporters].[Sales].[OrderLines] AS t1

LEFT JOIN [WideWorldImporters].[Warehouse].[StockItems] AS t2

ON t1.StockItemID = t2.StockItemID

)AS a2

ON a1.OrderID = a2.OrderID



1. **List of States and Avg dates for processing (confirmed delivery date – order date).**

SELECT Res\_Table.StateProvinceName

,AVG(Res\_Table.Process\_Days) AS AverageProcessDays

FROM

(

SELECT [State\_Table].StateProvinceName, [city\_to\_province].Process\_Days

FROM(

SELECT Deliv\_City\_Table.DeliveryCityID, Deliv\_City\_Table.Process\_Days, city.StateProvinceID

FROM(

SELECT Process\_Table.CustomerID, Process\_Table.Process\_Days, Cus\_Table.DeliveryCityID

FROM(

SELECT [deliv\_table].[OrderID]

,[deliv\_table].[CustomerID]

,DATEDIFF(day, [order\_date].[OrderDate], CAST([deliv\_table].[ConfirmedDeliveryTime] AS date)) AS Process\_Days

FROM [WideWorldImporters].[Sales].[Orders] AS order\_date

INNER JOIN [WideWorldImporters].[Sales].[Invoices] AS deliv\_table

ON order\_date.OrderID = deliv\_table.OrderID

)AS Process\_Table

FULL JOIN [WideWorldImporters].[Sales].[Customers] AS Cus\_Table

ON Process\_Table.CustomerID = Cus\_Table.CustomerID

)AS Deliv\_City\_Table

FULL JOIN [WideWorldImporters].[Application].[Cities] AS city

ON Deliv\_City\_Table.DeliveryCityID = city.CityID

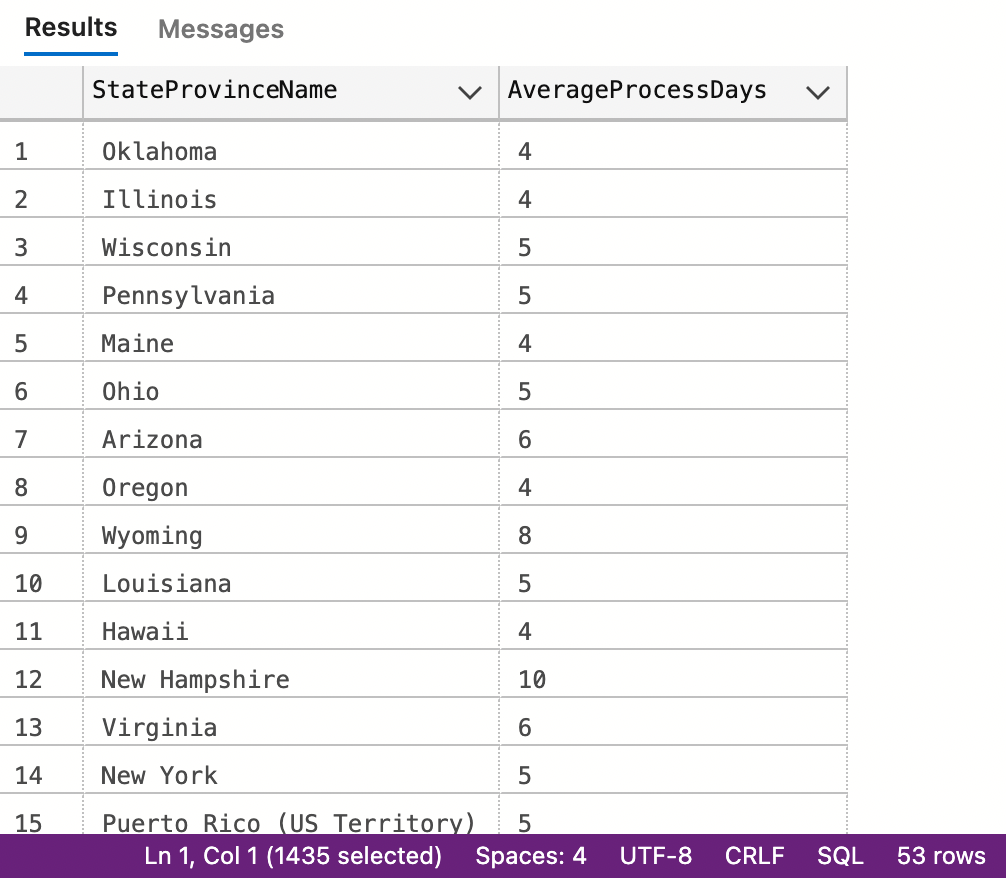
)AS city\_to\_province

FULL JOIN [WideWorldImporters].[Application].[StateProvinces] AS State\_Table

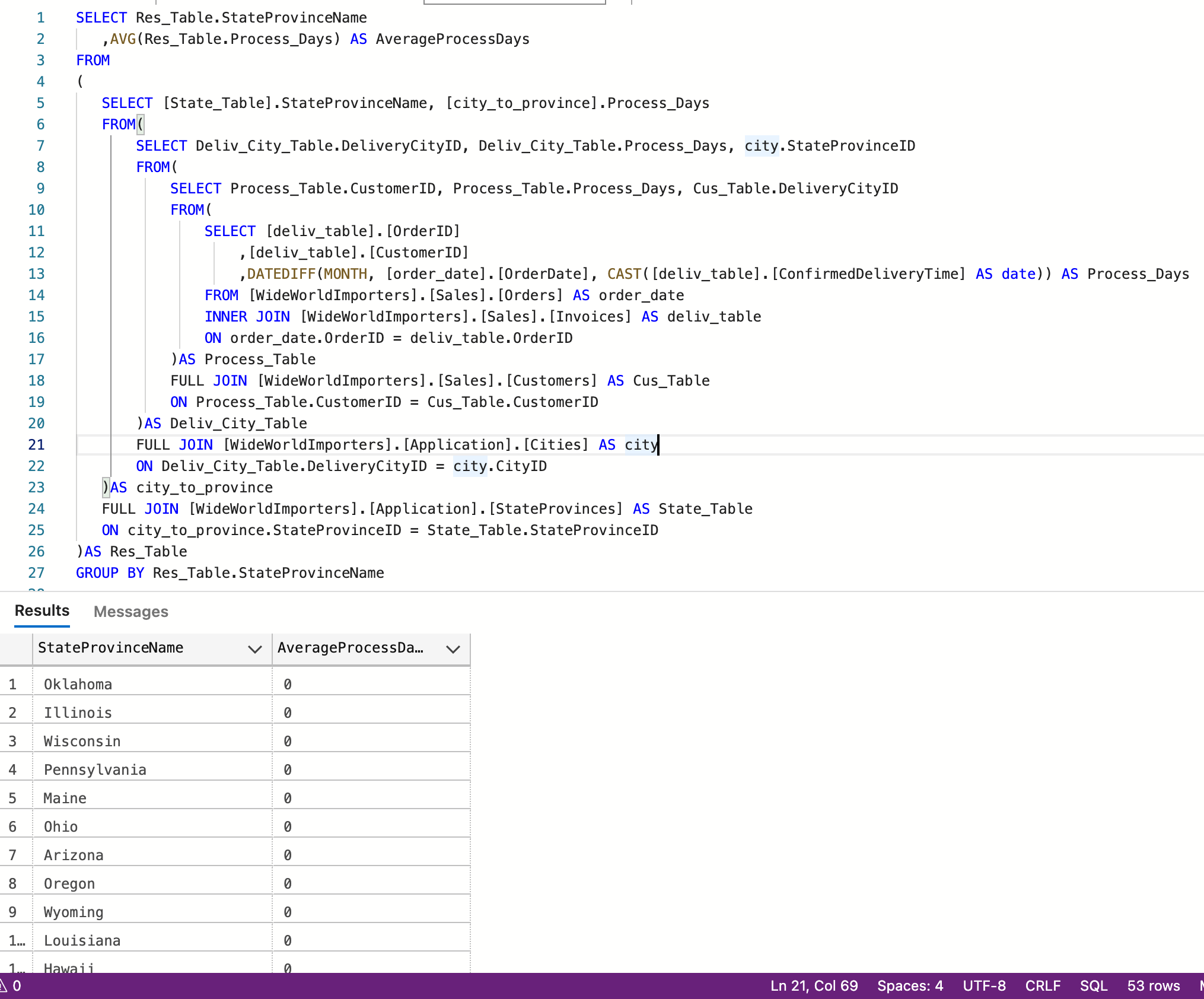
ON city\_to\_province.StateProvinceID = State\_Table.StateProvinceID

