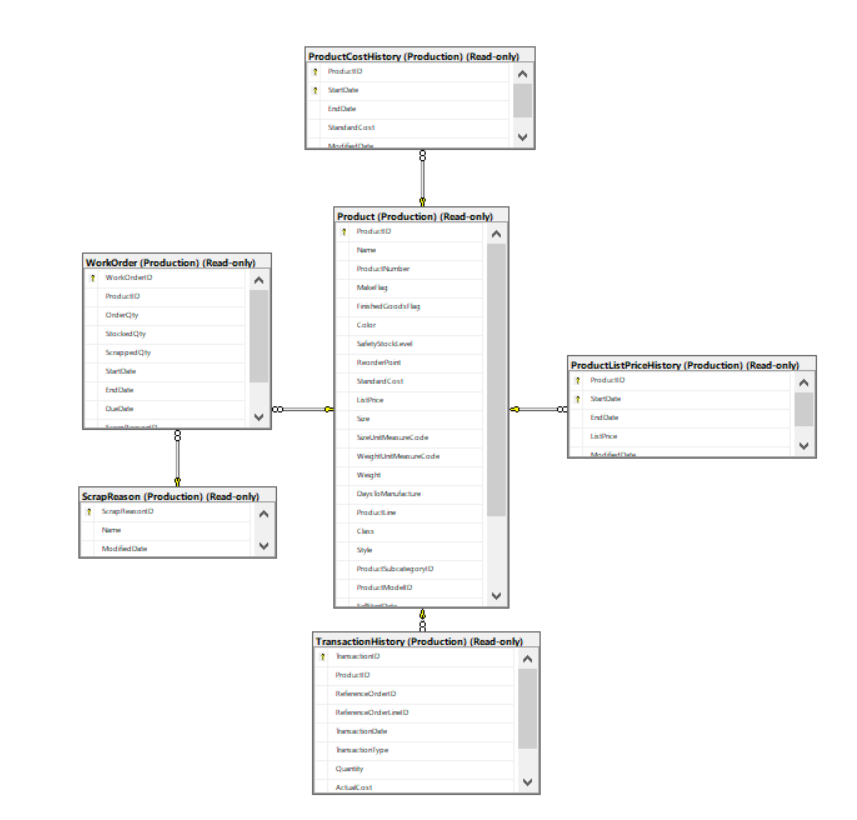
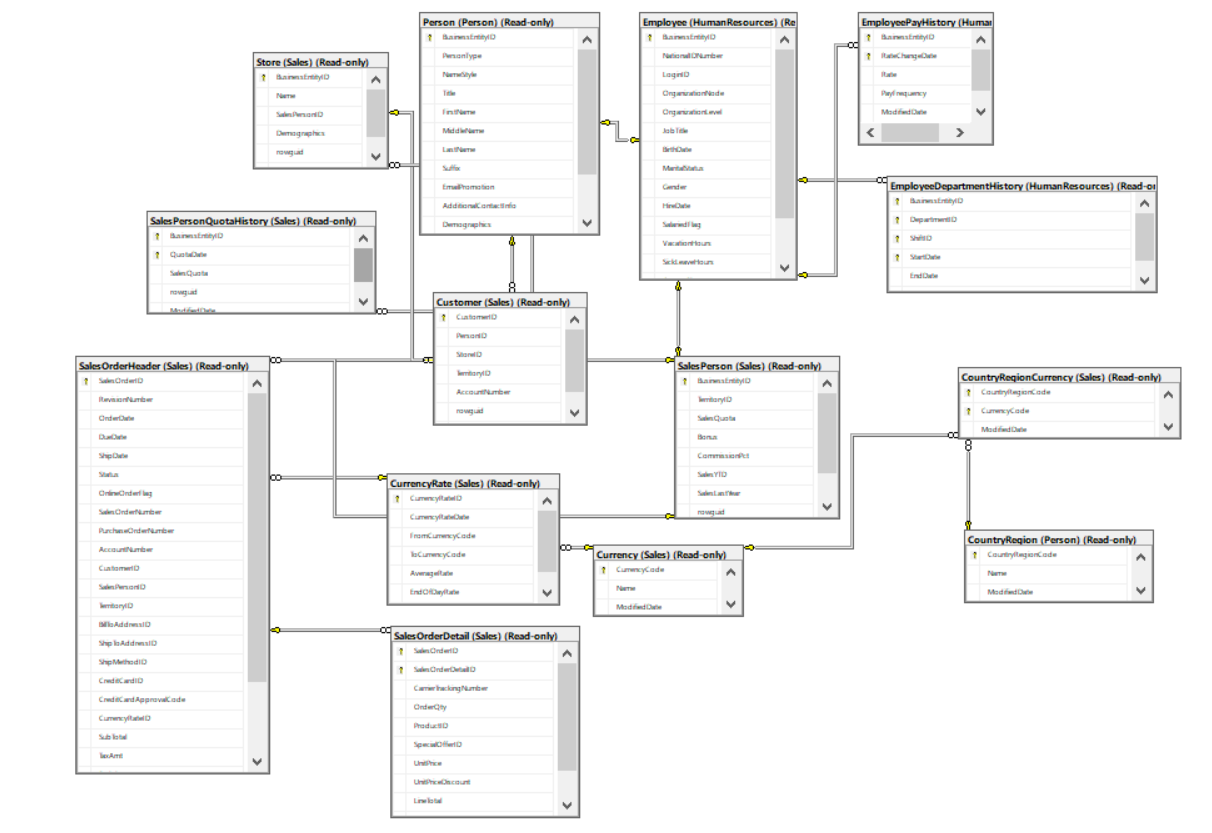
Alexis Montes - Group G9-1

AdventureWorks2014 Diagrams, with relevant tables used

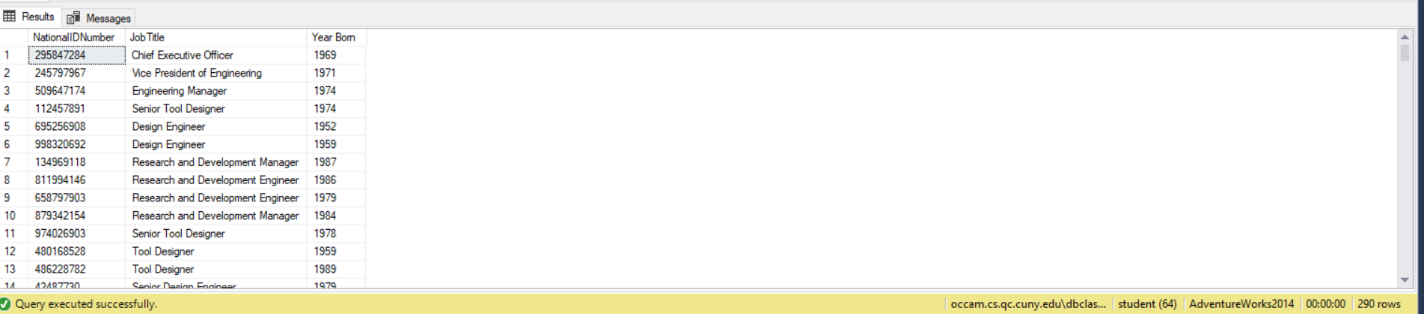
Adventure Works queries:

Query 1: Simple 1

Proposition: Find the list of employees born before the year 2000.

