Report

Laboratory Work 3

Dmitry Ladutsko

August 27, 2022

# 1. Exercise 1: Creating new database for HR department

You have to create a new database for HR department. The name of the database must be «HumanResources».The database must contain the following tables:

**dbo.EmployeesExternal:**

|  |  |  |
| --- | --- | --- |
| Column name | Type | Null? |
| EmployeeID (key) | INT | NOT NULL |
| FirstName | NVARCHAR(50) | NOT NULL |
| LastName | NVARCHAR(50) | NOT NULL |
| JobTitle | NVARCHAR(50) | NOT NULL |
| EmailAddress | NVARCHAR(50) | NULL |
| City | NVARCHAR(50) | NOT NULL |
| StateProvinceName | NVARCHAR(50) | NOT NULL |
| CountryRegionName | NVARCHAR(50) | NOT NULL |

**dbo.EmployeesAW:**

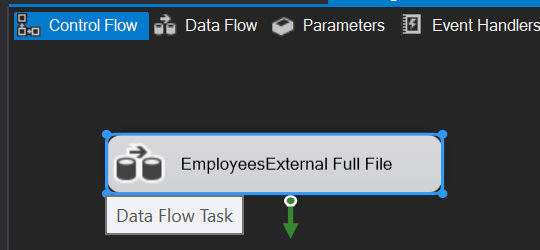
|  |  |  |
| --- | --- | --- |
| Column name | Type | Null? |
| BusinessEntityID (key) | INT | NOT NULL |
| FirstName | NVARCHAR(50) | NOT NULL |
| LastName | NVARCHAR(50) | NOT NULL |
| JobTitle | NVARCHAR(50) | NULL |

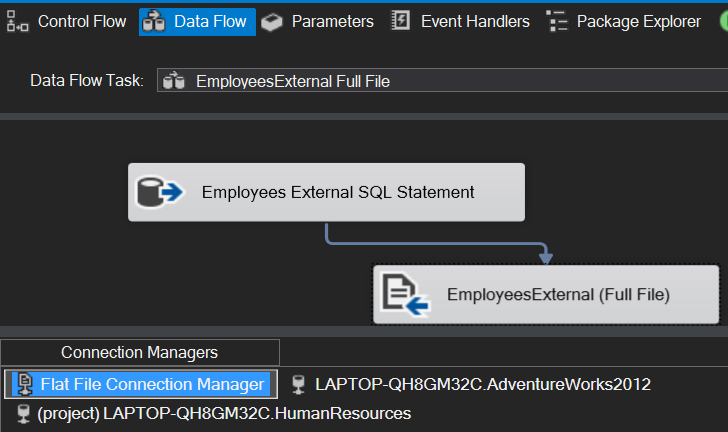
**dbo.EmailAddressesAW:**

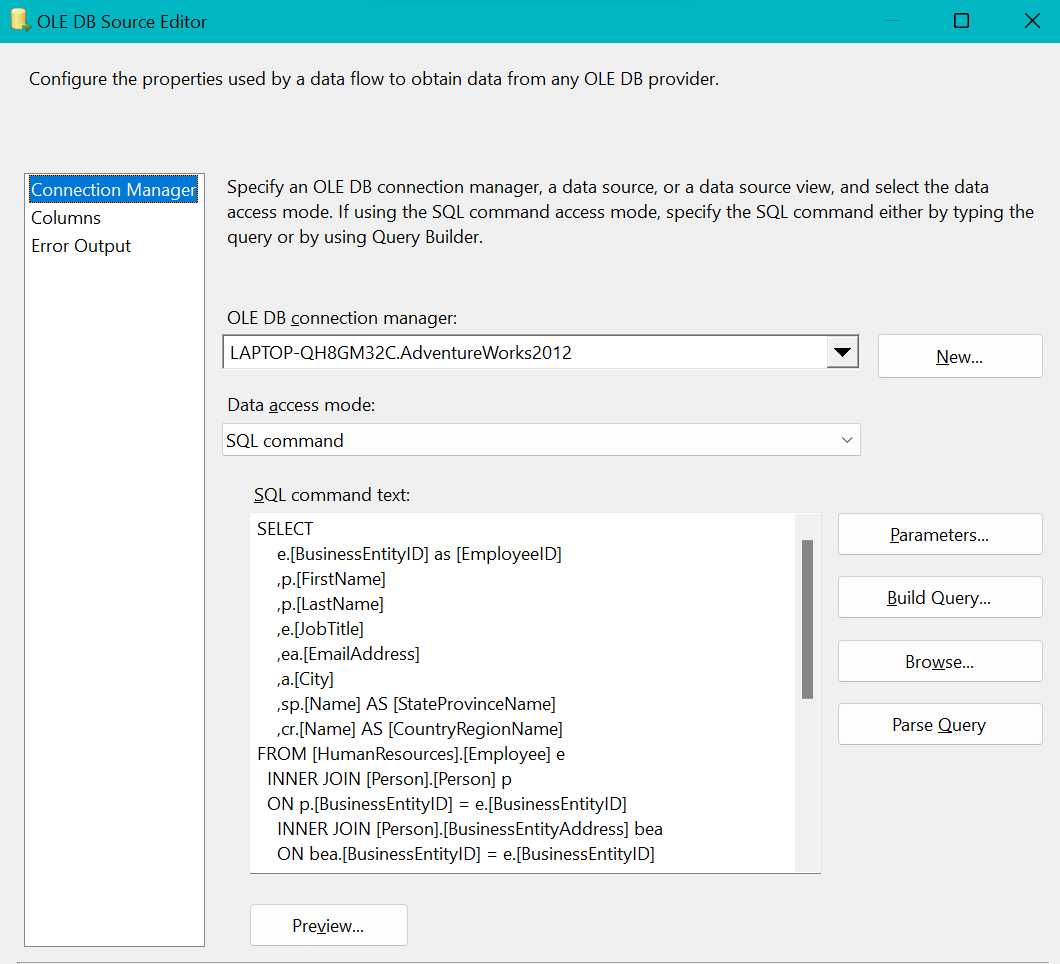
|  |  |  |
| --- | --- | --- |
| Column name | Type | Null? |
| EmailAddressId (key) | INT | NOT NULL |
| BusinessEntityID | INT | NOT NULL |
| EmailAddress | NVARCHAR(50) | NOT NULL |

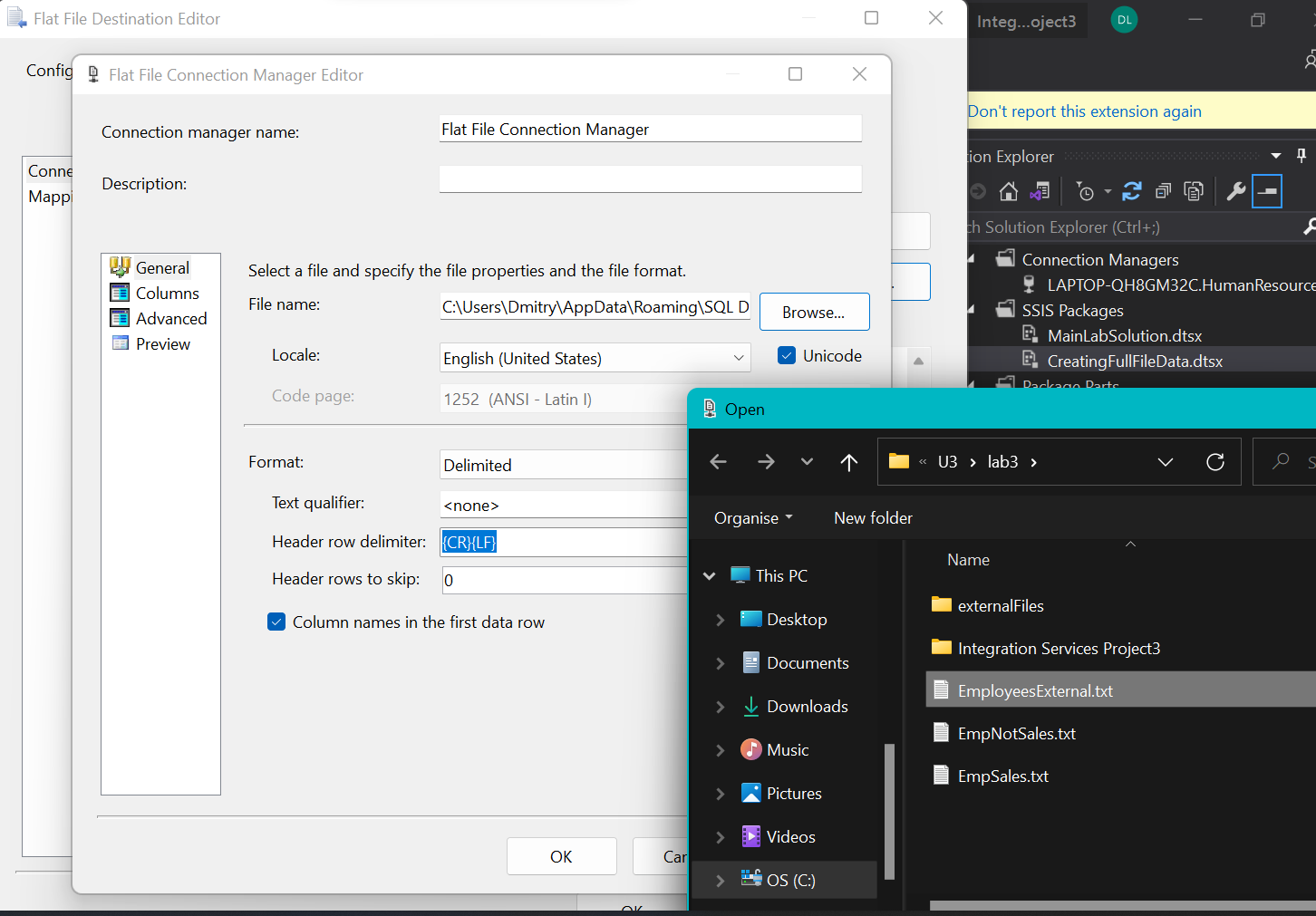
Create sql script to achieve the requirements above.

***Note.*** We need to make file with Employees called EmployeesExternal to fulfil it with comma – delimited data to later import it to our new created database HumanResources