)AS Res\_Table

GROUP BY Res\_Table.StateProvinceName



1. **List of States and Avg dates for processing (confirmed delivery date – order date) by month.**

****

1. **List of StockItems that the company purchased more than sold in the year of 2015.**

SELECT purchase.StockItemName, purchase.TotalPurchaseQuantity, sold.TotalSoldQuantity

FROM(

SELECT OuterTable.StockItemID, details.StockItemName

, OuterTable.TotalOrderedOuters \* details.QuantityPerOuter AS TotalPurchaseQuantity

FROM(

SELECT [StockItemID]

,SUM([OrderedOuters]) AS TotalOrderedOuters

FROM [WideWorldImporters].[Purchasing].[PurchaseOrderLines]

WHERE [LastReceiptDate] >= '2015-01-01'

GROUP BY [StockItemID]

)AS OuterTable

LEFT JOIN [WideWorldImporters].[Warehouse].[StockItems] AS details

ON OuterTable.StockItemID = details.StockItemID

)AS purchase

LEFT JOIN(

SELECT [StockItemID]

,SUM([Quantity]) AS TotalSoldQuantity

FROM [WideWorldImporters].[Sales].[OrderLines]

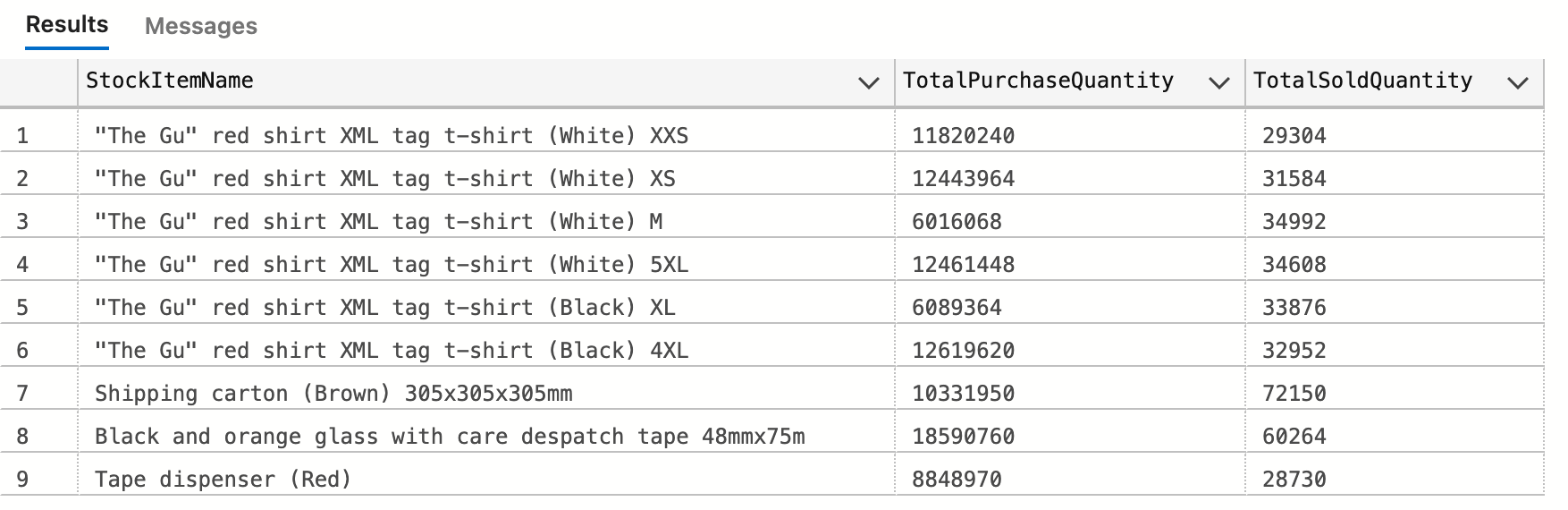
WHERE [LastEditedWhen] > '2015-01-01'

GROUP BY [StockItemID]

)AS sold

ON purchase.StockItemID = sold.StockItemID

WHERE purchase.TotalPurchaseQuantity > sold.TotalSoldQuantity



1. **List of Customers and their phone number, together with the primary contact person’s name, to whom we did not sell more than 10 mugs (search by name) in the year 2016.**

SELECT customerInfo.CustomerName, customer.MugSoldQuantity,

customerInfo.PhoneNumber, customerInfo.FullName AS PrimaryContactPersonName

FROM(

SELECT table1.CustomerID, SUM(table1.Quantity) AS MugSoldQuantity

FROM(

SELECT orderLines.[StockItemID]

,orderLines.[Quantity]

,orders.[CustomerID]

FROM [WideWorldImporters].[Sales].[OrderLines] AS orderLines

LEFT JOIN [WideWorldImporters].[Sales].[Orders] AS orders

ON orderLines.OrderID = orders.OrderID

WHERE orderLines.[LastEditedWhen] BETWEEN '2016-01-01' AND '2017-01-01'

)AS table1

RIGHT JOIN(

SELECT [StockItemID]

,[StockItemName]

,[OuterPackageID]

,[QuantityPerOuter]

FROM [WideWorldImporters].[Warehouse].[StockItems]

WHERE [SearchDetails] LIKE '%mug%'

)AS table2

ON table1.StockItemID = table2.StockItemID

GROUP BY table1.CustomerID

)AS customer

LEFT JOIN(

SELECT source1.[CustomerID], source1.[CustomerName], source1.[PhoneNumber]

, source1.[PrimaryContactPersonID], source2.FullName

FROM [WideWorldImporters].[Sales].[Customers] AS source1

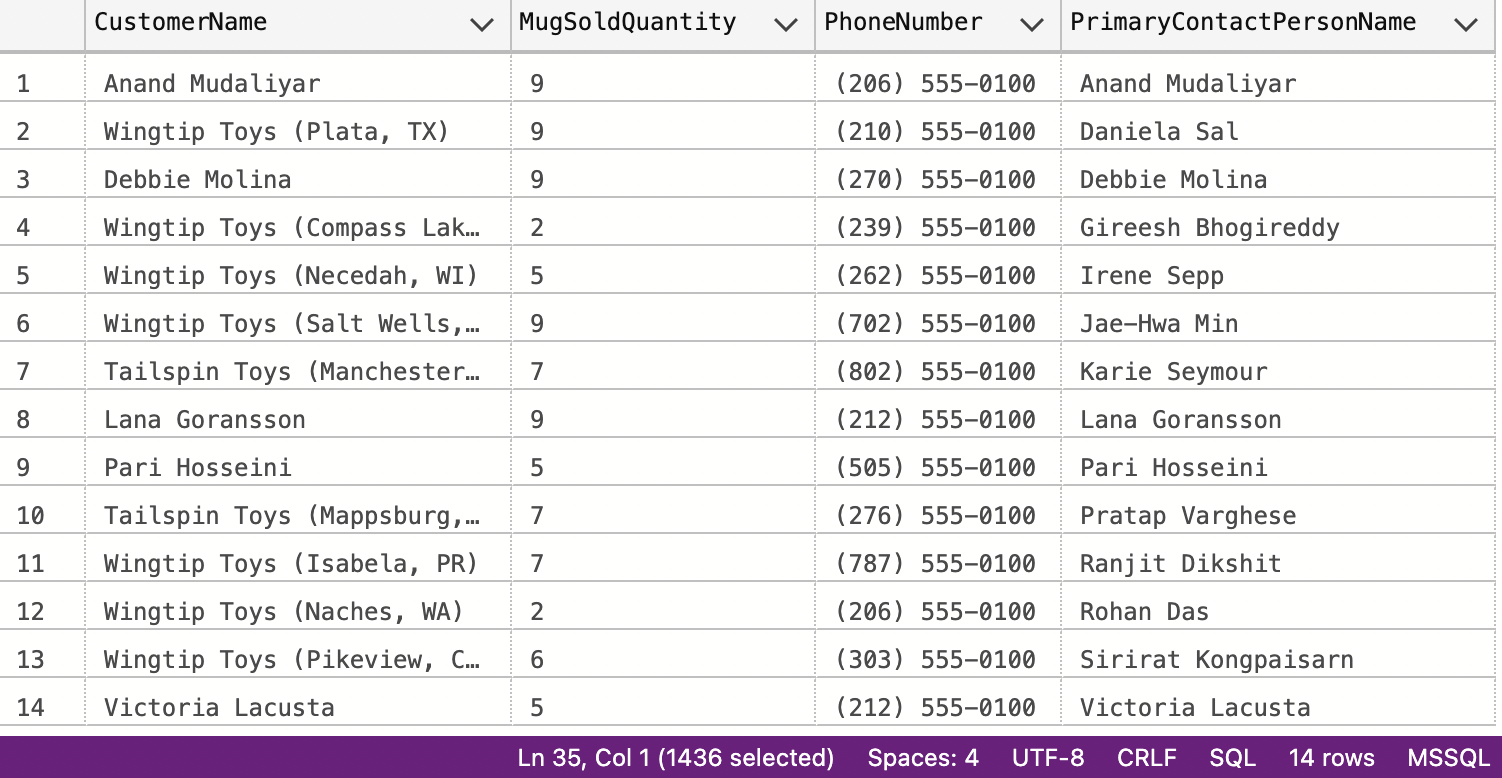
LEFT JOIN [WideWorldImporters].[Application].[People] AS source2