Query:

* USE AdventureWorks2014
* SELECT NationalIDNumber
* ,JobTitle
* ,DATEPART(year, BirthDate) AS [Year Born]
* FROM HumanResources.Employee
* WHERE DATEPART(year, BirthDate) < 2000;

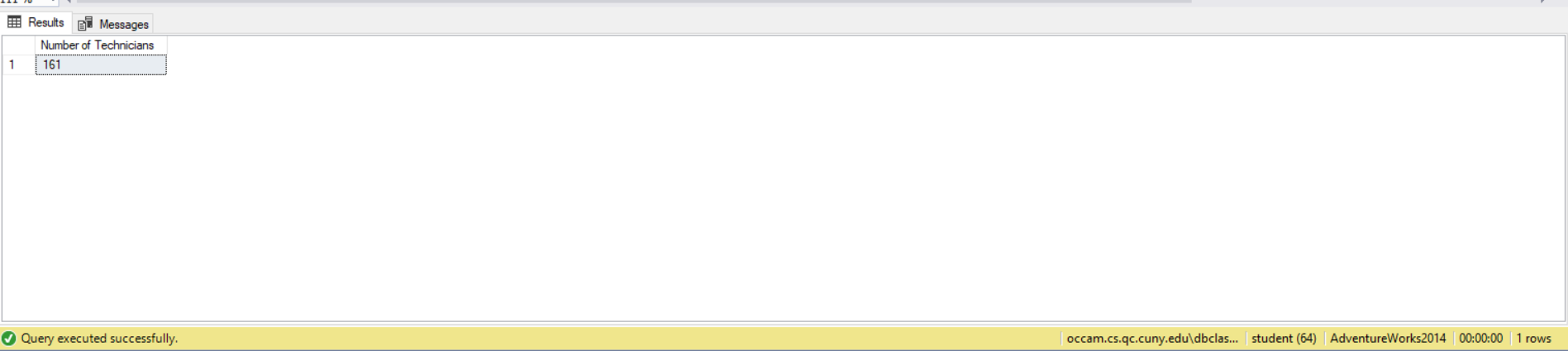


Query 2: Simple 2

Proposition: Find the employees that were salespersons last year and have made sales year to date that have made more then the previous year

Query:

* USE AdventureWorks2014
* SELECT BusinessEntityID
* ,SalesYTD
* ,SalesLastYear
* FROM Sales.SalesPerson
* WHERE SalesYTD > SalesLastYear
* AND SalesLastYear != 0.00;

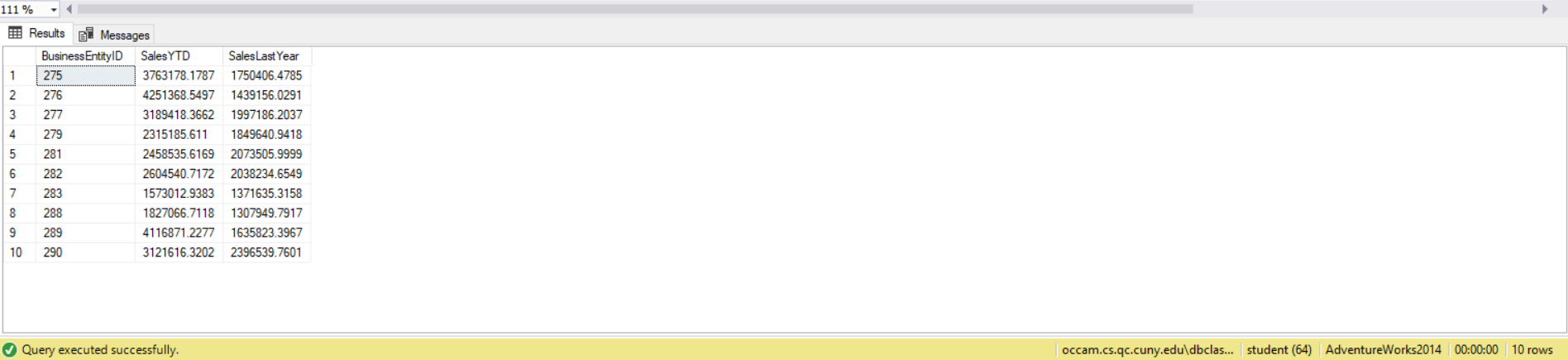


Query 3: Simple 3

Proposition: Find the employees that were employed as a technitician based on their job title

Query:

* USE AdventureWorks2014
* SELECT COUNT(NationalIDNumber) AS [Number of Technicians]
* FROM HumanResources.Employee
* WHERE JobTitle LIKE '%Technician%';

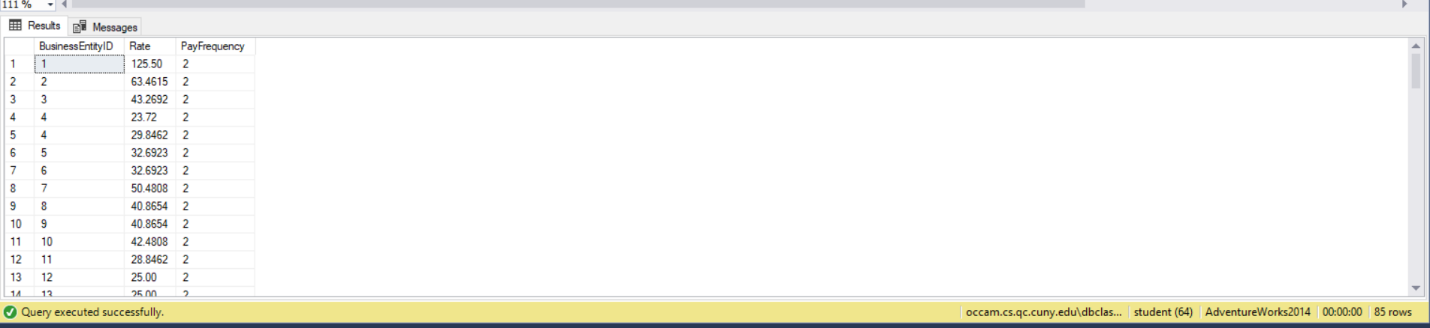


Query 4: Simple 4

Proposition: Find the list of employees that are paid over $20 and more then once a week