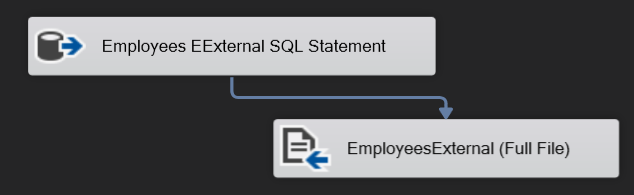


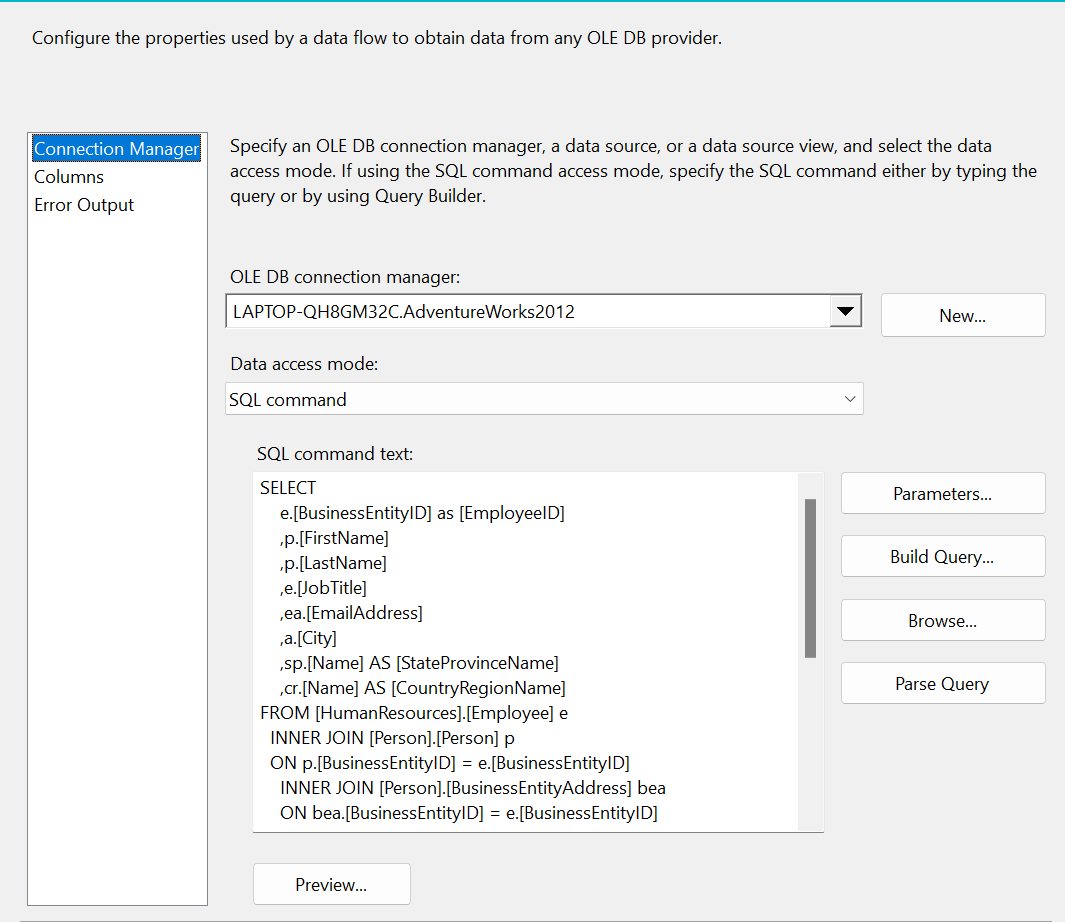






***Note.*** All import process will be be executed within one package but I separated a part of this lab in which I need to generate comma delimited file in txt format to later use it showing how I could specify package to import any amount of \*.txt files.





SELECT

e.[BusinessEntityID] as [EmployeeID]

,p.[FirstName]

,p.[LastName]

,e.[JobTitle]

,ea.[EmailAddress]

,a.[City]

,sp.[Name] AS [StateProvinceName]

,cr.[Name] AS [CountryRegionName]

FROM [HumanResources].[Employee] e

INNER JOIN [Person].[Person] p

ON p.[BusinessEntityID] = e.[BusinessEntityID]

INNER JOIN [Person].[BusinessEntityAddress] bea

ON bea.[BusinessEntityID] = e.[BusinessEntityID]

INNER JOIN [Person].[Address] a

ON a.[AddressID] = bea.[AddressID]

INNER JOIN [Person].[StateProvince] sp

ON sp.[StateProvinceID] = a.[StateProvinceID]

INNER JOIN [Person].[CountryRegion] cr

ON cr.[CountryRegionCode] = sp.[CountryRegionCode]

LEFT OUTER JOIN [Person].[PersonPhone] pp

ON pp.BusinessEntityID = p.[BusinessEntityID]

LEFT OUTER JOIN [Person].[PhoneNumberType] pnt

ON pp.[PhoneNumberTypeID] = pnt.[PhoneNumberTypeID]

LEFT OUTER JOIN [Person].[EmailAddress] ea

ON p.[BusinessEntityID] = ea.[BusinessEntityID]

order by p.[BusinessEntityID] asc;

***Note.*** I created database Human Resources with tables listed above. Script stores in file “create database and tables.sql”. Used Adventure Works database I joined tables to get following structures as shown above, but with data inside.

**dbo.EmployeesExternal:**

SELECT

e.[BusinessEntityID] as [EmployeeID]

,p.[FirstName]

,p.[LastName]

,e.[JobTitle]

,ea.[EmailAddress]

,a.[City]

,sp.[Name] AS [StateProvinceName]

,cr.[Name] AS [CountryRegionName]

FROM [HumanResources].[Employee] e

INNER JOIN [Person].[Person] p

ON p.[BusinessEntityID] = e.[BusinessEntityID]

INNER JOIN [Person].[BusinessEntityAddress] bea

ON bea.[BusinessEntityID] = e.[BusinessEntityID]

INNER JOIN [Person].[Address] a

ON a.[AddressID] = bea.[AddressID]

INNER JOIN [Person].[StateProvince] sp

ON sp.[StateProvinceID] = a.[StateProvinceID]

INNER JOIN [Person].[CountryRegion] cr

ON cr.[CountryRegionCode] = sp.[CountryRegionCode]

LEFT OUTER JOIN [Person].[PersonPhone] pp

ON pp.BusinessEntityID = p.[BusinessEntityID]

LEFT OUTER JOIN [Person].[PhoneNumberType] pnt

ON pp.[PhoneNumberTypeID] = pnt.[PhoneNumberTypeID]

LEFT OUTER JOIN [Person].[EmailAddress] ea

ON p.[BusinessEntityID] = ea.[BusinessEntityID]

order by p.[BusinessEntityID] asc;

**dbo.EmployeesAW:**

select p.[BusinessEntityID],[FirstName],[LastName],[JobTitle]

from [Person].[Person] p

inner join [HumanResources].[Employee] e

on p.BusinessEntityID = e.BusinessEntityID;

**dbo.EmailAddressesAW:**

select [EmailAddressID],p.[BusinessEntityID],[EmailAddress]

from [Person].[EmailAddress] e

inner join [Person].[Person] p

on e.[BusinessEntityID] = p.BusinessEntityID

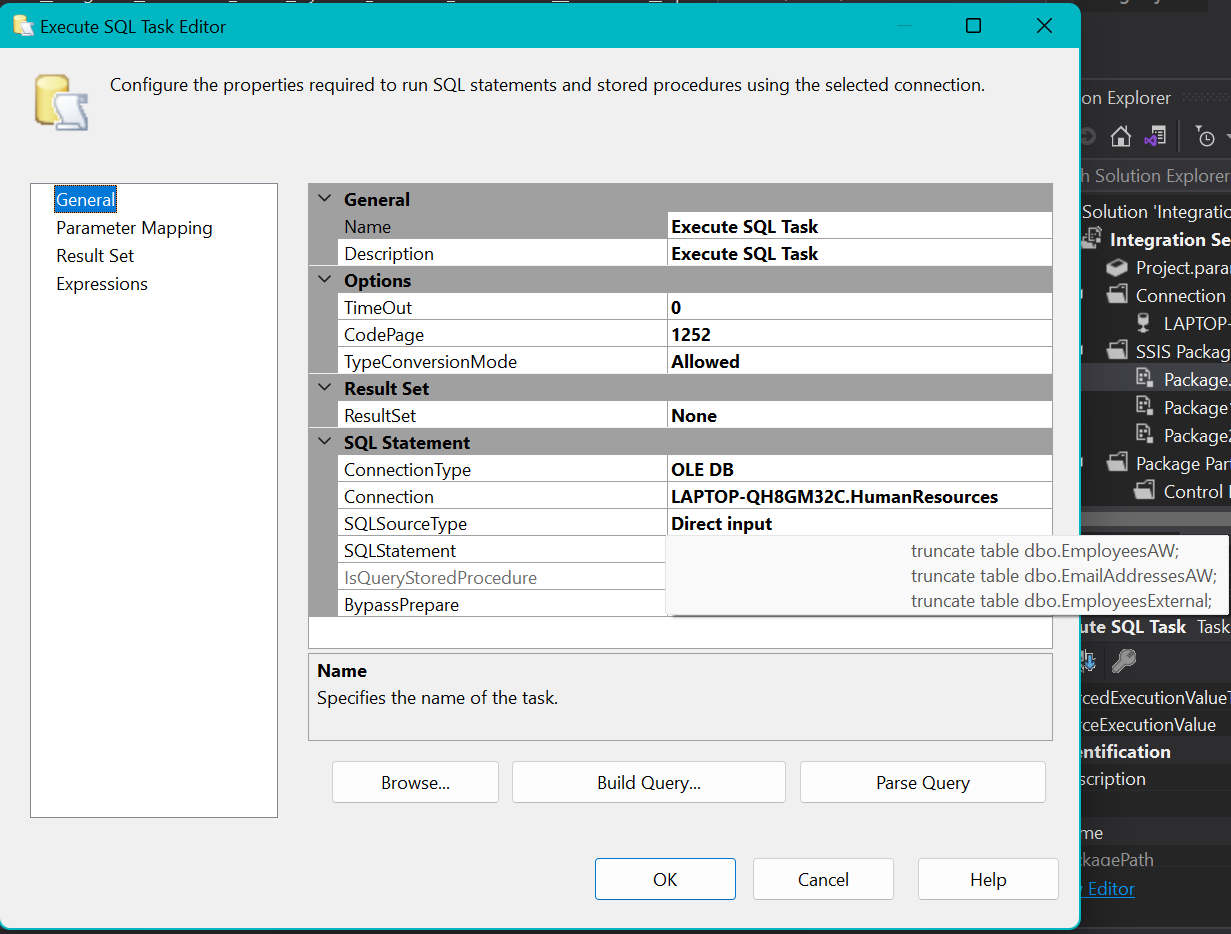
**Result:** SQL scripts that created needed objects.

# 2. Exercise 2: Creating ETL solution to import employees from different sources

You were given a few files that must be imported to the new database (see LabFiles). Your task is to create appropriate ETL solution. The following things should be taken into account:

* All import process must be executed within one package
* All files must be imported to the table Employees External and the solution should work if the numbers of files will be increased
* The import process can be executed more than once ( the destination tables are truncated every time )
* The table dbo.EmployeesAW must contain all employees from the database Adventure Works ( use tables HumanResources.Employee and Person.Person to get needed data)
* The table dbo.EmailAddressesAW must contain all emails of employees from the database Adventure Works (see table Person.EmailAddress)
* In case of any error during the process the tables does not contain partially loaded data (they must be empty)
* If the import process is successful then the package sends an email message with the subject «Employees have been imported successfully», otherwise the subject should be «Error loading employees».

***Note.*** So, now we have database consists of needed tables. We need to fulfil this tables with data using SSIS package (not to do this on DB level). Before that we need to truncate all this tables as it said in requirements to this Lab:

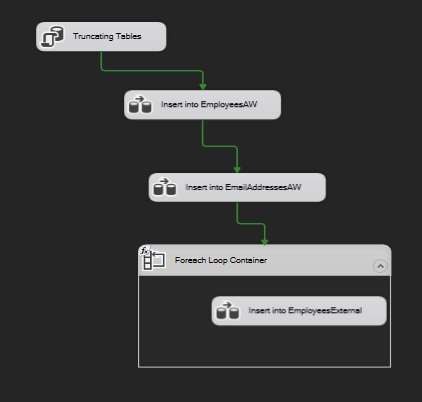


truncate table dbo.EmployeesAW;

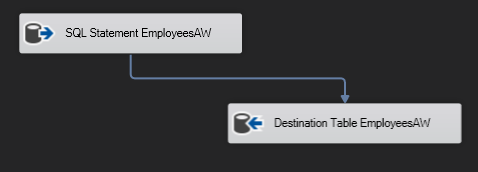
truncate table dbo.EmailAddressesAW;

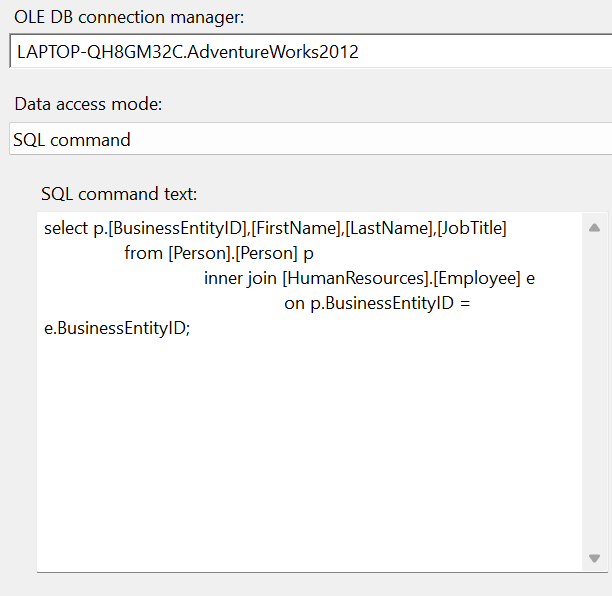
truncate table dbo.EmployeesExternal;