ON source1.PrimaryContactPersonID = source2.PersonID

)AS customerInfo

ON customer.CustomerID = customerInfo.CustomerID

WHERE customer.MugSoldQuantity < 10



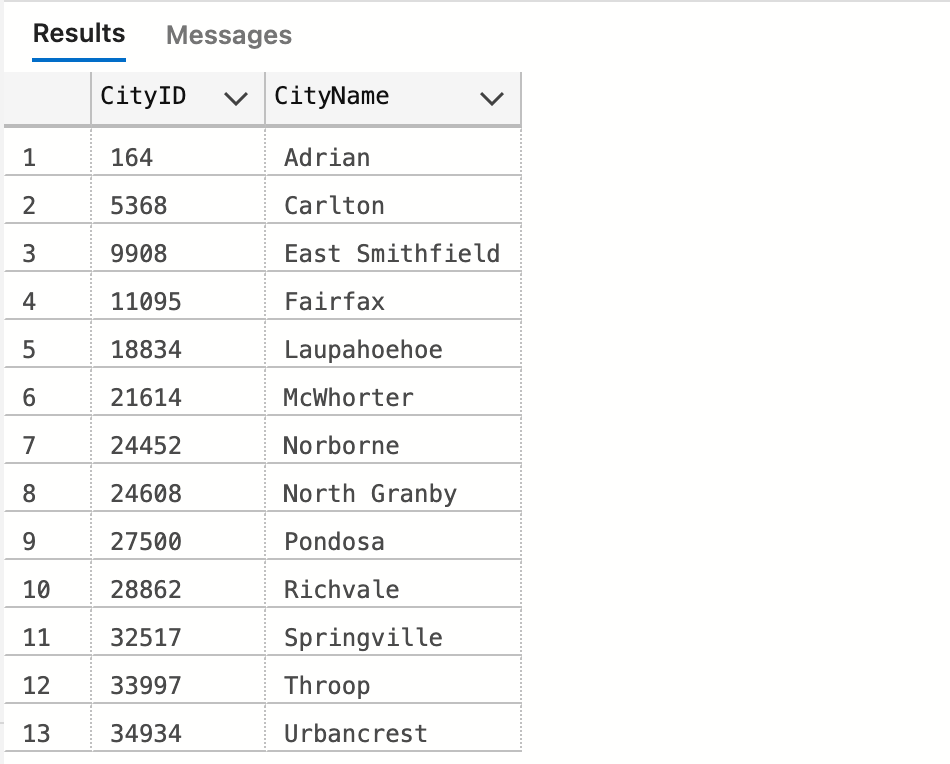
1. **List all the cities that were updated after 2015-01-01.**

SELECT [CityID]

,[CityName]

FROM [WideWorldImporters].[Application].[Cities]

WHERE [ValidFrom] > '2015-01-01'



1. **List all the Order Detail (Stock Item name, delivery address, delivery state, city, country, customer name, customer contact person name, customer phone, quantity) for the date of 2014-07-01. Info should be relevant to that date.**

SELECT people1.StockItemName, people1.DeliveryAddressLine1, people1.DeliveryAddressLine2

, people1.CityName AS City, people1.StateProvinceName AS States, people1.CountryName AS Country,

people1.CustomerName, people2.FullName AS ContactPersonName, people1.CustomerPhone, people1.Quantity

FROM(

SELECT\*

FROM(

SELECT table1.StockItemName, table2.[DeliveryAddressLine1], table2.[DeliveryAddressLine2]

,table2.[DeliveryCityID], table2.[CustomerName], table2.[PrimaryContactPersonID]

,table2.[PhoneNumber] AS CustomerPhone,table1.[CustomerID], table1.[Quantity]

FROM(

SELECT source2.StockItemName, source1.CustomerID, source1.Quantity

FROM(

SELECT StockItem.[StockItemID], StockItem.[Quantity]

,Customers.CustomerID, Customers.OrderDate

FROM [WideWorldImporters].[Sales].[OrderLines] AS StockItem

FULL JOIN(

SELECT CustomerID, OrderDate, OrderID

FROM [WideWorldImporters].[Sales].[Orders]

)AS Customers

ON StockItem.OrderID = Customers.OrderID

WHERE Customers.OrderDate = '2014-07-01'

)AS source1

LEFT JOIN [WideWorldImporters].[Warehouse].[StockItems] source2

ON source1.StockItemID = source2.StockItemID

)AS table1

LEFT JOIN [WideWorldImporters].[Sales].[Customers] AS table2

ON table1.CustomerID = table2.CustomerID

)AS table2

LEFT JOIN(

SELECT source1.CityID, source1.CityName, source1.StateProvinceName, country.CountryName

FROM(

SELECT city.[CityID],city.[CityName],states.[StateProvinceID]

, states.[StateProvinceName], states.[CountryID]