Query:

* USE AdventureWorks2014
* SELECT BusinessEntityID
* ,Rate
* ,PayFrequency
* FROM HumanResources.EmployeePayHistory
* WHERE Rate > 20
* AND PayFrequency > 1;

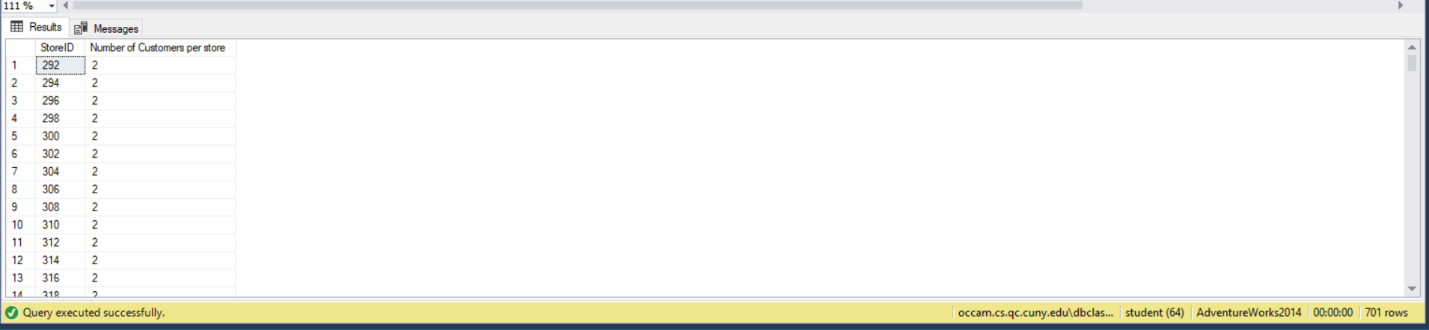


Query 5: Simple 5

Proposition: Create a list matching the store id and the number of customers that particular store has.

Query:

* USE AdventureWorks2014
* SELECT StoreID
* ,COUNT(CustomerID) AS [Number of Customers per store]
* FROM Sales.Customer
* WHERE (StoreID IS NOT NULL)
* GROUP BY StoreID
* ORDER BY StoreID;

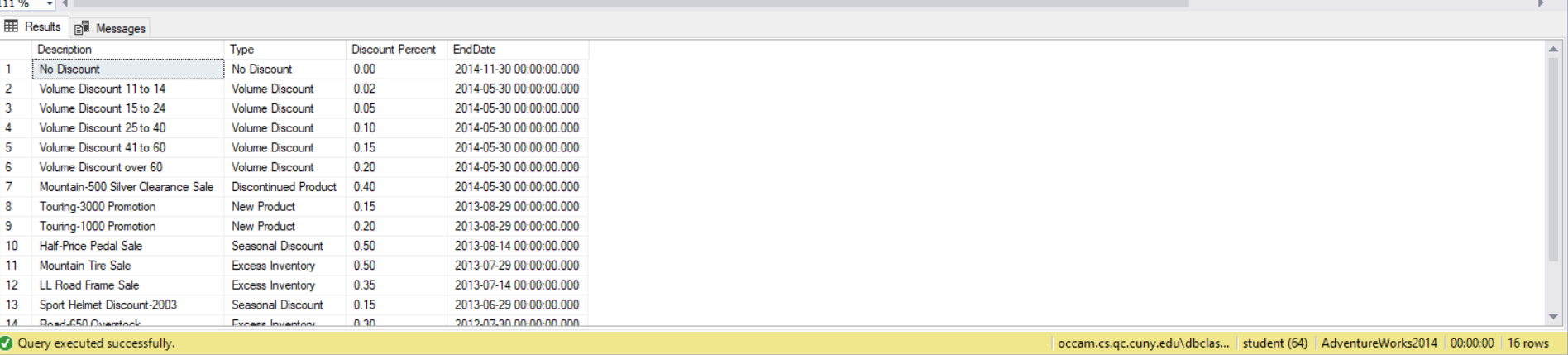


Query 6: Simple 6

Proposition: Show the list of discounts available sorted by their end date in descending order so that first to end is the first result.

Query:

* USE AdventureWorks2014
* SELECT [Description]
* ,[Type]
* ,DiscountPct AS [Discount Percent]
* ,EndDate
* FROM Sales.SpecialOffer
* ORDER BY EndDate DESC;

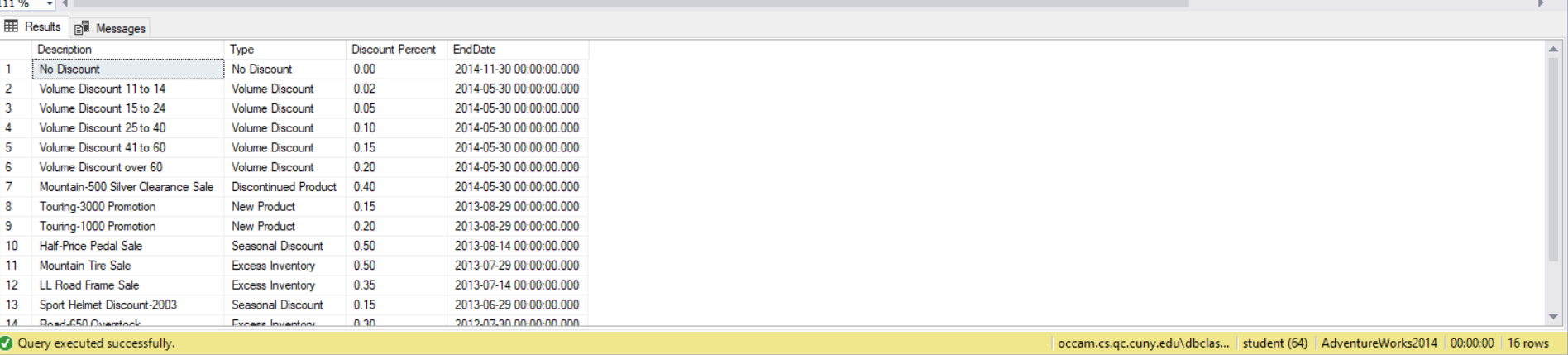


Query 7: Simple 7

Proposition: Display the list of products that have reviews and display their reviewer(person), their rating and the date it was reviewed