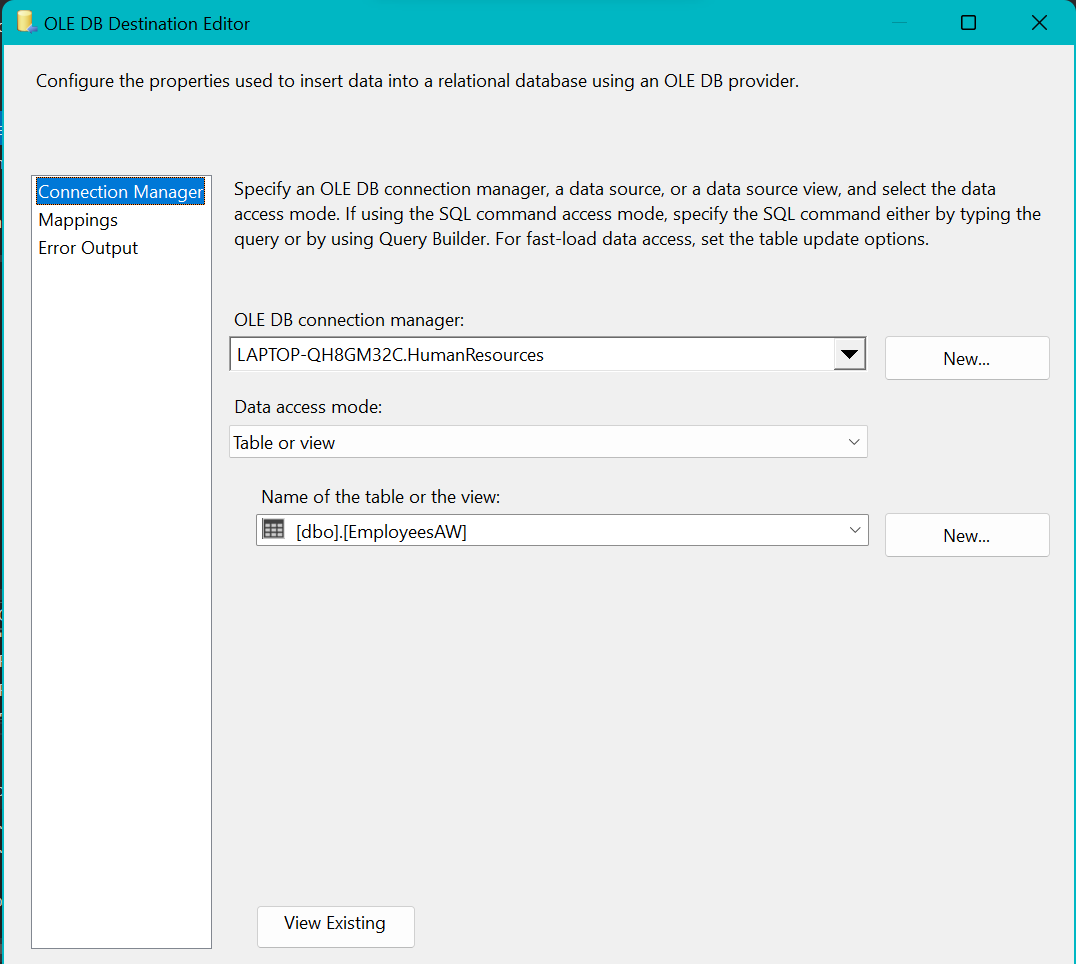
**Result:** SSIS solution.

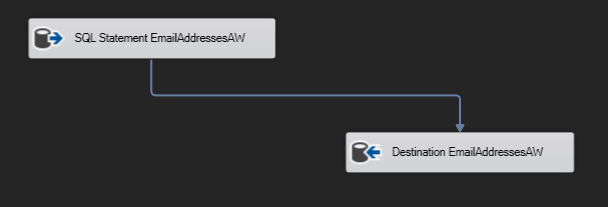


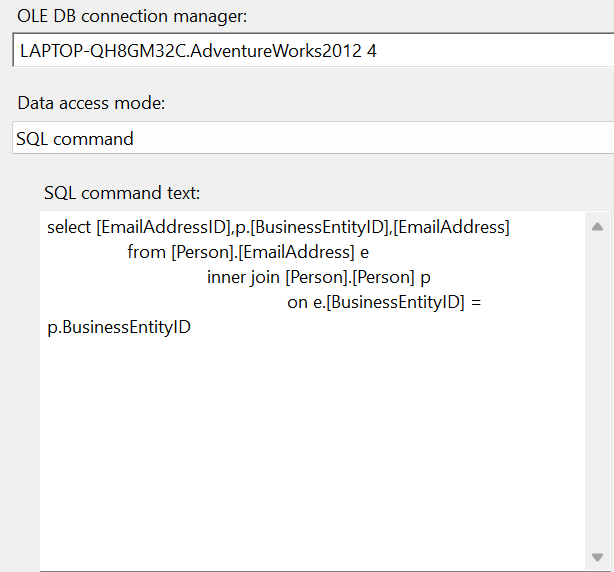
***Note.*** Let’s now see how we insert data into tables dbo.EmployeesAW and dbo.EmailAddressesAW.

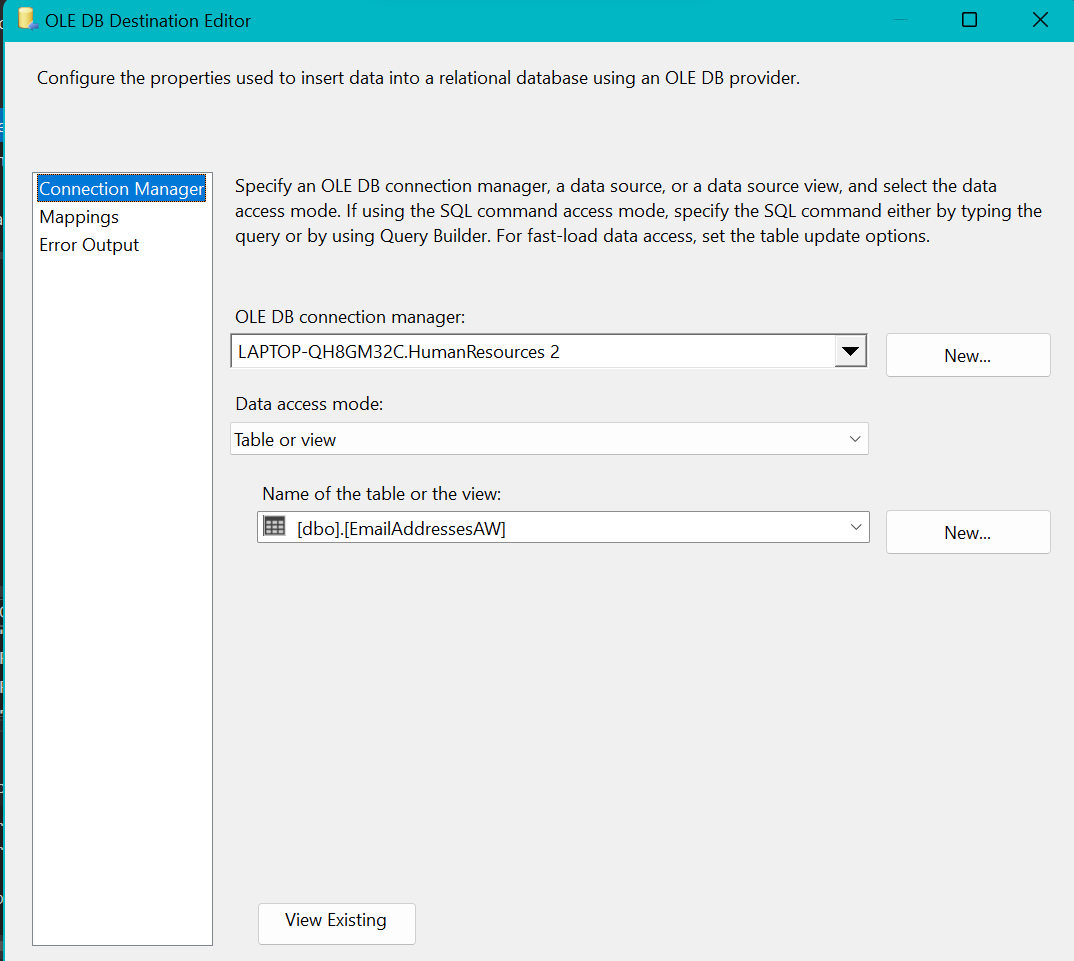




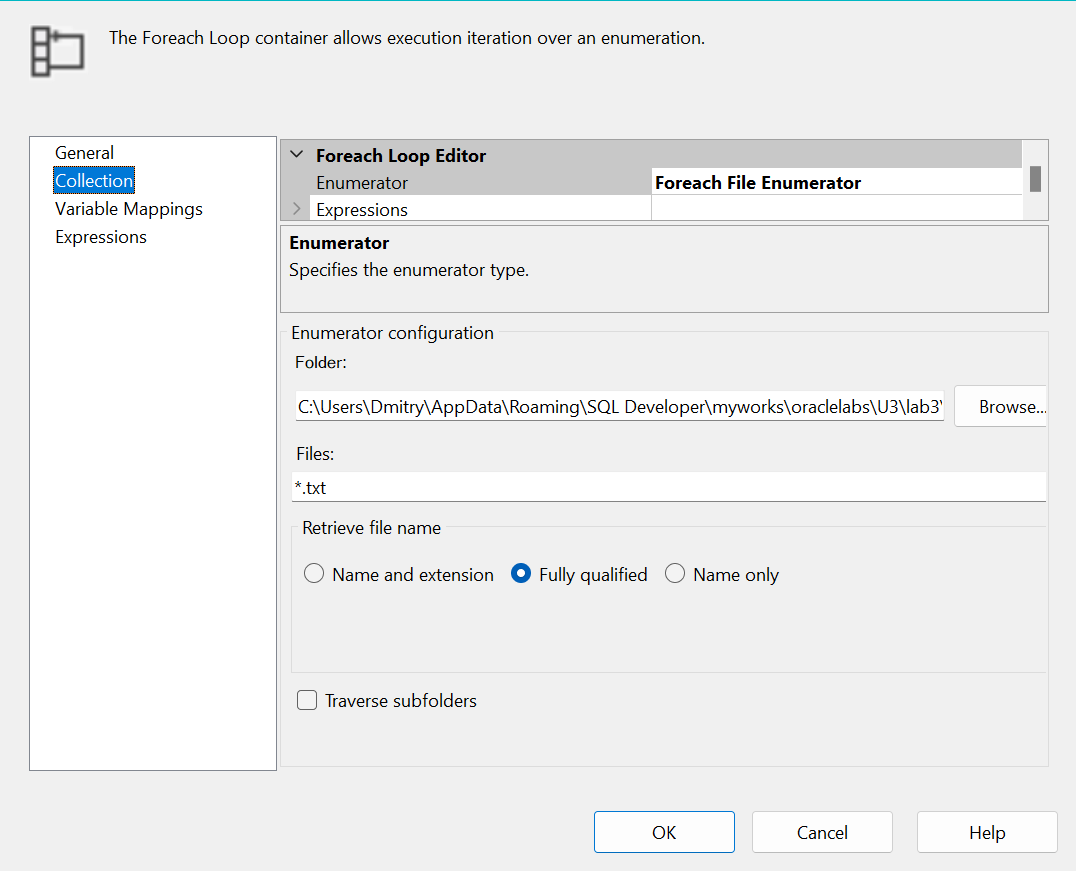


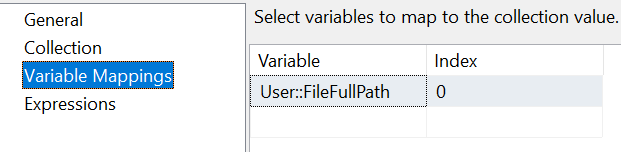


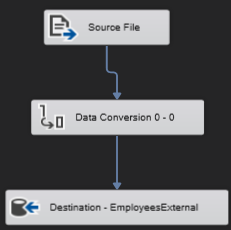




***Note.*** Now we have 2 tables with data. Let’s insert data into EmployeesExternal, but using 2 source files to satisfy this requirement. I am going to use For Each Loop container to loop every \*.txt file in source folder (so we can use any number of txt files).