FROM [WideWorldImporters].[Application].[Cities] AS city

FULL JOIN [WideWorldImporters].[Application].[StateProvinces] AS states

ON city.StateProvinceID = states.StateProvinceID

)AS source1

FULL JOIN [WideWorldImporters].[Application].[Countries] AS country

ON source1.CountryID = country.CountryID

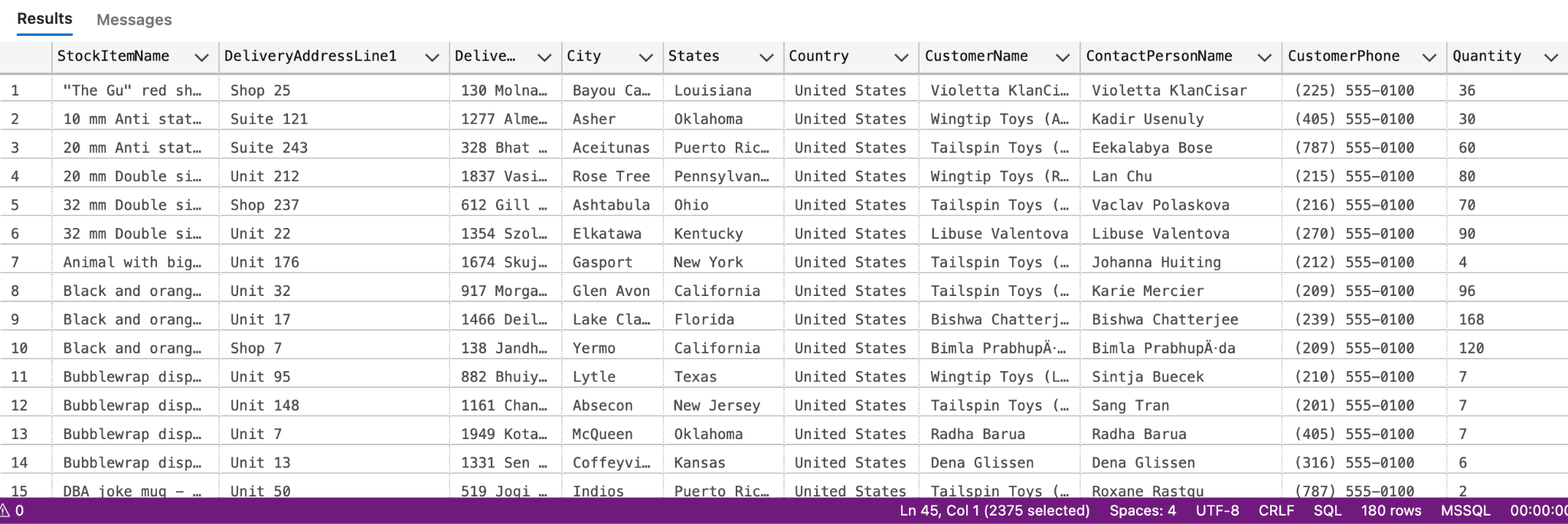
)AS table3

ON table2.DeliveryCityID = table3.CityID

)AS people1

LEFT JOIN [WideWorldImporters].[Application].[People] AS people2

ON people1.PrimaryContactPersonID = people2.PersonID



1. **List of stock item groups and total quantity purchased, total quantity sold, and the remaining stock quantity (quantity purchased – quantity sold) 9 rows**

WITH PT AS(

SELECT si.StockItemID

, SUM(CAST(si.QuantityPerOuter \* pol.ReceivedOuters as bigint)) AS PurchaseQuantity

FROM WideWorldImporters.Warehouse.StockItems AS si

JOIN WideWorldImporters.Purchasing.PurchaseOrderLines AS pol

ON si.StockItemID = pol.StockItemID

GROUP BY si.StockItemID

)

SELECT t2.StockGroupID

, SUM(p.PurchaseQuantity) AS PurchaseTotalQuantity

, SUM(o.Quantity) AS SoldTotalQuantity

, SUM(p.PurchaseQuantity) - SUM(o.Quantity) AS RemainingStockQuantity

FROM WideWorldImporters.Warehouse.StockItems AS t1

LEFT JOIN WideWorldImporters.Warehouse.StockItemStockGroups AS t2

ON t1.StockItemID = t2.StockItemID

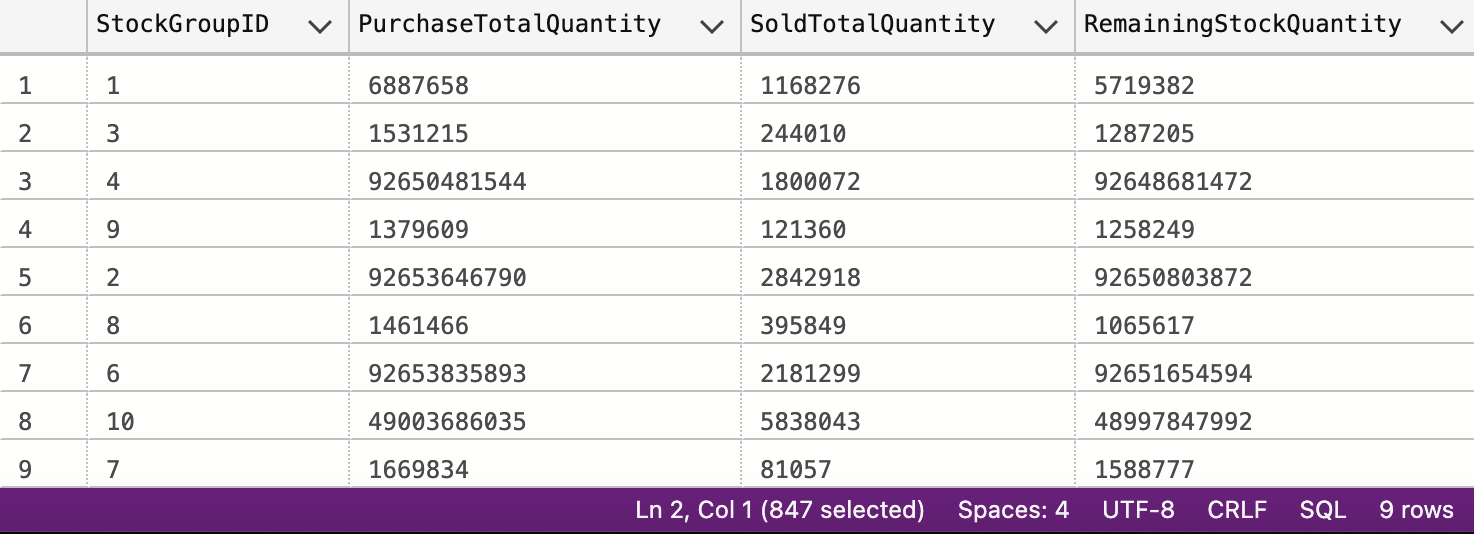
JOIN PT AS p

ON t1.StockItemID = p.StockItemID

JOIN WideWorldImporters.Sales.OrderLines AS o

ON t1.StockItemID = o.StockItemID

GROUP BY t2.StockGroupID

****

1. **List of Cities in the US and the stock item that the city got the most deliveries in 2016. If the city did not purchase any stock items in 2016, print “No Sales”.**

with t1 as(select ol.StockItemID, c.DeliveryCityID, ol.Quantity from Sales.Orders as o

join Sales.Customers as c on o.CustomerID = c.CustomerID

join Sales.OrderLines as ol on ol.OrderID = o.OrderID

where o.OrderDate between '2016-01-01' and '2016-12-31'

), t2 as(

select StockItemID, DeliveryCityID

from (

select StockItemID, DeliveryCityID,

DENSE\_RANK() OVER(PARTITION BY DeliveryCityId order by Quantity DESC)as rk

from t1) as t3

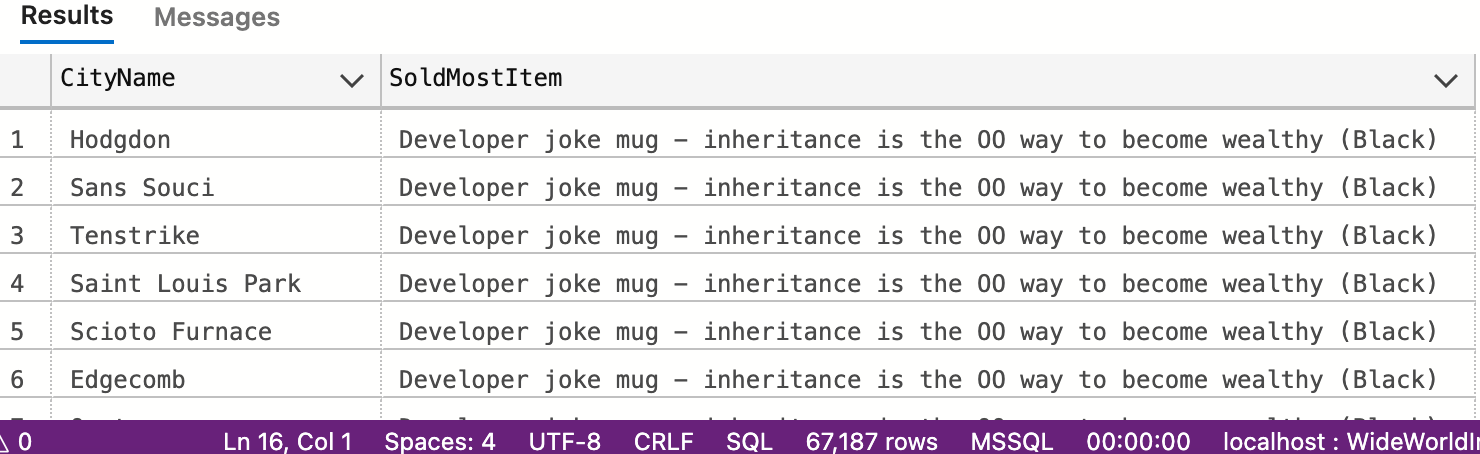
where rk = 1

)

select c.CityName, ISNULL(s.StockItemName, 'No Sale') as SoldMostItem

from t1 c1 join Warehouse.StockItems s on c1.StockItemID = s.StockItemID

right join Application.Cities c ON c1.DeliveryCityID = c.CityID

****

1. **List any orders that had more than one delivery attempt (located in invoice table).**