Query:

* USE AdventureWorks2014
* SELECT P.ProductID
* ,P.[Name]
* ,P.ListPrice
* ,PR.ReviewerName
* ,PR.Rating
* ,PR.Comments
* ,PR.ReviewDate
* FROM Production.Product AS P
* INNER JOIN Production.ProductReview AS PR ON P.ProductID = PR.ProductID;

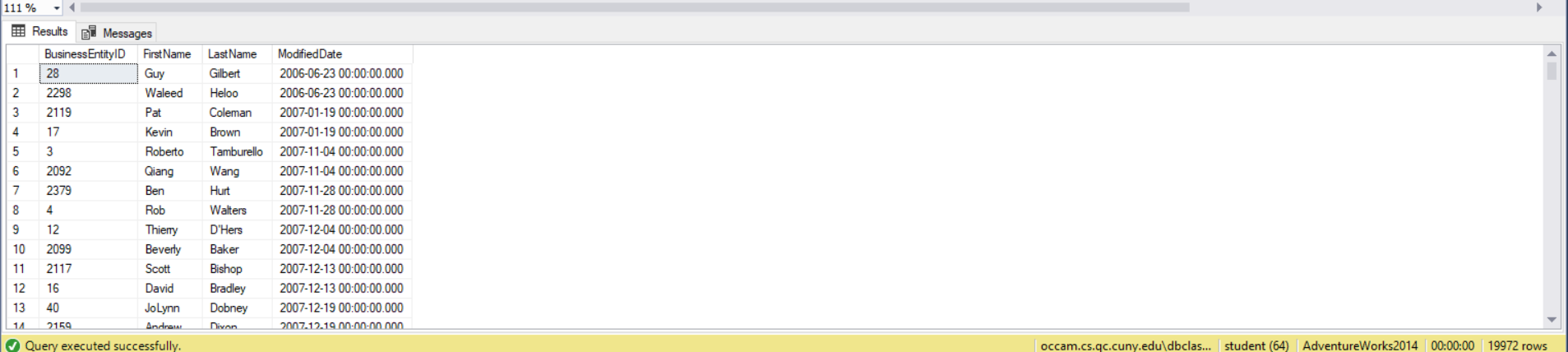


Query 8: Simple 8

Proposition: Sort the list of employees by last by last modified date

Query:

* USE AdventureWorks2014
* SELECT BusinessEntityID
* ,FirstName
* ,LastName
* ,ModifiedDate
* FROM Person.Person
* ORDER BY ModifiedDate;

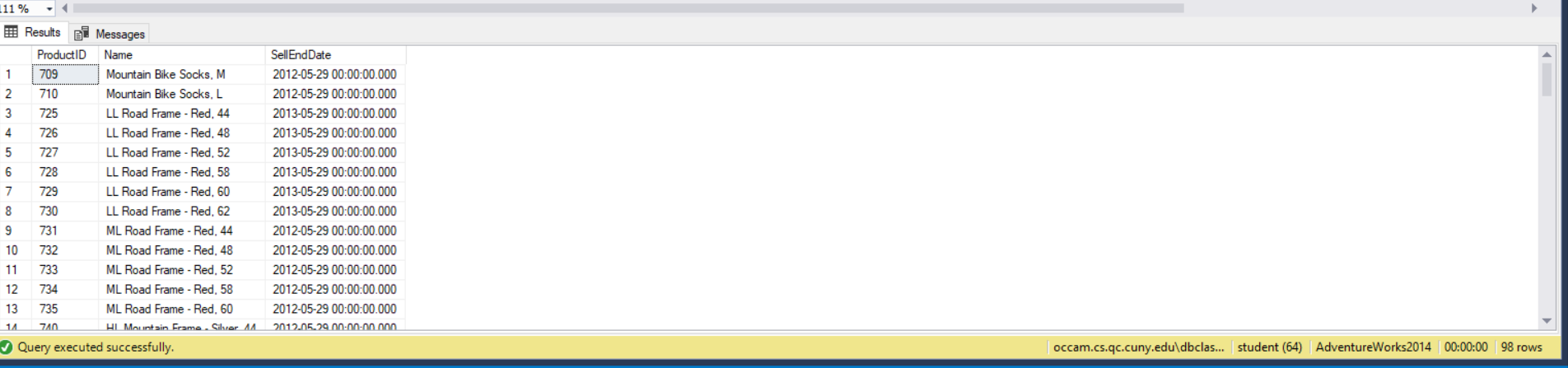


Query 9: Simple 9

Proposition: Create a list of Products that have reached their end of sale date, so that it creates a list of discontinued products.

Query:

* USE AdventureWorks2014
* SELECT ProductID
* ,[Name]
* ,SellEndDate
* FROM Production.Product
* WHERE SellEndDate < SYSDATETIME();

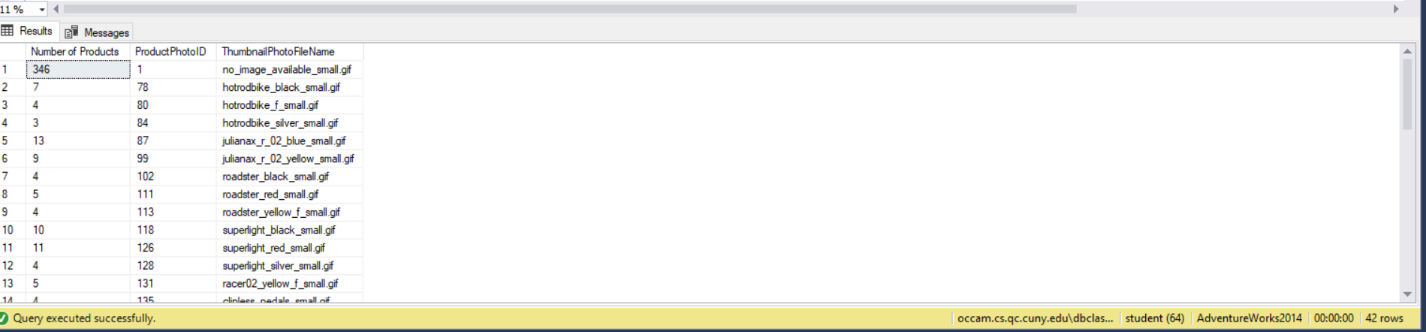


Query 10: Simple 10

Proposition: Find how many products use a specific photo file, that the company uses.