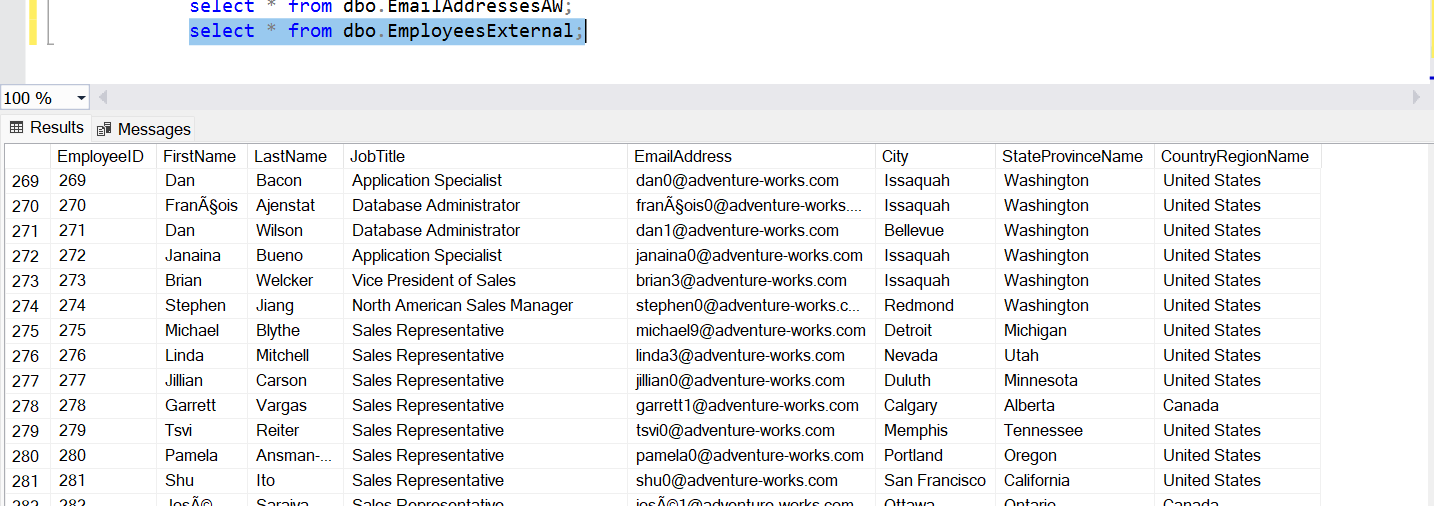


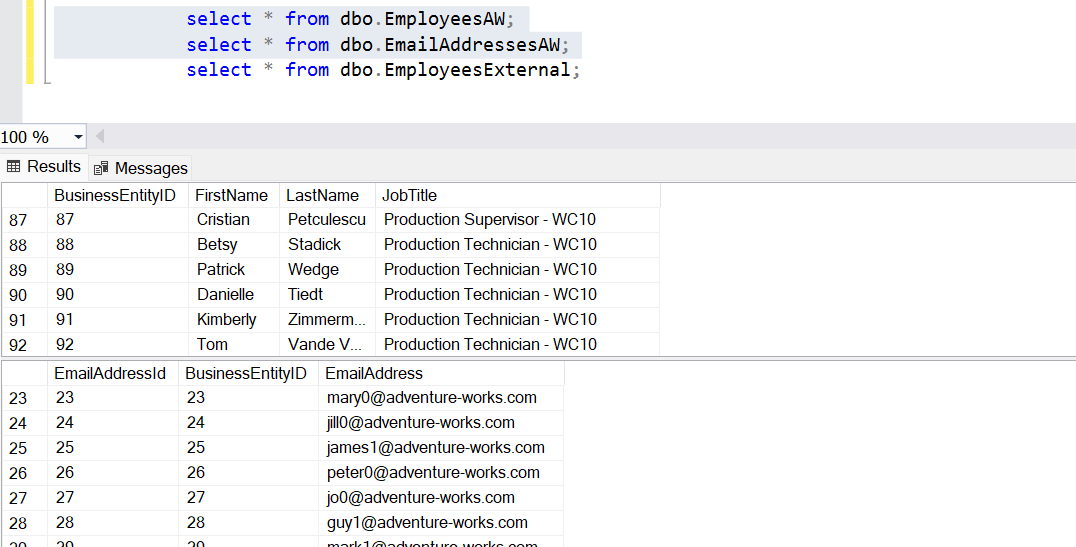


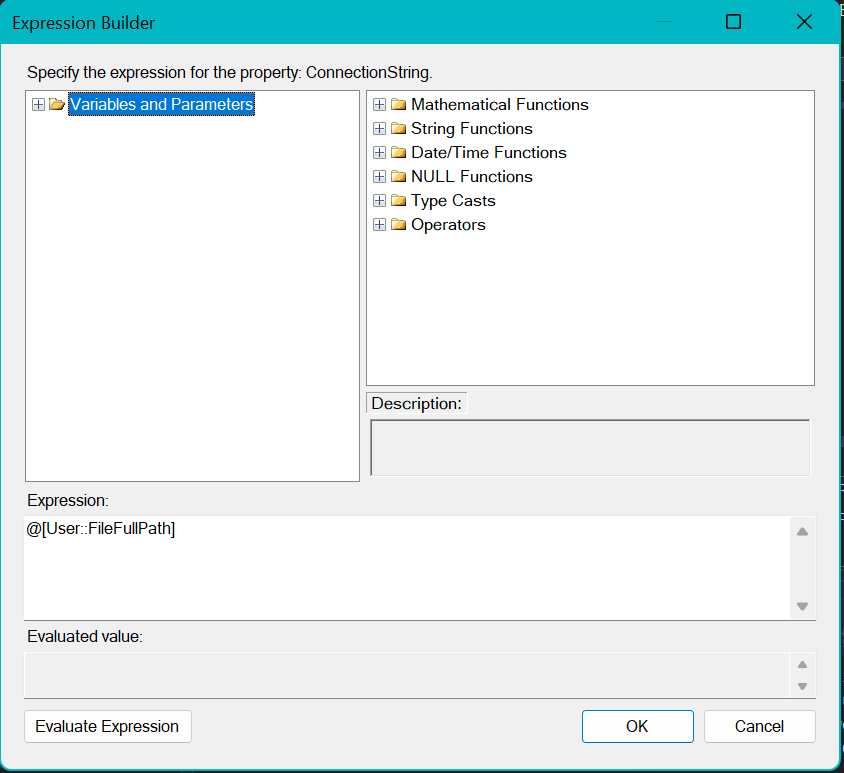


***Note.*** We can either try to specify encodings to load Unicode files but I chose to use Data Conversion, because we now can add any \*.txt files with any encodings, so this way we can simplify our interaction with files in future.

* All files must be imported to the table Employees External and the solution should work if the numbers of files will be increased

