For merging Logon and Person information into World Wide Importers database, these two parts of information can be merged into only one table called Application.People. Firstly, in order to make sure the data integrity of primary key constraint, We need to DECLARE a variable called @maxid that stores the maximum PersonID and then add this maximum value to the BussinessEntityID. This operation makes sure that we will not violate the primary key constraint. Since names in Adventure Works database are split into first names and last names, we need to combine first and last name together to make sure we can match it to Application.People. For IsPermittedToLogon, IsExternalLogonProvider and IsSystemUser information that is not allowed to be NULL and can not be found in Adventures Works database, we set all to 0. For PassWordHash information, we convert it to VARBINARY data type to match it to main database. For IsEmployee and IsSalesperson information, we use two LEFT JOIN operation to check whether it is 1 or 0. Because some PhoneNumber information has different data structures, we use SUBSTRING function to extract same part.

Graphical user interface

Description automatically generated with medium confidence

## This part is to merge data into person table

DECLARE @maxid INT;

DECLARE @col NVARCHAR(MAX) = '';

DECLARE @query NVARCHAR(MAX);

SELECT @maxid = MAX(personID)

FROM Application.People

SELECT p.BusinessEntityID + @maxid AS PersonID,

p.FirstName + ' ' + p.LastName AS FullName, p.FirstName AS PreferredName,

0 AS IsPermittedToLogon, e.EmailAddress AS LogonName,

0 AS IsExternalLogonProvider, CONVERT(VARBINARY, pa.PasswordHash) AS HashedPassword,

0 AS IsSystemUser,

CASE WHEN em.BusinessEntityID IS NOT NULL THEN 1 ELSE 0 END AS IsEmployee,

CASE WHEN s.BusinessEntityID IS NOT NULL THEN 1 ELSE 0 END AS IsSalesperson,

'(' + LEFT(RIGHT(ph.PhoneNumber, 12), 3) + ') ' + RIGHT(ph.PhoneNumber, 8) AS PhoneNumber,

e.EmailAddress AS EmailAddress, 1 AS LastEditedBy

INTO #person

FROM AdventureWorks2019.Person.Person p

JOIN AdventureWorks2019.Person.EmailAddress e ON p.BusinessEntityID = e.BusinessEntityID

JOIN AdventureWorks2019.Person.[Password] pa ON p.BusinessEntityID = pa.BusinessEntityID

LEFT JOIN AdventureWorks2019.HumanResources.Employee em ON p.BusinessEntityID = em.BusinessEntityID

LEFT JOIN AdventureWorks2019.Sales.SalesPerson s ON p.BusinessEntityID = s.BusinessEntityID

JOIN AdventureWorks2019.Person.PersonPhone ph ON p.BusinessEntityID = ph.BusinessEntityID

SELECT @col = @col + name +','

FROM tempdb.sys.columns

WHERE OBJECT\_ID = OBJECT\_ID('tempdb..#person')

SET @col = SUBSTRING(@col, 0, LEN(@col))

SET @query = 'INSERT INTO Application.People (' + @col +')

SELECT \*

FROM #person'

EXEC(@query)

For the part of product table in WideWorldImporters, first we have to an overall look at columns, focusing on whether a column is the primary key or foreign key or not null. The fact is that we have two unique id columns for product with product id and supplier id respectively. Since the corresponding id columns for products in AdventureWorks2019 are assigned for another line of ids, we must reassign the id column in product table in AdventureWorks2019 as a different series of id to avoid merge conflict. The rule is that we want to assign the id for new products from the new company continuing from the maximum id number of datasets in old system. Next phase is to find all columns with not null constraints because when we insert the new data in the old columns, we do not want the new data makes the conflict. The related columns involve StockItemName and LeadTimeDays, etc. However, sometimes we cannot find the corresponding column in the new table and the temporary solution is to assign a value to represent that we cannot find the value. For int of float value, we assign -1 because -1 cannot be used for price or quantity sold. For strings, we usually assign ‘Unknown’ to distinguish with other known values.

Another issue our group notice is that supplier id is a foreign key in product table in WideWorldImporters. Therefore, we also assign new id for these new suppliers in the original supplier table in WideWorldImporters.

After we grabbed all necessary and available data from new company dataset, we saved all these data into a temporary table and did the final check for all data types and constraints. The following is an example for the temp table.



Finally the data of product from AdventureWorks2019 was successfully merged into WideWorldImporters. The following part is the whole query we used.

## This part is to grab all corresponding columns

DECLARE @maxproductid INT;

DECLARE @maxsupplierid INT;

SELECT @maxproductid = MAX(StockItemID)

FROM Warehouse.StockItems

SELECT @maxsupplierid = MAX(SupplierID)

FROM Warehouse.StockItems

SELECT p.ProductID + @maxproductid AS StockItemID, p.[Name] AS StockItemName,

pv.BusinessEntityID + @maxsupplierid AS SupplierID, -1 AS UnitPackageID, -1 AS OuterPackageID,

v.Name AS Brand, p.Size, DaysToManufacture AS LeadTimeDays, -1 AS QuantityPerOuter,

0 AS IsChillerStock, 0 AS TaxRate, sd.UnitPrice as UnitPrice, -1 AS RecommendedRetailPrice,

p.Weight AS TypicalWeightPerUnit, pr.Comments AS MarketingComments, 1 AS LastEditedBy

INTO #product

FROM AdventureWorks2019.Production.Product as p

JOIN AdventureWorks2019.Purchasing.ProductVendor pv ON p.ProductID = pv.ProductID

JOIN AdventureWorks2019.Purchasing.Vendor v ON pv.BusinessEntityID = v.BusinessEntityID

JOIN AdventureWorks2019.Sales.SalesOrderDetail sd ON sd.ProductID = p.ProductID

JOIN AdventureWorks2019.Production.ProductReview pr ON p.ProductID = pr.ProductID

## This part is to add new supplier id in the supplier table.

DECLARE @maxsupplierid INT;

SELECT @maxsupplierid = MAX(SupplierID)

FROM Purchasing.Suppliers

SELECT v.BusinessEntityID + @maxsupplierid AS SupplierID, v.Name AS SupplierName, -1 AS SupplierCategoryID,

-1 AS AlternateContactPersonID, -1 AS DeliveryCityID, -1 AS PostalCityID, 0 AS PaymentDays,

'Unknown' AS PhoneNumber, 'Unknown' AS FaxNumber, isnull(v.PurchasingWebServiceURL, 'Unknown') AS WebsiteURL,

'Unknown' AS DeliveryAddressLine1, 'Unknown' AS DeliveryPostalCode, 'Unknown' AS DeliveryAddressLine1,

'Unknown' AS DeliveryPostalCode, 'Unknown' AS PostalAddressLine1, 'Unknown' AS PostalPostalCode, 1 AS LastEditedBy

INTO #supplier

FROM AdventureWorks2019.Purchasing.Vendor as v

## Finally merge everything into old dataset

INSERT INTO Warehouse.StockItems

(StockItemID, StockItemName, SupplierID, UnitPackageID, OuterPackageID,

Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, TaxRate,

UnitPrice, RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, LastEditedBy)

SELECT \* FROM tempdb..#product