SELECT res.OrderID

FROM

(

SELECT [OrderID], [ReturnedDeliveryData],

(LEN(ReturnedDeliveryData) - LEN(REPLACE(ReturnedDeliveryData, 'Attempt', ''))) / LEN('Attempt') AS AttemptCount

FROM [WideWorldImporters].[Sales].[Invoices]

)AS res

WHERE res.AttemptCount > 1

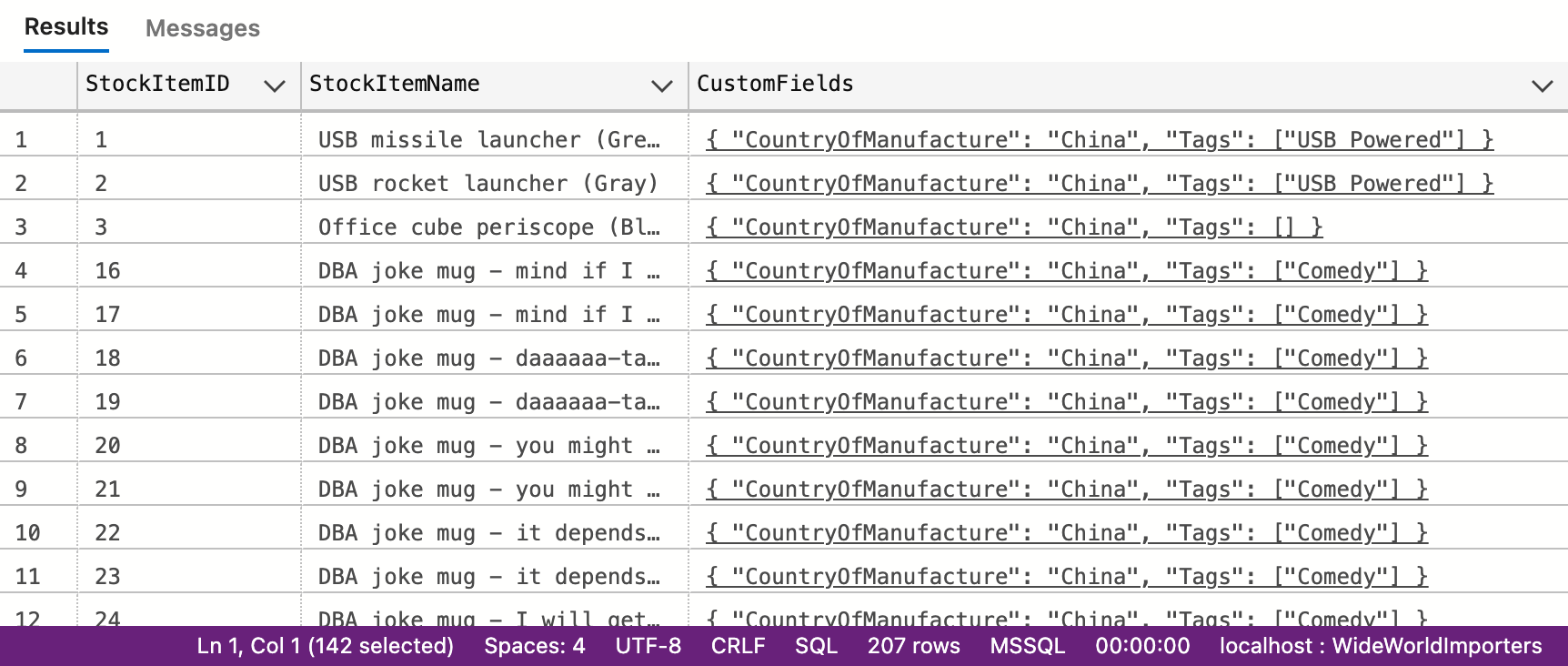


1. **List all stock items that are manufactured in China. (Country of Manufacture)**

SELECT [StockItemID], [StockItemName], [CustomFields]

FROM [WideWorldImporters].[Warehouse].[StockItems]

WHERE [CustomFields] LIKE '%China%'

****

1. **Total quantity of stock items sold in 2015, group by country of manufacturing.**

SELECT tab.Country, SUM(tab.TotalQuantity)

FROM(

SELECT res.TotalQuantity,

SUBSTRING(

res.CustomFields, 28,5

) AS Country

FROM(

SELECT t1.StockItemID, t2.[CustomFields], t1.TotalQuantity

FROM(

SELECT s2.StockItemID, SUM(s2.Quantity) AS TotalQuantity

FROM(

SELECT OrderID

FROM WideWorldImporters.Sales.Orders

WHERE OrderDate BETWEEN '2015-01-01' AND '2016-01-01'

)AS s1

LEFT JOIN WideWorldImporters.Sales.OrderLines AS s2

ON s1.OrderID = s2.OrderID

GROUP BY s2.StockItemID

)AS t1

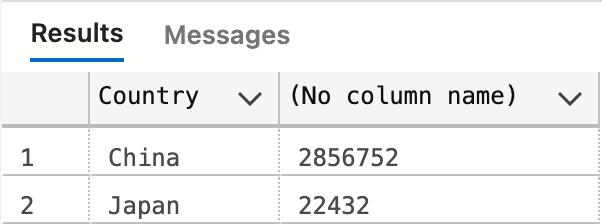
LEFT JOIN [WideWorldImporters].[Warehouse].[StockItems] AS t2

ON t1.StockItemID = t2.StockItemID

)AS res

)AS tab

GROUP BY tab.Country

****

1. **Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Stock Group Name, 2013, 2014, 2015, 2016, 2017]**

CREATE VIEW Q18Answer AS

select n.StockGroupName, t2013.Quantity as Sold2013, t2014.Quantity as Sold2014, t2015.Quantity as Sold2015

, t2016.Quantity as Sold2016, t2017.Quantity as Sold2017

from(

select t2.StockGroupID, SUM(t1.Quantity) as Quantity

from(

select s2.StockItemID, SUM(s2.Quantity) as Quantity

from Sales.Orders s1 left join Sales.OrderLines s2 ON s1.OrderID = s2.OrderID

where s1.OrderDate between '2013-01-01' and '2013-12-31'

group by s2.StockItemID

)as t1

left join Warehouse.StockItemStockGroups as t2

on t1.StockItemID = t2.StockItemID

group by t2.StockGroupID

)as t2013 join(

select t2.StockGroupID, SUM(t1.Quantity) as Quantity

from(

select s2.StockItemID, SUM(s2.Quantity) as Quantity

from Sales.Orders s1 left join Sales.OrderLines s2 ON s1.OrderID = s2.OrderID

where s1.OrderDate between '2014-01-01' and '2014-12-31'

group by s2.StockItemID

)as t1

left join Warehouse.StockItemStockGroups as t2

on t1.StockItemID = t2.StockItemID

group by t2.StockGroupID

)as t2014 on t2013.StockGroupID = t2014.StockGroupID

join(

select t2.StockGroupID, SUM(t1.Quantity) as Quantity

from(

select s2.StockItemID, SUM(s2.Quantity) as Quantity

from Sales.Orders s1 left join Sales.OrderLines s2 ON s1.OrderID = s2.OrderID

where s1.OrderDate between '2015-01-01' and '2015-12-31'

group by s2.StockItemID

)as t1

left join Warehouse.StockItemStockGroups as t2

on t1.StockItemID = t2.StockItemID

group by t2.StockGroupID

)as t2015 on t2013.StockGroupID = t2015.StockGroupID

join(

select t2.StockGroupID, SUM(t1.Quantity) as Quantity

from(

select s2.StockItemID, SUM(s2.Quantity) as Quantity

from Sales.Orders s1 left join Sales.OrderLines s2 ON s1.OrderID = s2.OrderID

where s1.OrderDate between '2016-01-01' and '2016-12-31'

group by s2.StockItemID

)as t1

left join Warehouse.StockItemStockGroups as t2

on t1.StockItemID = t2.StockItemID

group by t2.StockGroupID

)as t2016 on t2013.StockGroupID = t2016.StockGroupID

left join(

select t2.StockGroupID, SUM(t1.Quantity) as Quantity

from(

select s2.StockItemID, SUM(s2.Quantity) as Quantity

from Sales.Orders s1 left join Sales.OrderLines s2 ON s1.OrderID = s2.OrderID

where s1.OrderDate between '2017-01-01' and '2017-12-31'