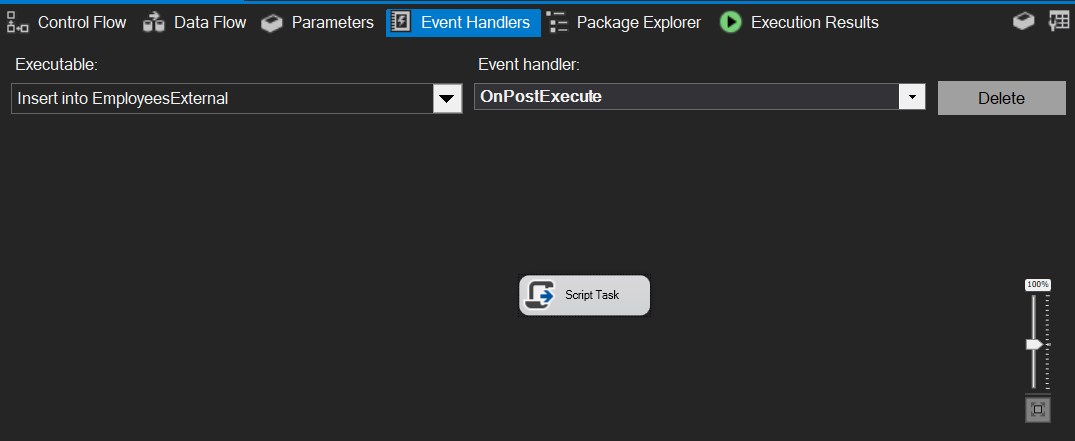


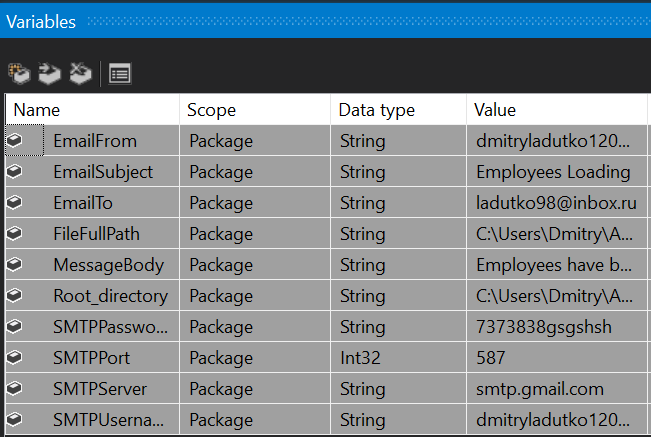


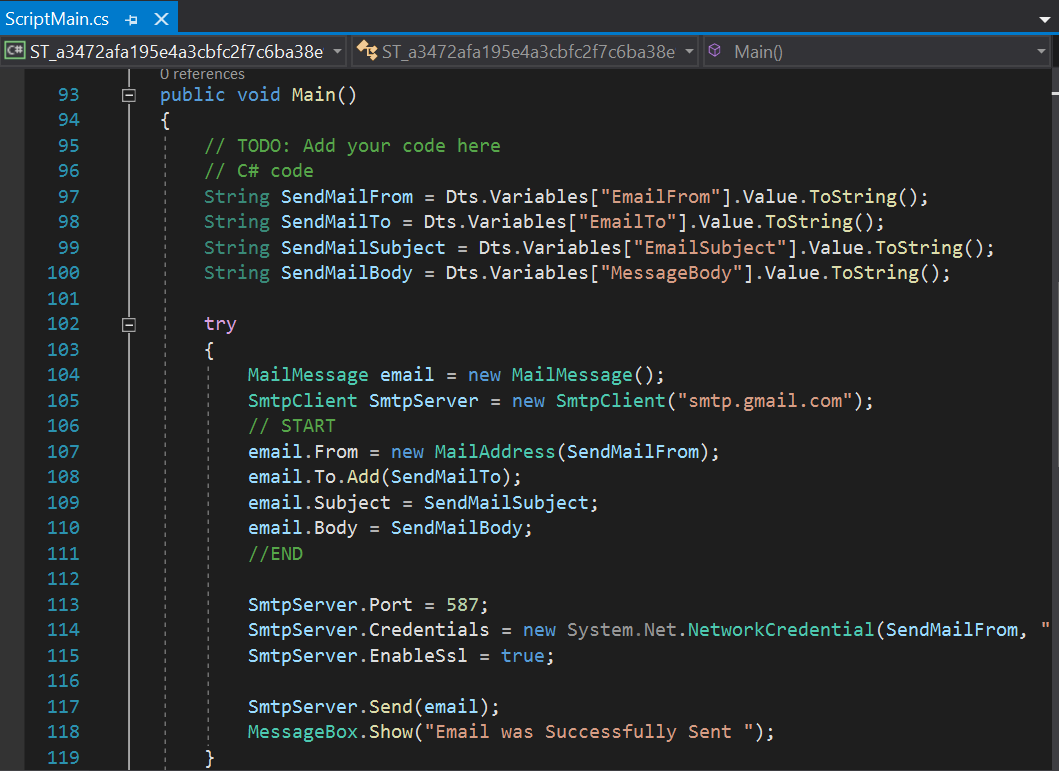


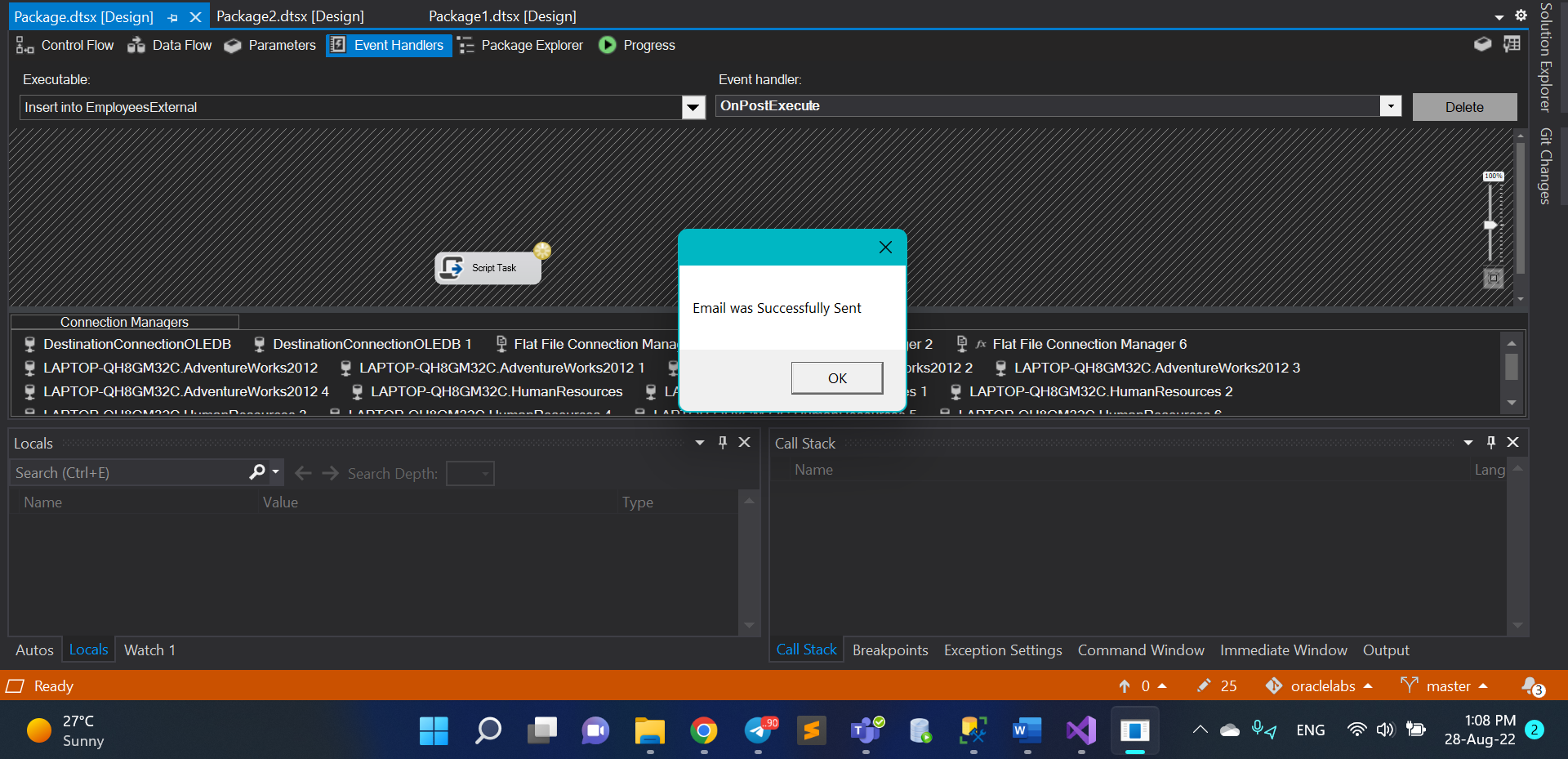
***Note.***  I have also specified an expression in connection used by flat file destination to fulfil it with next file its path by loop.

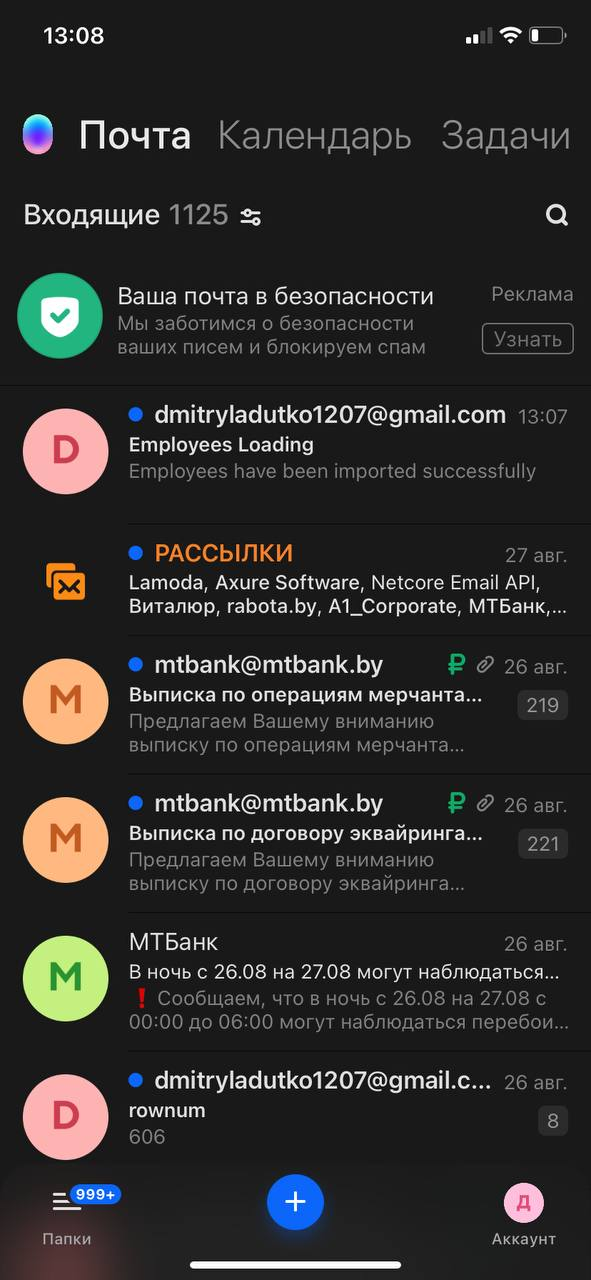
***Note.*** Now we need to get an email in case of successfully loaded data.