Query:

* USE AdventureWorks2014
* SELECT COUNT(PPP.ProductPhotoID) AS [Number of Products]
* ,PPP.ProductPhotoID
* ,PP.ThumbnailPhotoFileName
* FROM Production.ProductProductPhoto AS PPP
* INNER JOIN Production.ProductPhoto AS PP ON PPP.ProductPhotoID = PP.ProductPhotoID
* GROUP BY PPP.ProductPhotoID
* ,PP.ThumbnailPhotoFileName;

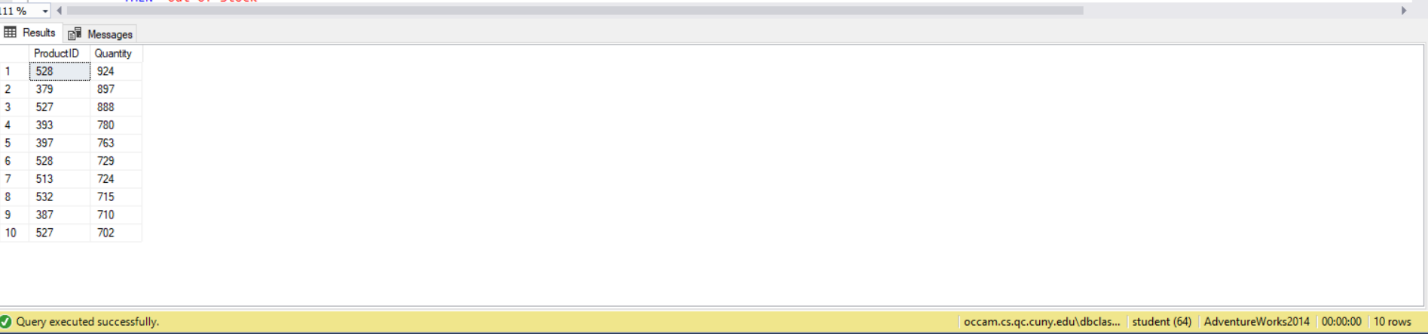


Query 11: Simple 11

Proposition: Find the top 10 products that have the most quantity in their inventory:

Query:

* USE AdventureWorks2014
* SELECT TOP (10) ProductID
* ,Quantity
* FROM Production.ProductInventory
* ORDER BY Quantity DESC;

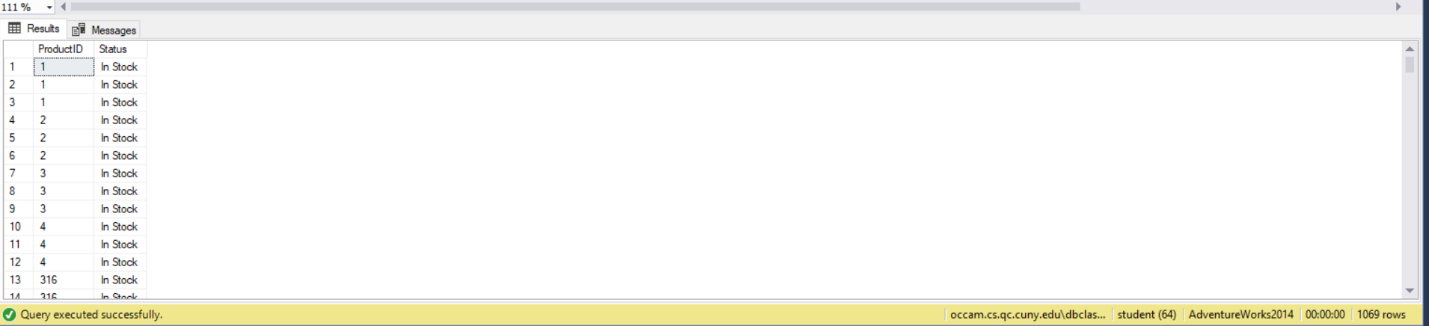


Query 12: Simple 12

Proposition: Create a list of products and use a case statement where it printed if the items was in stock or not:

Query:

* USE AdventureWorks2014
* SELECT ProductID
* ,CASE
* WHEN Quantity = 0
* THEN 'Out Of Stock'
* ELSE 'In Stock'
* END AS [Status]
* FROM Production.ProductInventory
* ORDER BY ProductID;

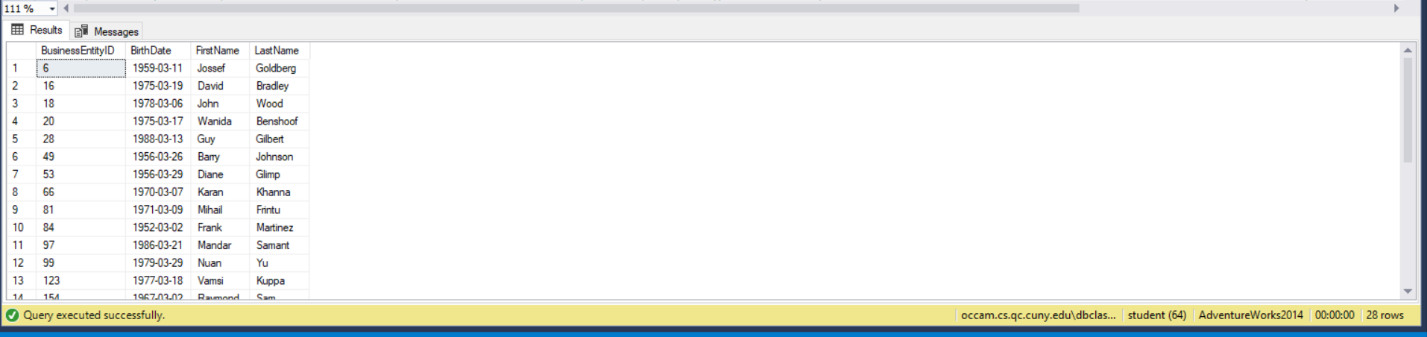


Query 13: Medium 1

Proposition: Find the list of employees who have birthdays in March

Query:

* USE AdventureWorks2014
* SELECT HumanResources.Employee.BusinessEntityID
* ,BirthDate
* ,Person.Person.FirstName
* ,Person.Person.LastName
* FROM HumanResources.Employee
* INNER JOIN Person.Person ON Person.Person.BusinessEntityID = HumanResources.Employee.BusinessEntityID
* WHERE DATEPART(month, BirthDate) = 3
* GROUP BY HumanResources.Employee.BusinessEntityID
* ,Person.Person.FirstName
* ,Person.Person.LastName
* ,BirthDate;