group by s2.StockItemID

)as t1

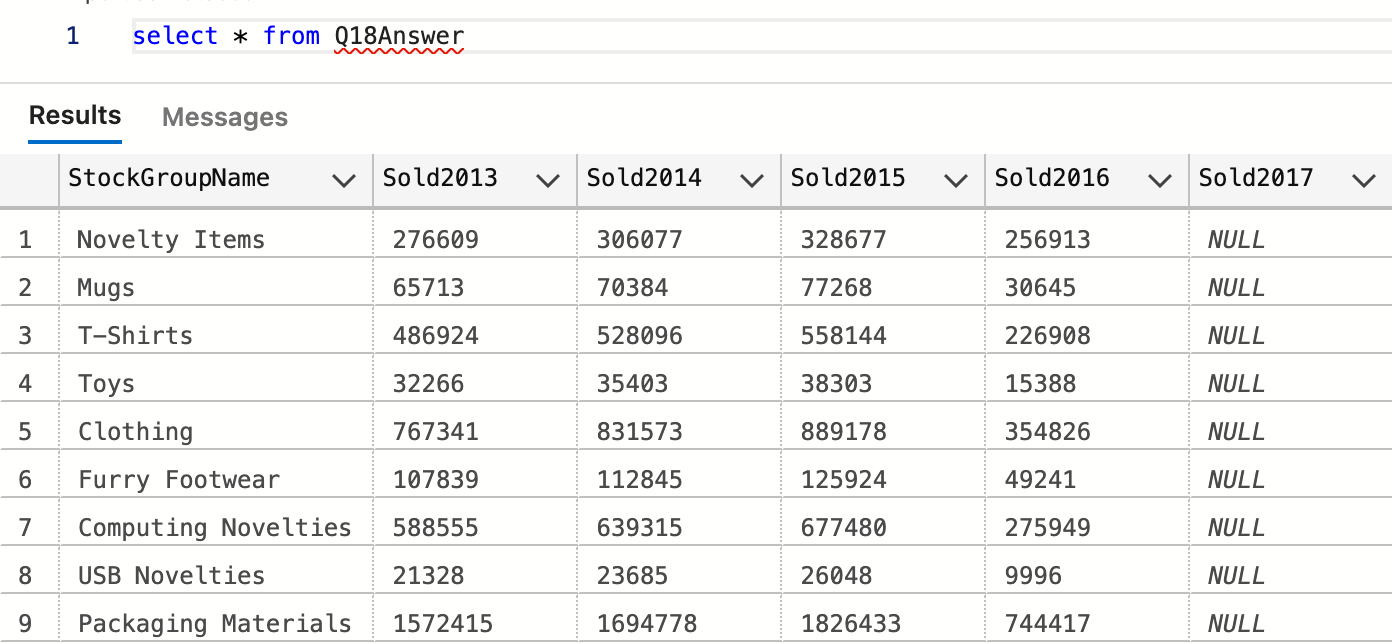
left join Warehouse.StockItemStockGroups as t2

on t1.StockItemID = t2.StockItemID

group by t2.StockGroupID

)as t2017 on t2013.StockGroupID = t2017.StockGroupID

left join Warehouse.StockGroups n on t2013.StockGroupID = n.StockGroupID

****

1. **Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Year, Stock Group Name1, Stock Group Name2, Stock Group Name3, … , Stock Group Name10**

CREATE VIEW Q19Ans AS

WITH SoldYear AS(

SELECT StockGroupID, Sold\_year,SUM(Quantity) AS QuantityYear

FROM (

SELECT s3.StockGroupID,year(o.OrderDate) AS Sold\_year, ol.Quantity

FROM Sales.OrderLines ol JOIN Sales.Orders o ON ol.OrderID = o.OrderID

JOIN Warehouse.StockItems s1 ON ol.StockItemID = s1.StockItemID

JOIN Warehouse.StockItemStockGroups s2 ON s2.StockItemID = s1.StockItemID

JOIN Warehouse.StockGroups s3 ON s2.StockGroupID = s3.StockGroupID

) AS t1

GROUP BY Sold\_year,StockGroupID

), PivotTable AS (

SELECT\* FROM SoldYear

PIVOT(

SUM(QuantityYear) FOR StockGroupID IN([1],[2],[3],[4],[5],[6],[7],[8],[9],[10])

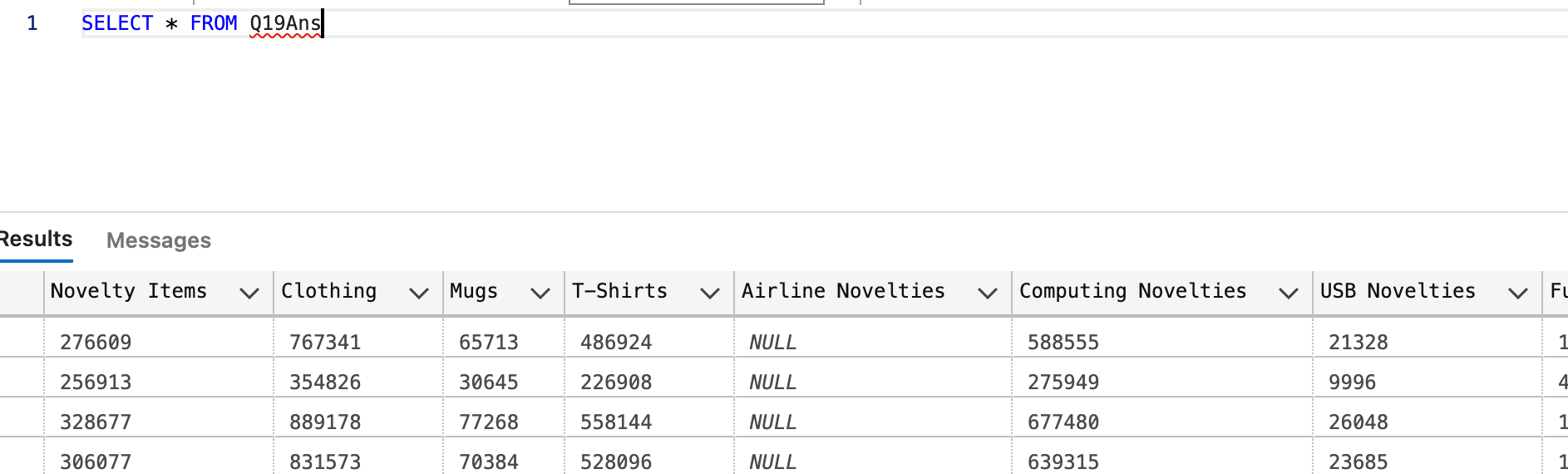
)AS t2

)

SELECT "1" as 'Novelty Items', "2" as 'Clothing', "3" as 'Mugs', "4" as 'T-Shirts', "5" as 'Airline Novelties'

, "6" as 'Computing Novelties', "7" as 'USB Novelties', "8" as 'Furry Footwear', "9" as 'Toys', "10" as 'Packaging Materials'

FROM PivotTable



1. **Create a function, input: order id; return: total of that order. List invoices and use that function to attach the order total to the other fields of invoices.**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [Website].[OrderTotal] (@OrderID int)

RETURNS decimal(18,2) AS

BEGIN

DECLARE @result decimal(18,2);

SET @result = (

SELECT SUM(t1.OrderSum) AS res FROM(

SELECT ol.OrderID, ol.Quantity \* ol.UnitPrice AS OrderSum

FROM Sales.OrderLines AS ol

)AS t1

WHERE t1.OrderID = @OrderID

GROUP BY t1.OrderID

)

