

# CALL CENTER DATA MANGEMENT

ETL Project

## Group members:

- Horlain Nemkenang
- Lucas Favero
- Bhargavi Mummina
- Hugo Beffeyte
- Massop Aurelie

## I. Table of contents

I.	Introduction.....	2
II.	Data Presentation.....	2
III.	Data Model.....	3
IV.	Pipeline design .....	4
V.	Staging database .....	4
1.	Employee Table .....	4
2.	US States Table.....	6
3.	Call Types Table .....	7
4.	Call Charges .....	9
5.	Call Data Table.....	10
VI.	ODS Database.....	15
1.	ODS - Employee Table .....	15
2.	ODS - Call Charges Table .....	19
3.	ODS - Call Data Table.....	23
VII.	DWH Database .....	28
4.	DimEmployee .....	28
5.	DimCallCharges.....	31
6.	Fact Table Call Data .....	33

## II. Introduction

To be able to use and extract value from available data, it first needs to be integrated into an IT system. This implies that all the data coming from various sources needed to be unified and standardized to be used by other programs. One way to achieve this is to implement a Data Warehouse.

In this project, we are going to design and implement a Data Warehouse with data from a call center named ServiceSpot. For this project we will use SQL Server and SSIS.

## III. Data Presentation

Our data is composed of seven Excel files .

The first file "Employees".csv" has data of employees from ServiceSpot company. The data is arranged as follow :

- Each column represent an information about an employee, we have "EmployeeID","EmployeeName","Site","ManagerName",
- Each line represent an Employee,

The second file "Us States.csv" represents data about the Us State. The data is arranged as follow :

- Each column represent an Information about a Us State, we have "StateCD","Name","Region",
- Each line represent a Us State,

The third file " Call types.csv" represents data about the different call types of the call center "ServiceSpot". The data is arranged as follow :

- "CallTypeID" ID of the different call
- "CallTypeName" The name of call

The fourth file "Call charges.csv" represents data about the charges of each call through yeah. The data is arranged as follow :

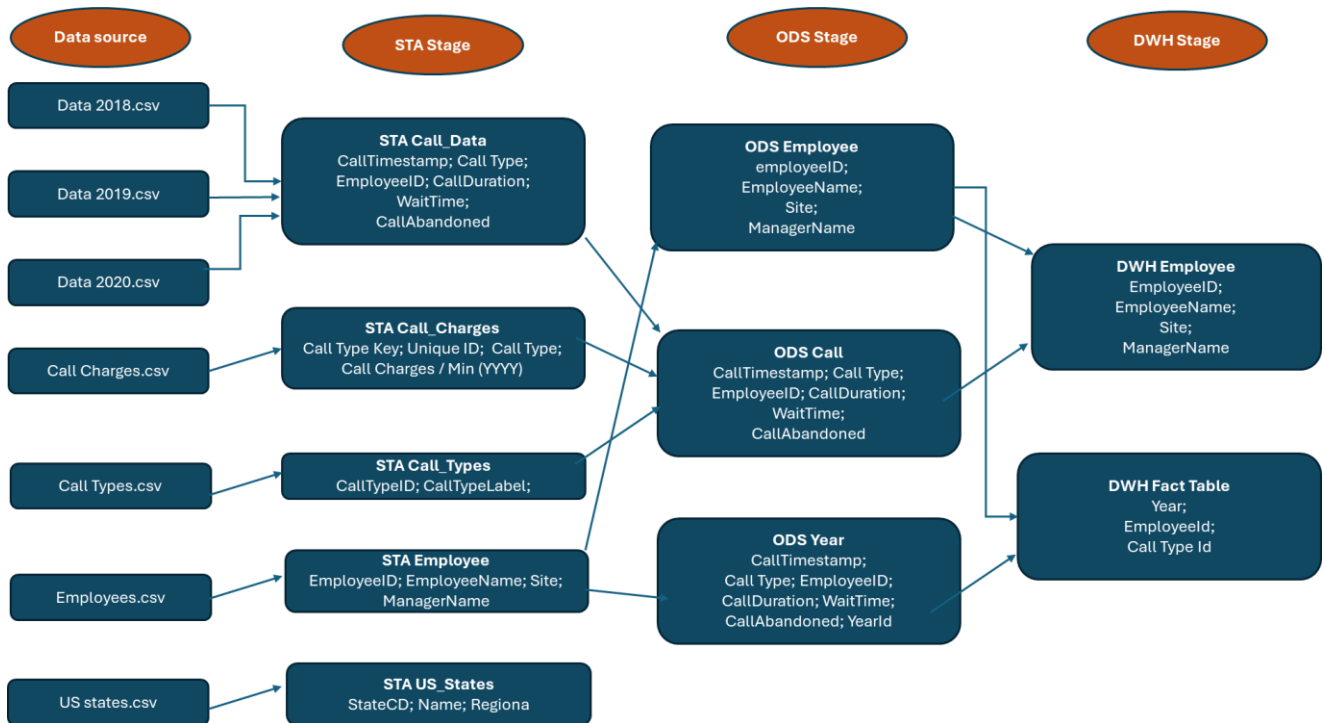
- Each column represent call charges for each year (2018,2019,2020,2021)
- Each line represent a call types "Sales","Biling","TechSupport",