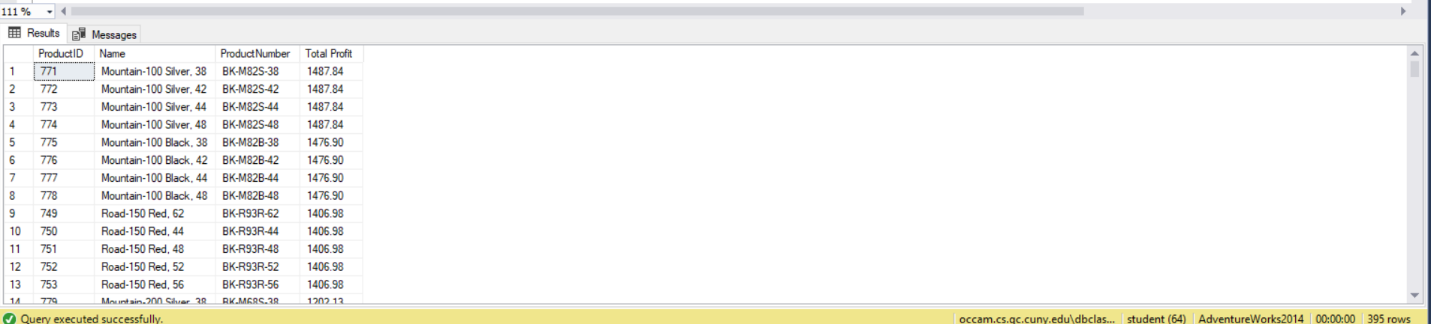


Query 14: Medium 2

Proposition: Create a list of products that the company sells and calculate the amount of profit is made on each item

Query:

* USE AdventureWorks2014
* SELECT P.ProductID
* ,P.Name
* ,ProductNumber
* ,ROUND(P.ListPrice - PCH.StandardCost, 2) AS [Total Profit]
* FROM Production.Product AS P
* INNER JOIN Production.ProductCostHistory AS PCH ON PCH.ProductID = P.ProductID
* GROUP BY P.ProductID
* ,P.Name
* ,ProductNumber
* ,P.ListPrice
* ,PCH.StandardCost
* ORDER BY [Total Profit] DESC;

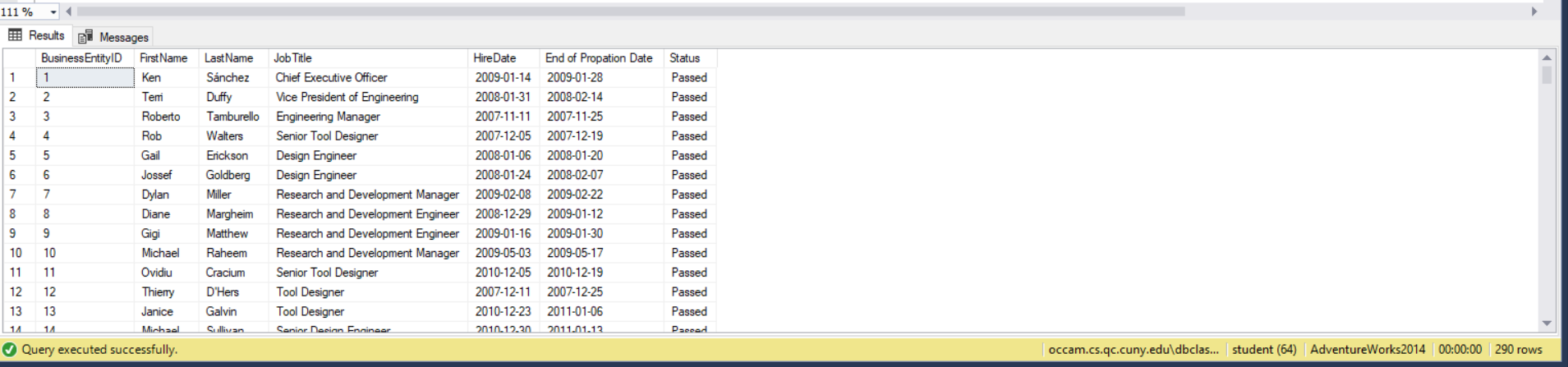


Query 15: Medium 3

Proposition: Show the list of employees, with the date of their end of 2 week probation and if they are still under probation, by comparing todays date and the date their probation ends

Query:

* USE AdventureWorks2014
* SELECT E.BusinessEntityID
* ,P.FirstName
* ,P.LastName
* ,E.JobTitle
* ,E.HireDate
* ,DATEADD(week, 2, E.HireDate) AS [End of Propation Date]
* ,CASE
* WHEN DATEADD(week, 2, E.HireDate) < SYSDATETIME()
* THEN 'Passed'
* ELSE 'Still Under'
* END AS [Status]
* FROM HumanResources.Employee AS E
* INNER JOIN Person.Person AS P ON P.BusinessEntityID = E.BusinessEntityID
* GROUP BY E.BusinessEntityID
* ,P.FirstName
* ,P.LastName
* ,E.JobTitle
* ,E.HireDate;

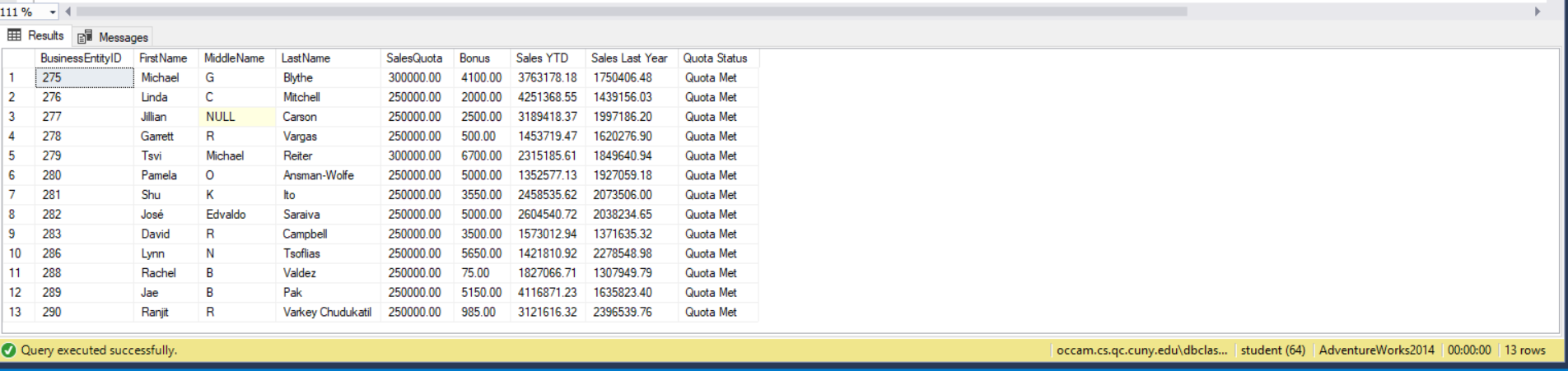


Query 16: Medium 4

Proposition: Create a summery table of Sales persons from within the company, and create a column using a case statement to see if they have met their quota