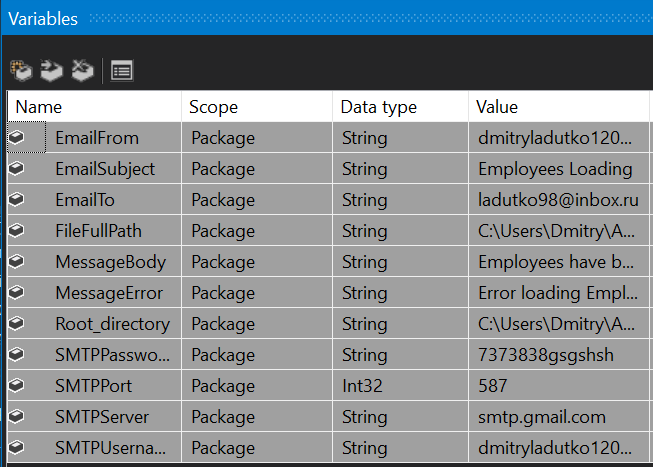




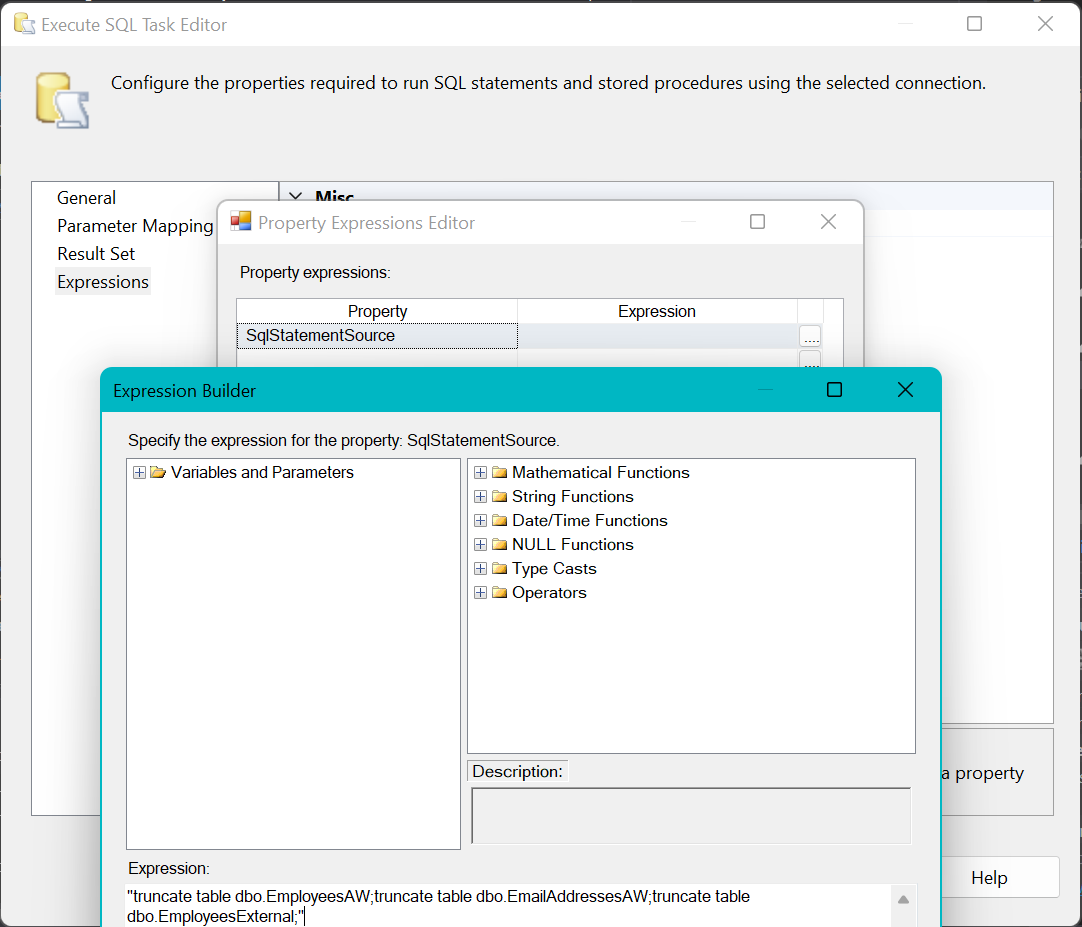


***Note.*** Now we also need to receive a mail in case of errors in loading employees

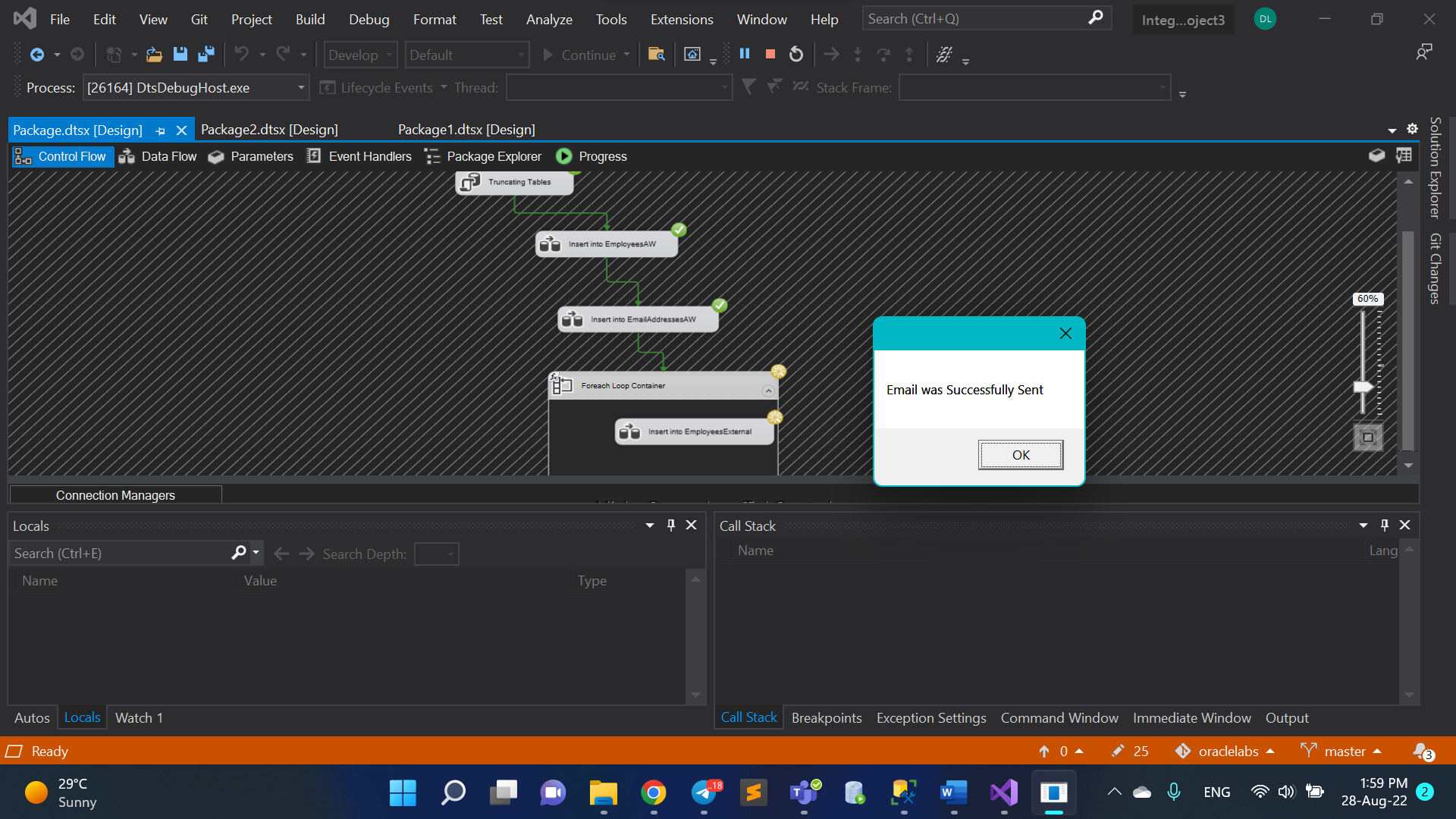
I added a variable MessageError for it. And used the same script to send an email but changed MessageBody to MessageError variable to send another text, also added one more event on error to send different messages depend on result (as it said in requirements).



***Note.*** The last requirement that I need to realize is that if there is (are) some error(-s) we do not have to insert partially data.



***Note.*** As you can see all of the requirements are satisfied! To check it one more time let’s execute the whole package once again (pay attention to the time):





# 3. Exercise 3: Creating ETL solution to export the data from HumanResourcesdb

You should generate two comma separated text files with names EmpSales.txt and EmpNotSales .txt based on employee data (dbo.EmployeesAW and dbo.EmailAddressesAW). The files must have the following structures:

EmpSales.txt:

FirstName, Last Name, EmailAddress

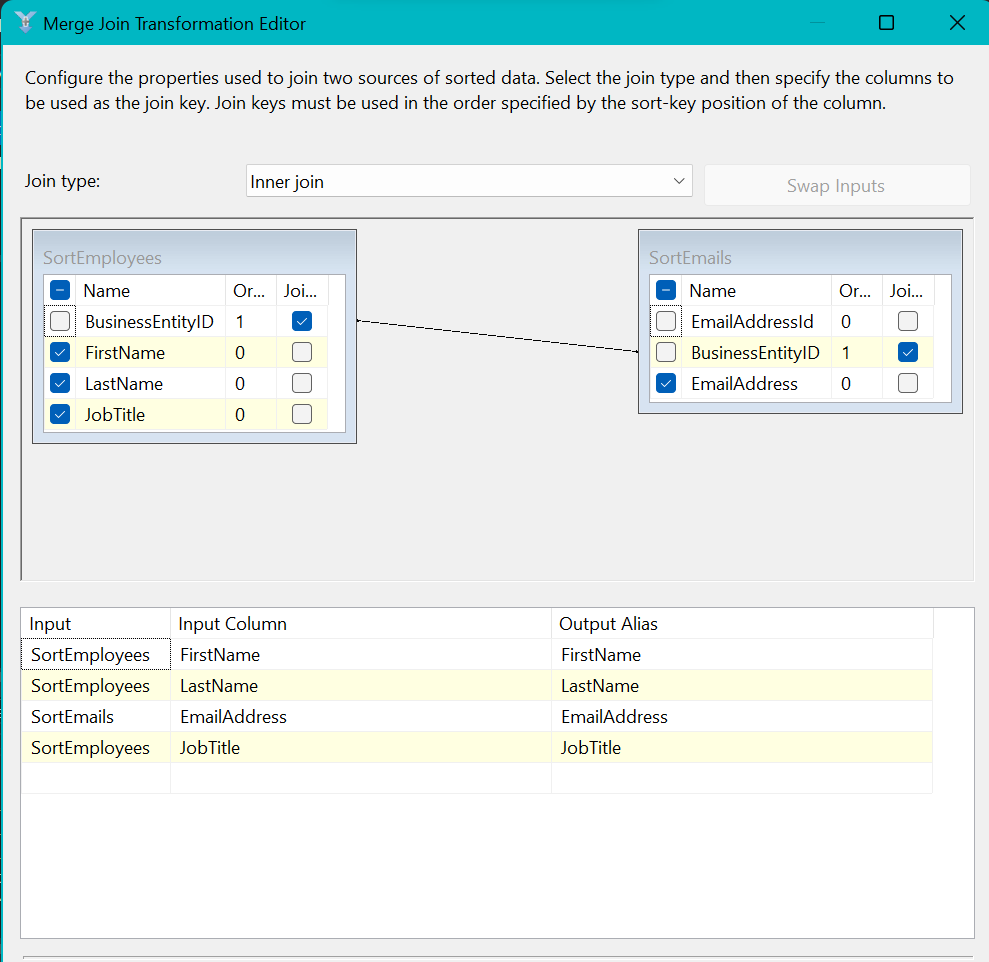
EmpNotSales.txt:

FirstName, LastName, EmailAddress

Employee e-mail addresses are stored separately from employee details, so your data flow must include a lookup to retrieve e-mail addresses based on the employee ID (using SQL Query to join the data is not allowed). The data in the files must be sorted by the employees' last names and then first names. The first text file should include only employees with JobTitle = “Sales Representative”. The second text file should include all other employees. The files should be generated within one data flow task.

**Result:** SSIS solution and the output files.

***Note.*** Now we need to create appropriate task to eventually get file output structure that is shown above. I used 2 sources (dbo.EmployeesAW and dbo.EmailAddressesAW) tables, then I used sorting to get BusinessEntityID from both tables so I can join them with merge join



***Note.*** Also we need to split it into 2 files with following structure:

EmpSales.txt:

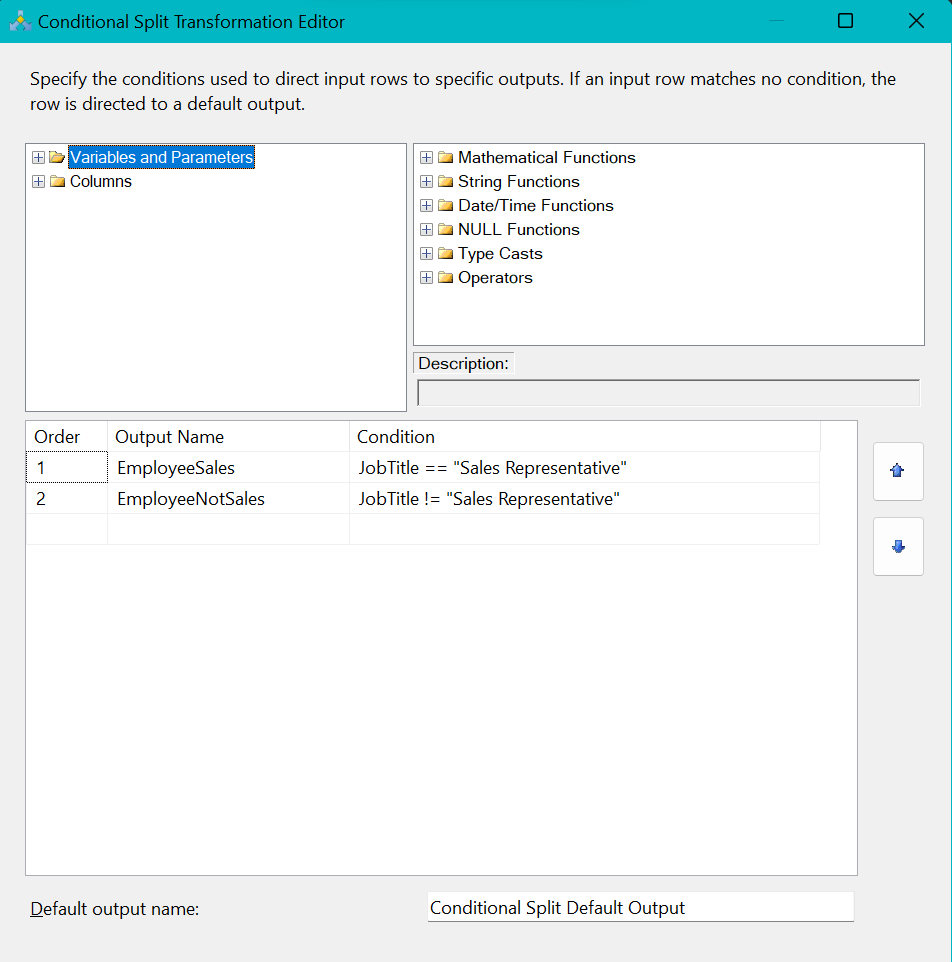
FirstName, Last Name, EmailAddress

EmpNotSales.txt:

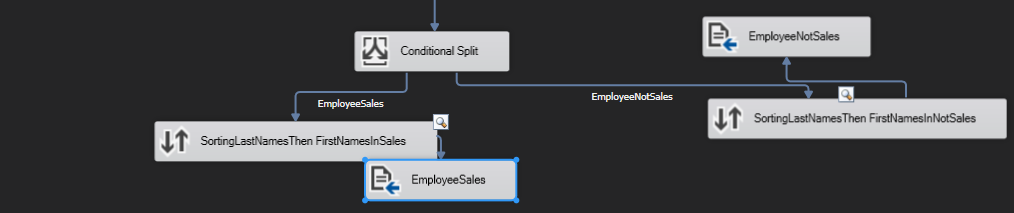
FirstName, LastName, EmailAddress

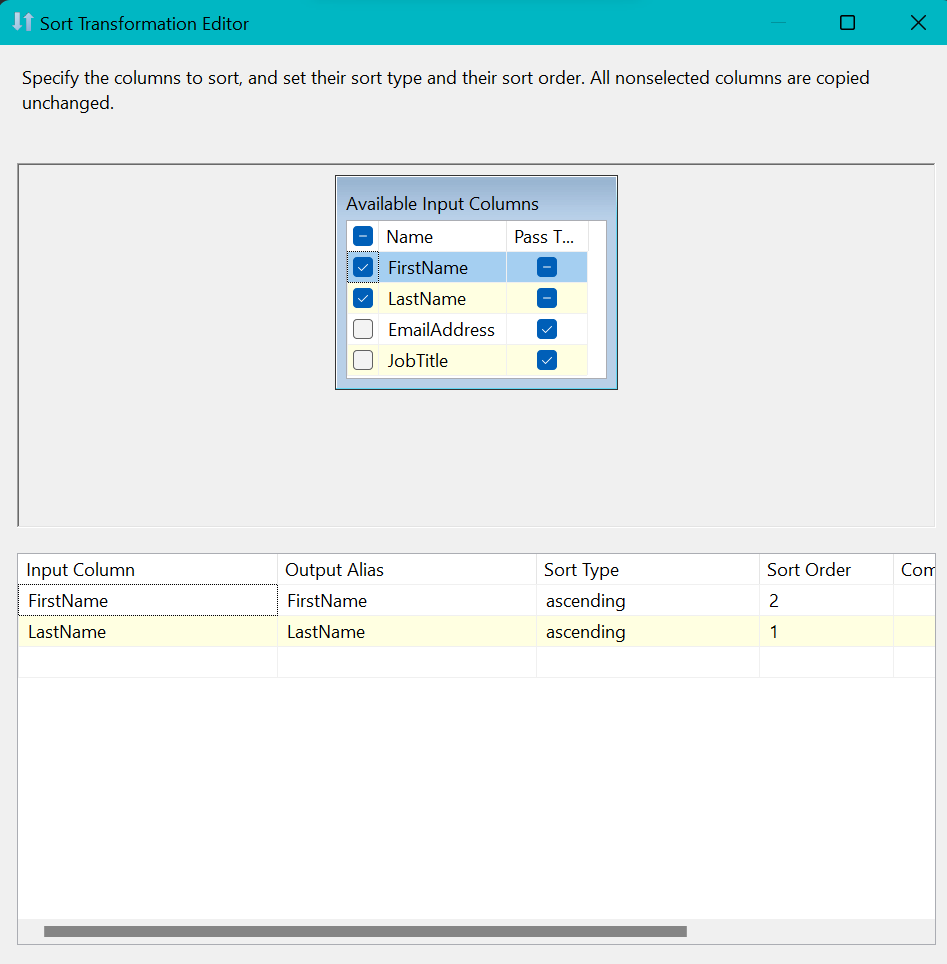


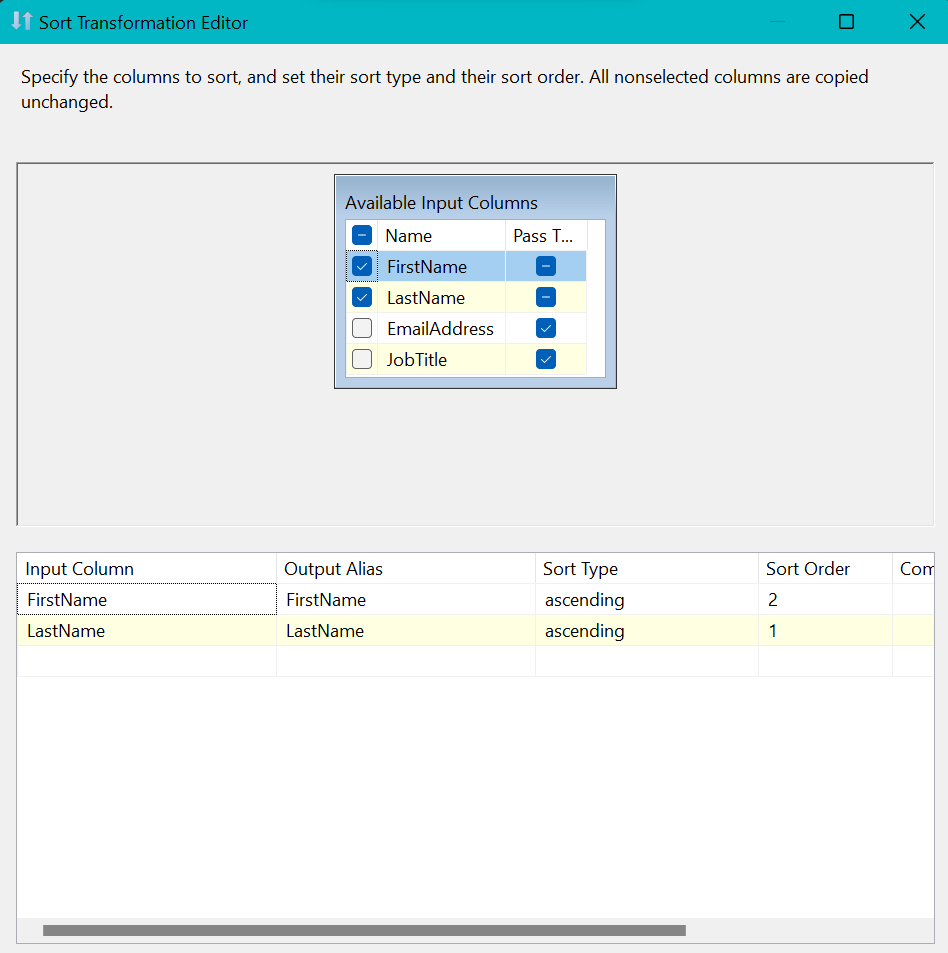
***Note.*** As it said in requirements, we need to divide employees by their JobTitle into 2 groups (Sales And NotSales). At the screenshot below you can see conditions I used to satisfy it.

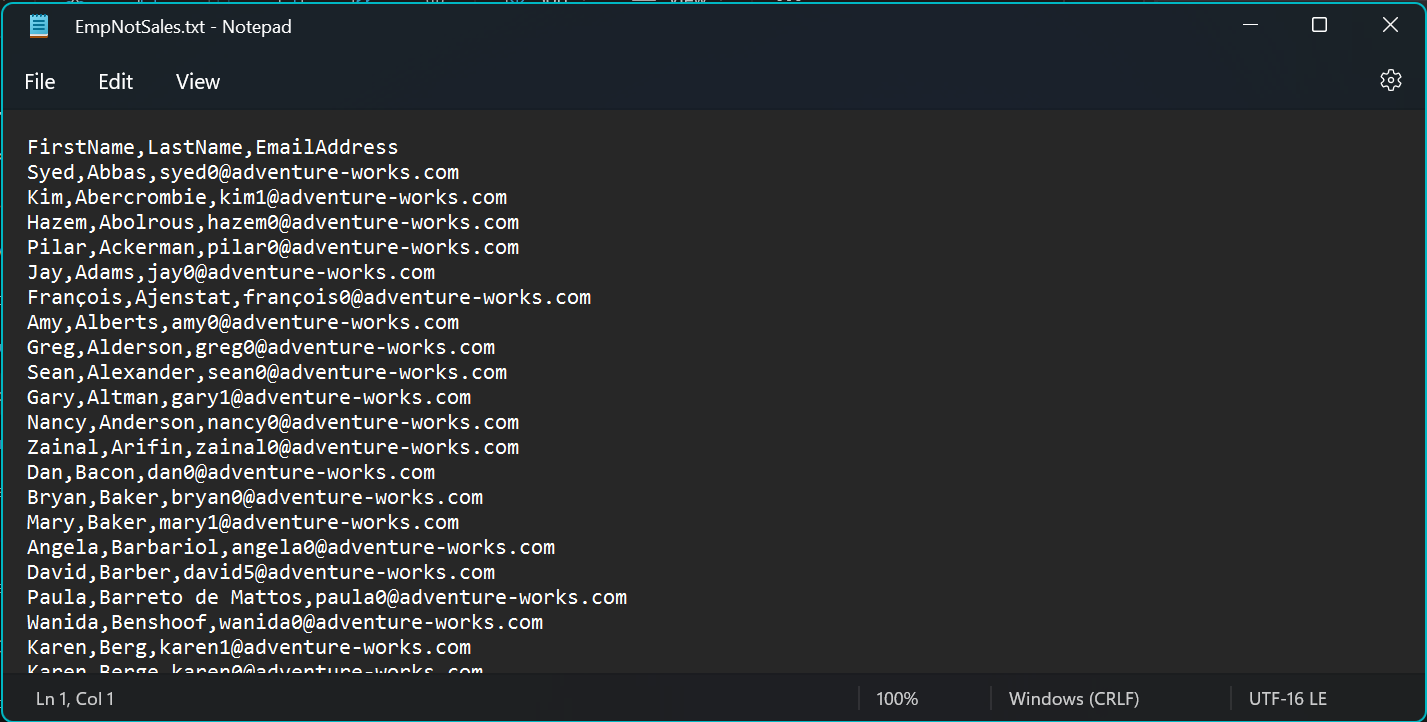


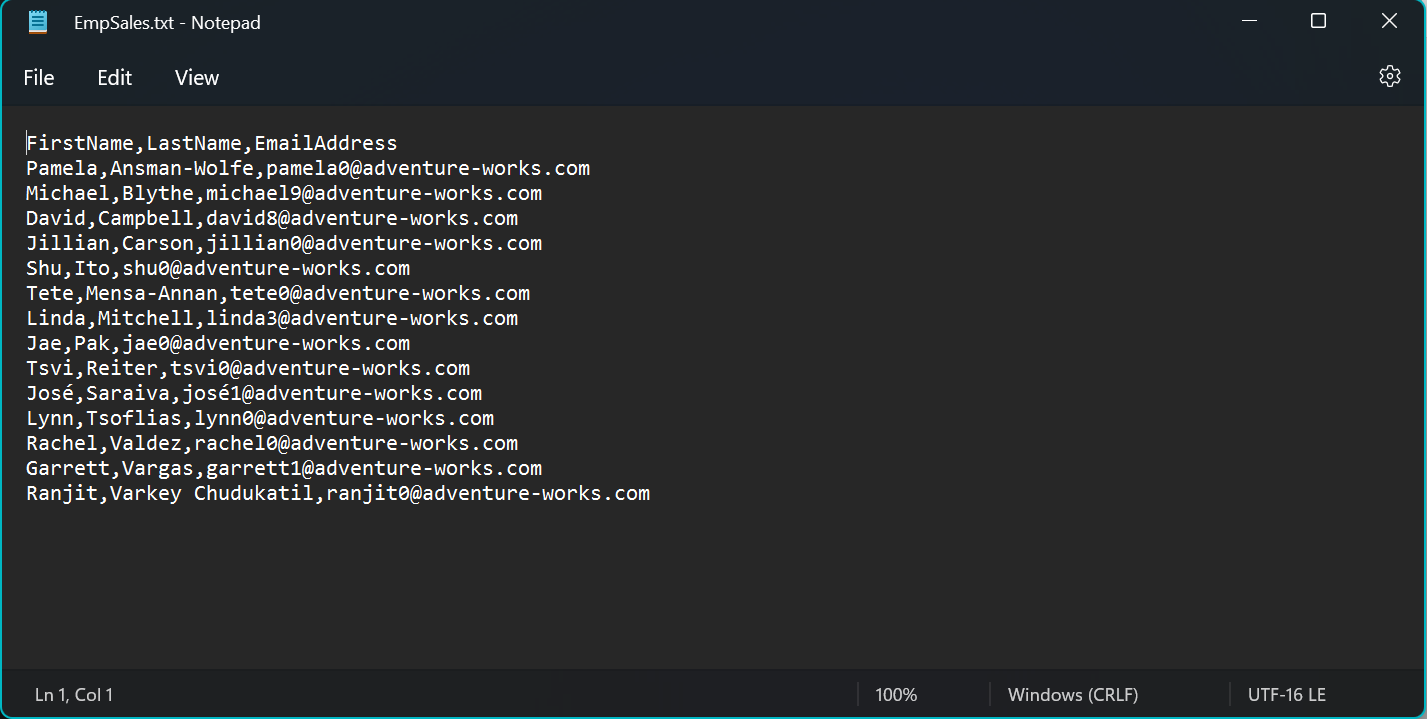
***Note.*** It also were given that I need to sort Employees by their Last Names and then their First Names in both groups. Thus I added sorting for them.