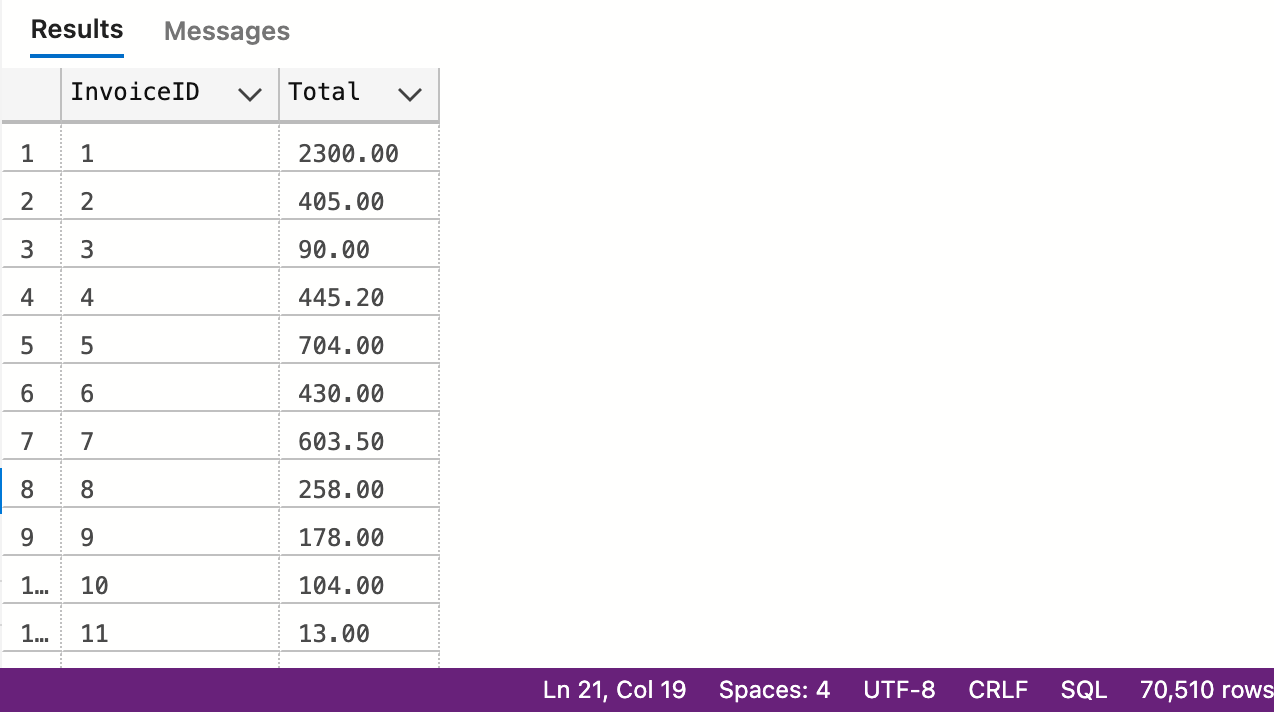
RETURN @result

END

GO

SELECT InvoiceID, Website.OrderTotal(orderID) AS Total

FROM Sales.Invoices

****

1. **Create a new table called ods.Orders. Create a stored procedure, with proper error handling and transactions, that input is a date; when executed, it would find orders of that day, calculate order total, and save the information (order id, order date, order total, customer id) into the new table. If a given date is already existing in the new table, throw an error and roll back. Execute the stored procedure 5 times using different dates.**
2. **Create a new table called ods.StockItem. It has following columns: [StockItemID], [StockItemName] ,[SupplierID] ,[ColorID] ,[UnitPackageID] ,[OuterPackageID] ,[Brand] ,[Size] ,[LeadTimeDays] ,[QuantityPerOuter] ,[IsChillerStock] ,[Barcode] ,[TaxRate] ,[UnitPrice],[RecommendedRetailPrice] ,[TypicalWeightPerUnit] ,[MarketingComments] ,[InternalComments], [CountryOfManufacture], [Range], [Shelflife]. Migrate all the data in the original stock item table**

CREATE TABLE ods\_StockItem(StockItemID INT PRIMARY KEY,

StockItemName NVARCHAR(100) NOT NULL,

SupplierID INT NOT NULL,

ColorID INT NULL,

UnitPackageID INT NOT NULL,

OuterPackageID INT NOT NULL,

Brand NVARCHAR(50) NULL,

Size NVARCHAR(20) NULL,

LeadTimeDays INT NOT NULL,

QuantityPerOuter INT NOT NULL,

IsChillerStock BIT NULL,

Barcode NVARCHAR(50) NULL,

TaxRate DECIMAL(18, 3) NOT NULL,

UnitPrice DECIMAL(18, 2) NOT NULL,

RecommendedRetailPrice DECIMAL(18, 2) NULL,

TypicalWeightPerUnit DECIMAL(18, 3) NOT NULL,

MarketingComments NVARCHAR(MAX) NULL,

InternalComments NVARCHAR(MAX) NULL,

CountryOfManufacture NVARCHAR(20) NULL,

[Range] NVARCHAR(20) NULL,

Shelflife NVARCHAR(20) NULL)

MERGE INTO ods\_StockItem AS os

USING Warehouse.StockItems AS s

ON os.StockItemID = s.StockItemID

WHEN NOT MATCHED

THEN INSERT VALUES (s.StockItemID,

s.StockItemName,

s.SupplierID,

s.ColorID,

s.UnitPackageID,

s.OuterPackageID,

s.Brand,

s.Size,

s.LeadTimeDays,

s.QuantityPerOuter,

s.IsChillerStock,

s.Barcode,

s.TaxRate,

s.UnitPrice,

s.RecommendedRetailPrice,

s.TypicalWeightPerUnit,

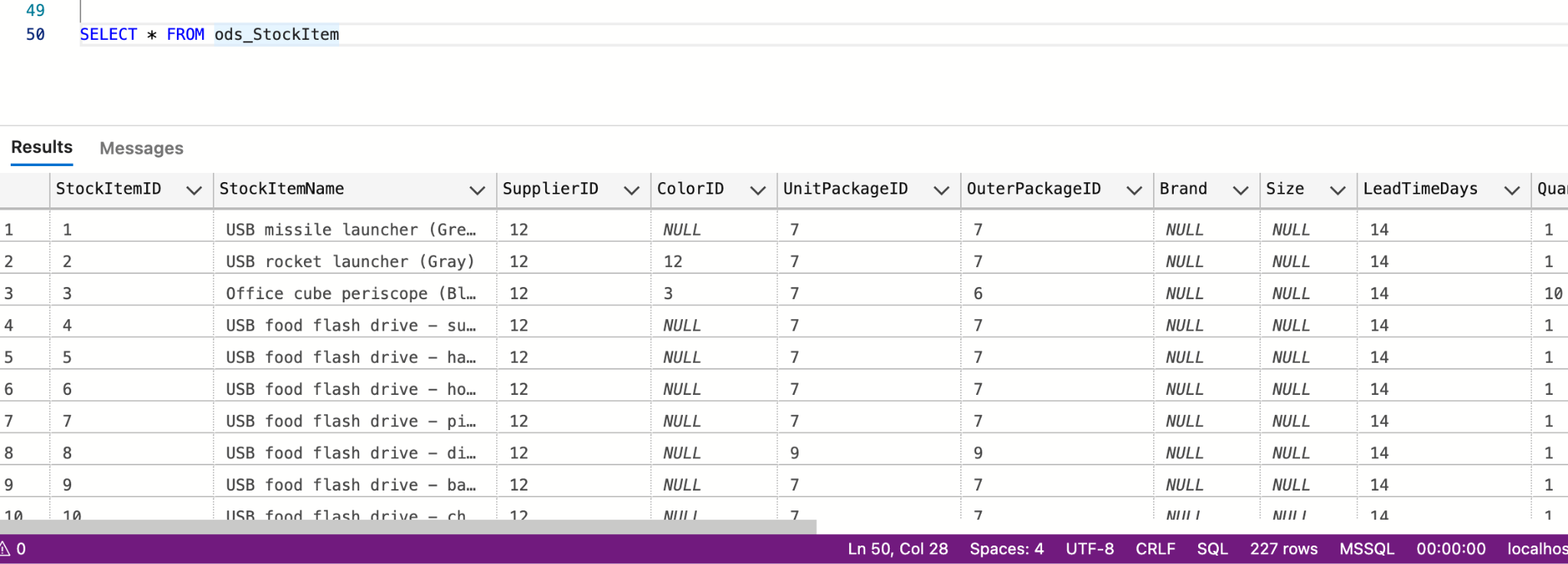
s.MarketingComments,

s.InternalComments,

JSON\_VALUE(s.CustomFields,'$.CountryOfManufacture'),

JSON\_VALUE(s.CustomFields,'$.Range'),

JSON\_VALUE(s.CustomFields,'$.ShelfLife'));