Query:

* USE AdventureWorks2014
* SELECT SP.BusinessEntityID
* ,P.FirstName
* ,P.MiddleName
* ,P.LastName
* ,SP.SalesQuota
* ,SP.Bonus
* ,ROUND(SP.SalesYTD, 2) AS [Sales YTD]
* ,ROUND(SP.SalesLastYear, 2) AS [Sales Last Year]
* ,CASE
* WHEN SalesYTD > SalesQuota
* THEN 'Quota Met'
* ELSE 'Quota not yet met'
* END AS [Quota Status]
* FROM Sales.SalesPerson AS SP
* INNER JOIN Person.Person AS P ON SP.BusinessEntityID = P.BusinessEntityID
* WHERE SP.SalesQuota IS NOT NULL
* AND SP.SalesLastYear IS NOT NULL
* AND SP.SalesLastYear != 0.00;

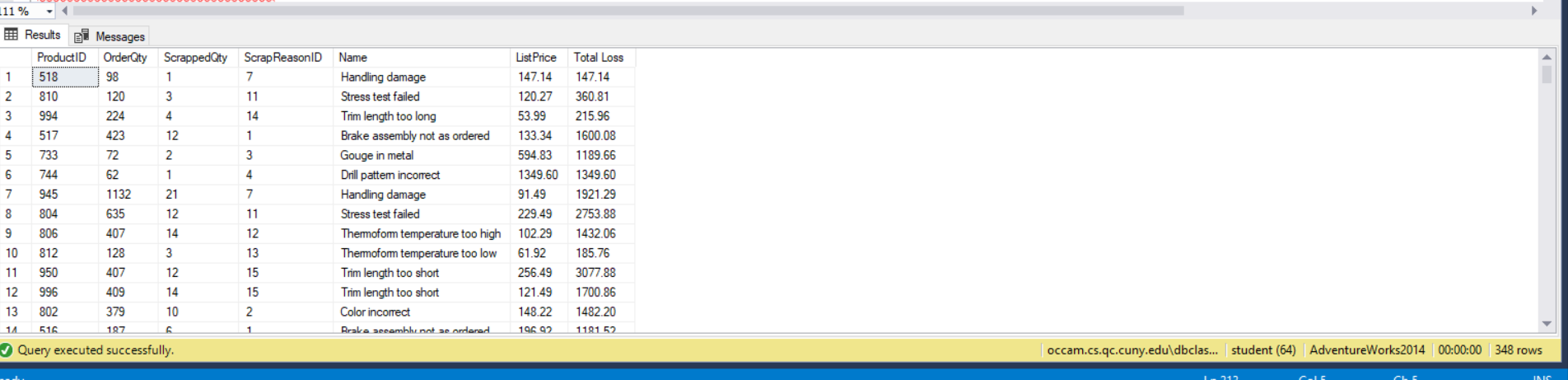


Query 17: Medium 5

Proposition: Find how much products were lossed during production of the product with the reason the product was scraped and how much potential revenue was lost.

Query:

* SELECT WO.ProductID
* ,WO.OrderQty
* ,WO.ScrappedQty
* ,WO.ScrapReasonID
* ,SR.[Name]
* ,P.ListPrice
* ,ROUND((P.ListPrice \* WO.ScrappedQty), 2) AS [Total Loss]
* FROM Production.WorkOrder AS WO
* INNER JOIN Production.ScrapReason AS SR ON WO.ScrapReasonID = SR.ScrapReasonID
* INNER JOIN Production.Product AS P ON WO.ProductID = P.ProductID
* WHERE WO.ScrappedQty > 0
* AND P.ListPrice != 0.00;



Query 18: Complex 1

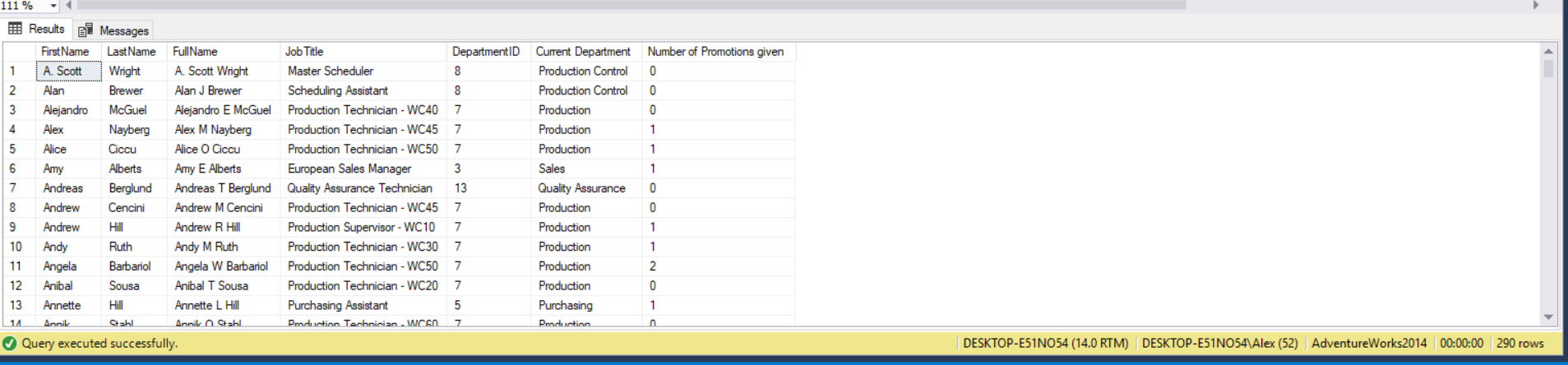
Proposition: Create a summery of employees with their full name created by a function and other details about said employee like their title, department info, and how many times they have been promoted

Query:

* USE AdventureWorks2014
* SELECT P.FirstName
* ,P.LastName
* , dbo.getFullName(P.FirstName, P.MiddleName, P.LastName) as FullName
* ,E.JobTitle
* ,EDH.DepartmentID
* ,D.Name AS [Current Department]
* ,P.EmailPromotion AS [Number of Promotions given]
* FROM Person.Person AS P
* INNER JOIN HumanResources.Employee AS E ON P.BusinessEntityID = E.BusinessEntityID
* INNER JOIN HumanResources.EmployeeDepartmentHistory AS EDH ON P.BusinessEntityID = EDH.BusinessEntityID
* INNER JOIN HumanResources.Department AS D ON EDH.DepartmentID = D.DepartmentID
* WHERE EDH.EndDate IS NULL
* GROUP BY P.FirstName
* ,P.MiddleName
* ,P.LastName
* ,E.JobTitle
* ,EDH.DepartmentID
* ,D.Name
* ,P.EmailPromotion;

Function:

* CREATE FUNCTION getFullName (
* @firstName VARCHAR(250)
* ,@middleName VARCHAR(250)
* ,@lastName VARCHAR(250)
* )
* RETURNS VARCHAR(250)
* AS
* BEGIN
* IF @middleName IS NOT NULL
* BEGIN
* RETURN CONCAT (
* @firstName
* ,' '
* ,@middleName
* ,' '
* ,@lastName
* )
* END
* RETURN CONCAT (
* @firstName
* ,' '
* ,@lastName
* )
* END;
* GO



Query 19: Complex 2

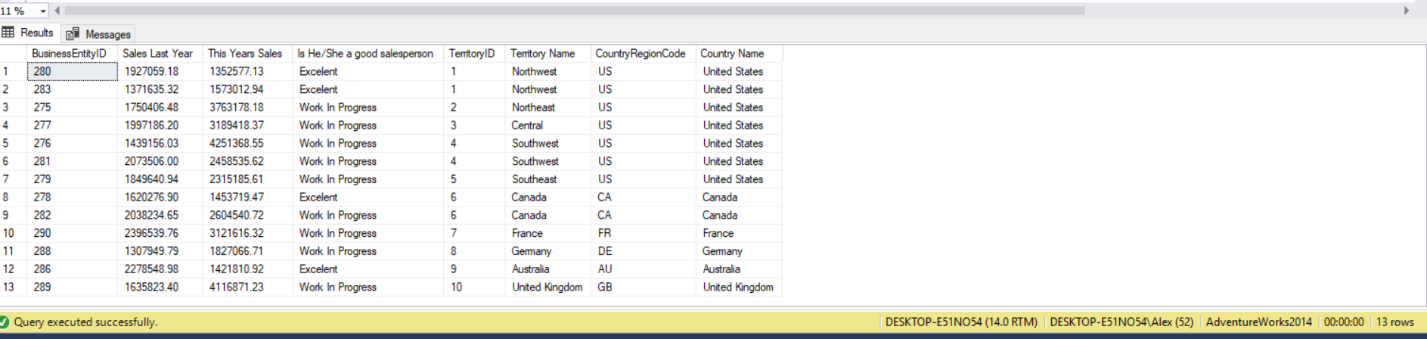
Proposition: Create a summery of salesperson, their sales data from last year, this year and if they can be considered a good salesperson if they have reached their quota plus an additional 15%

Query:

* USE AdventureWorks2014
* SELECT SP.BusinessEntityID
* ,ROUND(SP.SalesLastYear, 2) AS [Sales Last Year]
* ,ROUND(SP.SalesYTD, 2) AS [This Years Sales]
* ,dbo.goodSalesPerson(ROUND(SP.SalesLastYear, 2), ROUND(SP.SalesYTD, 2)) AS [Is He/She a good salesperson]
* ,SP.TerritoryID
* ,ST.Name AS [Territory Name]
* ,ST.CountryRegionCode
* ,CR.Name AS [Country Name]
* FROM Sales.SalesPerson AS SP
* INNER JOIN Sales.SalesTerritory AS ST ON SP.TerritoryID = ST.TerritoryID
* INNER JOIN Person.CountryRegion AS CR ON ST.CountryRegionCode = CR.CountryRegionCode
* WHERE SP.SalesLastYear != 0.00
* GROUP BY SP.BusinessEntityID
* ,SP.SalesLastYear
* ,SP.SalesYTD
* ,SP.TerritoryID
* ,ST.Name
* ,ST.CountryRegionCode
* ,CR.Name
* ORDER BY SP.TerritoryID;

Function:

* CREATE FUNCTION goodSalesPerson (
* @lastYear DECIMAL(10, 2)
* ,@thisYear DECIMAL(10, 2)
* )
* RETURNS VARCHAR(250)
* AS
* BEGIN
* IF (@lastYear \* 1.15 >= @thisYear)
* BEGIN
* RETURN 'Excelent'
* END
* RETURN 'Work In Progress'
* END;



Query 20: Complex 3

Proposition Create a string representation using a function that displays how much the total sale was in the foreign currency used and other details of the sale.

Query:

* USE AdventureWorks2014
* SELECT SOH.SalesOrderID
* ,SOH.SalesOrderNumber
* ,SOH.TerritoryID
* ,SOH.CurrencyRateID
* ,SOH.TotalDue AS [Total Due in Dollars]
* ,CR.ToCurrencyCode
* ,C.Name
* ,dbo.getTotalValue(CAST(SOH.CurrencyRateID AS INT), CAST(SOH.TotalDue AS DECIMAL(10, 2))) AS [Total in foregn currency]
* FROM Sales.SalesOrderHeader AS SOH
* INNER JOIN Sales.CurrencyRate AS CR ON SOH.CurrencyRateID = CR.CurrencyRateID
* INNER JOIN Sales.Currency AS C ON C.CurrencyCode = CR.ToCurrencyCode;

Function:

* CREATE
* OR
* ALTER FUNCTION getTotalValue (
* @id INT
* ,@total DECIMAL(10, 2)
* )
* RETURNS VARCHAR(250)
* AS
* BEGIN
* DECLARE @toRate DECIMAL(10, 2)
* ,@finalAmt DECIMAL(10, 2)
* ,@CName VARCHAR(250)
* SET @toRate = CAST((
* SELECT AverageRate
* FROM Sales.CurrencyRate
* WHERE CurrencyRateID = @id
* ) AS INT);
* SET @finalAmt = @total \* @toRate;
* SET @CName = (
* SELECT ToCurrencyCode
* FROM Sales.CurrencyRate
* WHERE @id = CurrencyRateID
* )
* RETURN CONCAT (
* CAST(@finalAmt AS VARCHAR(250))
* ,' '
* ,CAST((
* SELECT NAME
* FROM Sales.Currency
* WHERE @CName = CurrencyCode
* ) AS VARCHAR(250))
* ,'s'
* )
* END;