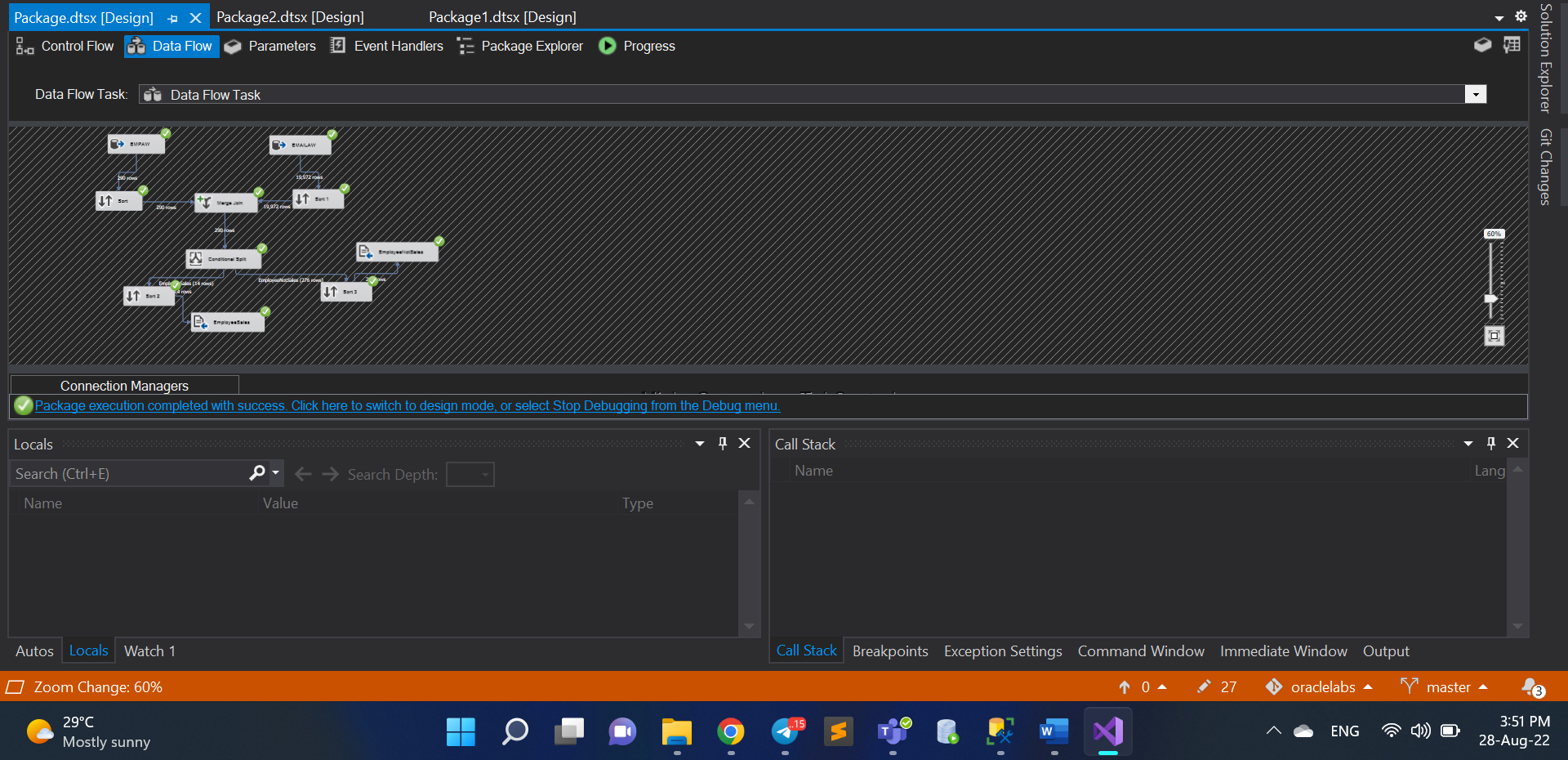












***Note.*** Now please pay attention to destination flat files. Firstly, last names are sorted and then, in group of current character first names are sorted.

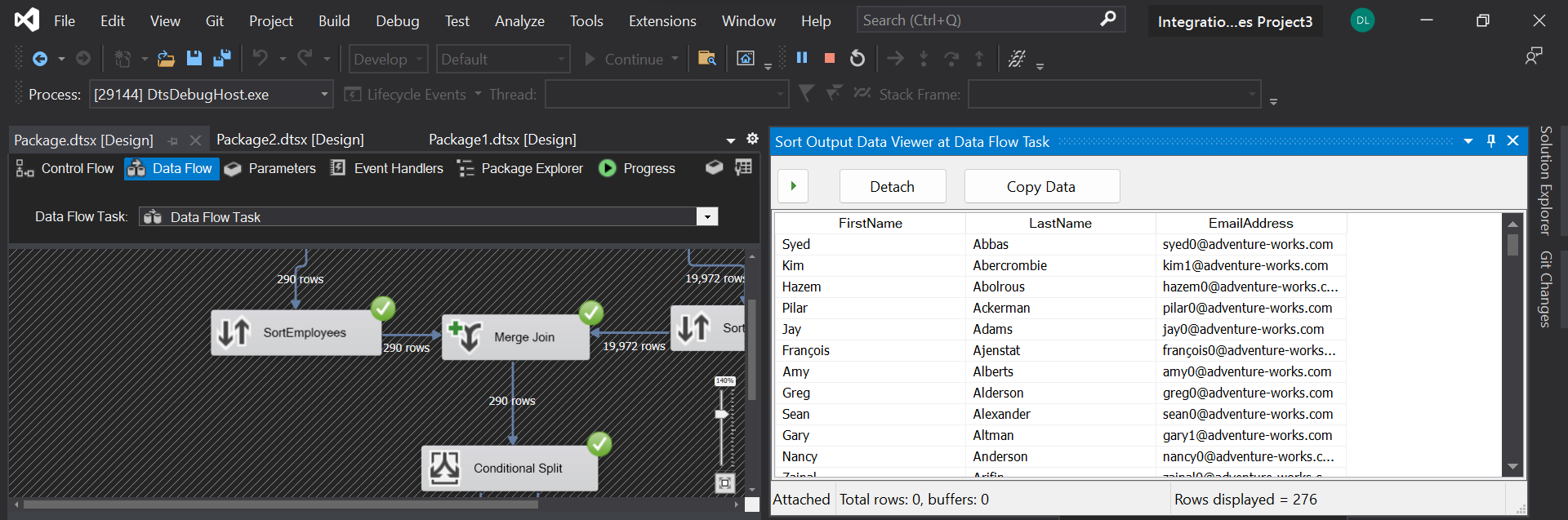
All the requirements are satisfied, package executes correctly.

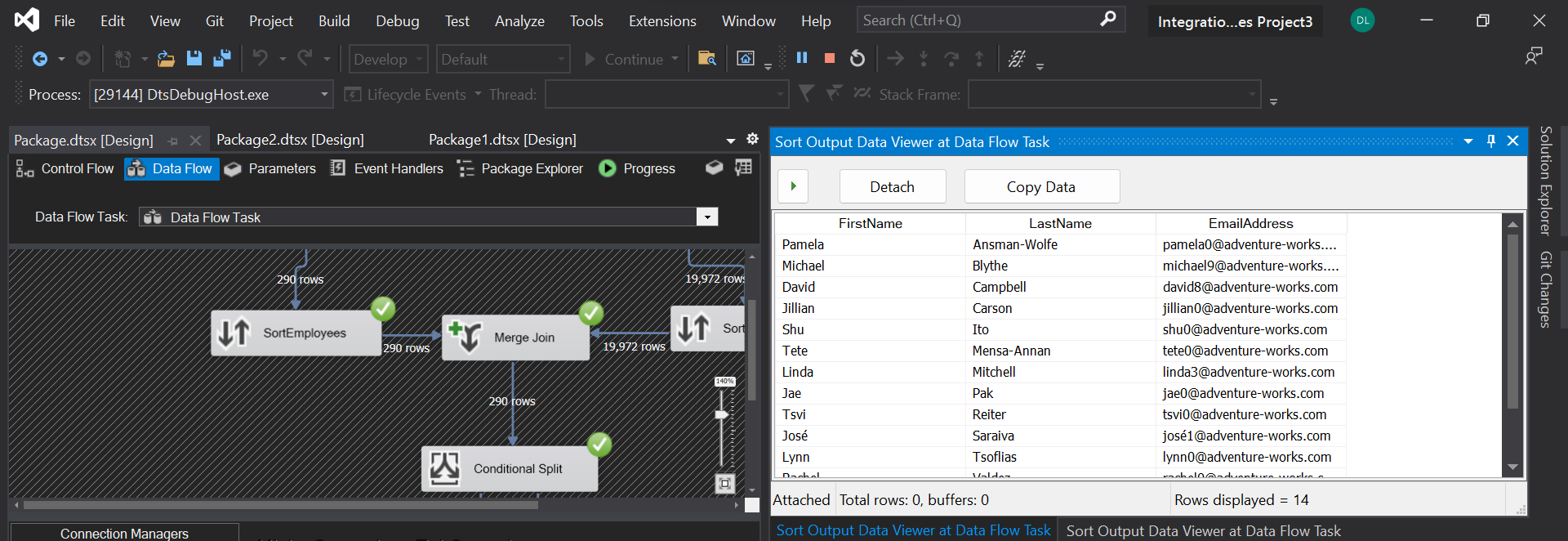
# 4. Exercise 4: Using Data Viewers

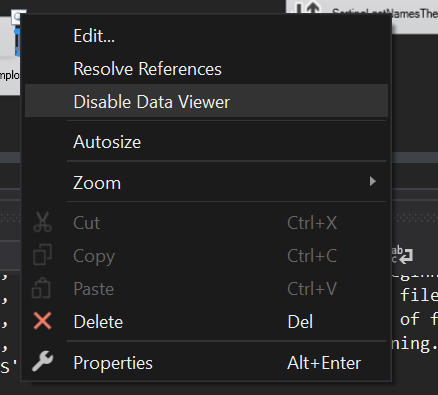
Now that you have created your data flows, you need to verify and test their functionality using data viewers.

This exercise's main tasks are:

1. Add a grid data viewer.
2. Run the package, and view the data viewer.
3. Remove the data viewer.





****

***Note.*** In fact we can check files manually after package has been executed but it is much more convenient to use Grid Data Viewer to see the data.

**Result:** Screenshots of the viewers

**Laboratory Work Summary**

**At this Laboratory Work** we more practiced about building package **solutions in SSIS.** Now we know how we could work with different objects to make a good **data flow.** We used lot’s of instruments to **satisfy listed requirements**. Practiced more how we can get useful **notifications about data loadings**, used **data converter**s to create **multifunctional loading schemas.**

**We** also **were introduces** to the **data viewer, merge join and conditional split functionality.**

**P.S.** The solution is executable from **one package**. I only putted the part with creating external file into another package to logically separate them.