1. **Rewrite your stored procedure in (21). Now with a given date, it should wipe out all the order data prior to the input date and load the order data that was placed in the next 7 days following the input date.**
2. **Consider the JSON file:**

**{**

**"PurchaseOrders":[**

**{**

**"StockItemName":"Panzer Video Game",**

**"Supplier":"7",**

**"UnitPackageId":"1",**

**"OuterPackageId":[**

**6,**

**7**

**],**

**"Brand":"EA Sports",**

**"LeadTimeDays":"5",**

**"QuantityPerOuter":"1",**

**"TaxRate":"6",**

**"UnitPrice":"59.99",**

**"RecommendedRetailPrice":"69.99",**

**"TypicalWeightPerUnit":"0.5",**

**"CountryOfManufacture":"Canada",**

**"Range":"Adult",**

**"OrderDate":"2018-01-01",**

**"DeliveryMethod":"Post",**

**"ExpectedDeliveryDate":"2018-02-02",**

**"SupplierReference":"WWI2308"**

**},**

**{**

**"StockItemName":"Panzer Video Game",**

**"Supplier":"5",**

**"UnitPackageId":"1",**

**"OuterPackageId":"7",**

**"Brand":"EA Sports",**

**"LeadTimeDays":"5",**

**"QuantityPerOuter":"1",**

**"TaxRate":"6",**

**"UnitPrice":"59.99",**

**"RecommendedRetailPrice":"69.99",**

**"TypicalWeightPerUnit":"0.5",**

**"CountryOfManufacture":"Canada",**

**"Range":"Adult",**

**"OrderDate":"2018-01-025",**

**"DeliveryMethod":"Post",**

**"ExpectedDeliveryDate":"2018-02-02",**

**"SupplierReference":"269622390"**

**}**

**]**

**}**

**Looks like that it is our missed purchase orders. Migrate these data into Stock Item, Purchase Order and Purchase Order Lines tables. Of course, save the script.**

This one fails but I made some progress….

DECLARE @json NVARCHAR(MAX)

SET @json = N'[

{

"StockItemName":"Panzer Video Game",

"Supplier":"7",

"UnitPackageId":"1",

"OuterPackageId":[6,7],

"Brand":"EA Sports",

"LeadTimeDays":"5",

"QuantityPerOuter":"1",

"TaxRate":"6",

"UnitPrice":"59.99",

"RecommendedRetailPrice":"69.99",

"TypicalWeightPerUnit":"0.5",

"CountryOfManufacture":"Canada",

"Range":"Adult",

"OrderDate":"2018-01-01",

"DeliveryMethod":"Post",

"ExpectedDeliveryDate":"2018-02-02",

"SupplierReference":"WWI2308"

},

{

"StockItemName":"Panzer Video Game",

"Supplier":"5",

"UnitPackageId":"1",

"OuterPackageId":"7",

"Brand":"EA Sports",

"LeadTimeDays":"5",

"QuantityPerOuter":"1",

"TaxRate":"6",

"UnitPrice":"59.99",

"RecommendedRetailPrice":"69.99",

"TypicalWeightPerUnit":"0.5",

"CountryOfManufacture":"Canada",

"Range":"Adult",

"OrderDate":"2018-01-025",

"DeliveryMethod":"Post",

"ExpectedDeliveryDate":"2018-02-02",

"SupplierReference":"269622390"

}

]'

insert into Warehouse.StockItems

select \* from OPENJSON(@json) with (

StockItemName nvarchar(100) '$.StockItemName',

SupplierID int '$.Supplier', UnitPackageId int '$.UnitPackageId', OuterPackageId int '$.OuterPackageId',

Brand nvarchar(20) '$.Brand', LeadTimeDays int '$.LeadTimeDays', QuantityPerOuter int '$.QuantityPerOuter',

TaxRate decimal(18,3) '$.TaxRate', UnitPrice decimal(18,2) '$.UnitPrice', RecommendedRetailPrice decimal(18,2) '$.RecommendedRetailPrice',

TypicalWeightPerUnit decimal(18,3) '$.TypicalWeightPerUnit', CountryOfManufacture nvarchar(max) '$.CountryOfManufacture',

OrderDate nvarchar(20) '$.OrderDate',SupplierReference nvarchar(20) '$.SupplierReference'

)as jtable

1. **Revisit your answer in (19). Convert the result in JSON string and save it to the server using TSQL FOR JSON PATH**

SELECT \* FROM Q19 FOR JSON PATH

1. **Revisit your answer in (19). Convert the result into an XML string and save it to the server using TSQL FOR XML PATH.**

WITH SoldYear AS(

SELECT StockGroupID, Sold\_year,SUM(Quantity) AS QuantityYear

FROM (

SELECT s3.StockGroupID,year(o.OrderDate) AS Sold\_year, ol.Quantity

FROM Sales.OrderLines ol JOIN Sales.Orders o ON ol.OrderID = o.OrderID

JOIN Warehouse.StockItems s1 ON ol.StockItemID = s1.StockItemID

JOIN Warehouse.StockItemStockGroups s2 ON s2.StockItemID = s1.StockItemID

JOIN Warehouse.StockGroups s3 ON s2.StockGroupID = s3.StockGroupID

) AS t1

GROUP BY Sold\_year,StockGroupID

), PivotTable AS (

SELECT\* FROM SoldYear

PIVOT(

SUM(QuantityYear) FOR StockGroupID IN([1],[2],[3],[4],[5],[6],[7],[8],[9],[10])

)AS t2

), Q19 AS(

SELECT "1" as 'NoveltyItems', "2" as 'Clothing', "3" as 'Mugs', "4" as 'T-Shirts', "5" as 'AirlineNovelties'

, "6" as 'ComputingNovelties', "7" as 'USBNovelties', "8" as 'FurryFootwear', "9" as 'Toys', "10" as 'PackagingMaterials'

FROM PivotTable)

SELECT \* FROM Q19 FOR XML PATH('GroupQuantity'), ROOT('Year')

****

1. **Create a new table called ods.ConfirmedDeviveryJson with 3 columns (id, date, value) . Create a stored procedure, input is a date. The logic would load invoice information (all columns) as well as invoice line information (all columns) and forge them into a JSON string and then insert into the new table just created. Then write a query to run the stored procedure for each DATE that customer id 1 got something delivered to him.**
2. **Write a short essay talking about your understanding of transactions, locks and isolation levels.**

Transaction is SQL’s save/undo button. Most of the transactions happens in OLTP. A proper relational transaction database needs to satisfy the ACID rules. A as atomicity, whenver you execute a command, the execution either all go through or none; C as consistency to ensure the quality of data; I adn isolation for datas not interfuse with each other and D ad durability, result of transaction should be permanent. These principle makes Transaction so important, because it ensure that our data never fails into an inconsistent state, causing by operations that only partialt completes.

Lock is the way that SQL manages transaction concurrency. Locks ensures the integrity of the data by forces every transaction to pass the ACID principle. There are many locks for different scenario, like index lock, table lock, database lock and so on. when objects are locked, SQL server will prevent other transactions from making changes of the stored data. There are 5 types of Lock Mode: shared(S), updated(U), exclusive(X), Intent(I), Schema(Sch) and Bulk Update(BU). Also, deadlock occurs when exclusive locks are held on resources required by several process.

Isolation level is data modifications mad by different transactions. Isolation determines how transaction integrity is visible to other users and systems. It define the degree to which a transaction must be isolated from the data modifications made by any other transaction in the database system, which has five level of phenomena: Read Uncommitted, Read Committted, Repeatable Read, Serializable, and Snapshot. Snapshots follows the optimistic model and avoids most blocking by row versioning. Read Uncommitted allows one transaction to read the data before the data changed from another process meanwhile read committed only allowed to read data that is committed. In repeatable level, transaction has to wait till other on is complete. The highest level, serializable, we can ask any transaction to wait until the current one completes.

1. **Write a short essay, plus screenshots talking about performance tuning in SQL Server. Must include Tuning Advisor, Extended Events, DMV, Logs and Execution Plan.**