The next three files are in a folder named "Calls Data", we find three files called "Data 2018.csv", "Data 2019.csv", "Data 2020.csv" each one representing data from one year. The data is arranged as follow :

- "CallTimeStamp" Date about a call, year,month,day,hours,minute,second
- "CallType" which type of call
- "EmployeeID" which employee receive the call

- "CallDuration" How much second the call last
- "WaitTime" How much time the client wait before an employee answer
- "CallAbandoned" If the call have been answer or no
- Each line represent a call

## IV. Data Model



## V. Pipeline Design

The pipeline will be done in three steps:

- The Staging area (STA) will allow to load the data as is, or with minimal changes.
- The Operational Data Store area (ODS) will allow to clean and standardize the data. If the data don't pass the quality criteriums, they will be put in the "Technical\_Rejects" table as technical rejects.
- The Data WareHouse area (DWH) will organize the data in one fact table related to multiple dimensions tables. If records can't be integrated in the schema, they will be put in the "Functional\_Rejects" table as functional reject. Alternatively, some placeholder relations can be created.

There will be one STA and ODS package per file.

## VI. Staging database

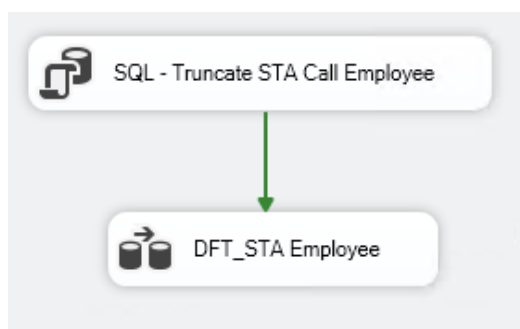
Here, the role of the staging database is to store all the data coming from the different sources. We want to access all available data.

### 1. Employee Table

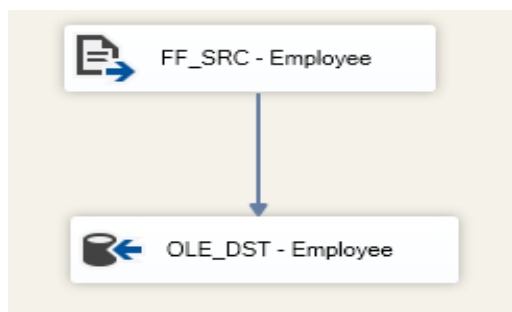
Here is an extract from the file "Employee.csv"

EmployeeID	EmployeeName	Site	ManagerName
N772493	Onita Trojan	Spokane, WA	Deidre Robbs
F533051	Stormy Seller	Aurora, CO	Elsie Taplin
S564705	Mable Ayoub	Aurora, CO	Shala Lion
I281837	Latrisha Buckalew	Aurora, CO	Rana Taub
Y193775	Adrianna Duque	Spokane, WA	Collin Trotman
J632516	Keiko Daulton	Spokane, WA	Jamar Prael
G727038	Dolores Lundeen	Aurora, CO	Shala Lion
V126561	Wilbur Mohl	Jacksonville, FL	Casey Bainbridge
E243130	Ileen Bornstein	Jacksonville, FL	Gonzalo Lesage

For this table, we don't need to add additional data. Here we just make sure to truncate the table "Employee" before running the package:



The dataflow is an import of a flat file :



We need to create the destination table with the following command:

```

CREATE TABLE [OLE_DST - Employee] (
    [EmployeeID] varchar(50),
    [EmployeeName] varchar(50),
    [Site] varchar(50),
    [ManagerName] varchar(50)
)
  
```

The data is then loaded into the table “Employee”:

OLE DB connection manager:

localhost.A\_23\_ETL\_GroupProject\_STA

Data access mode:

Table or view

Name of the table or the view:

[dbo].[OLE\_DST - Employee]

We also make sure to change some of the names to avoid problems with accents, spaces and SQL reserved words:

Available Input Columns		Available Destination Columns	
Name		Name	
EmployeeID		EmployeeID	
EmployeeName		EmployeeName	
Site		Site	
ManagerName		ManagerName	

Input Column	Destination Column
EmployeeID	EmployeeID
EmployeeName	EmployeeName
Site	Site
ManagerName	ManagerName

Here is the first ten lines of the results:

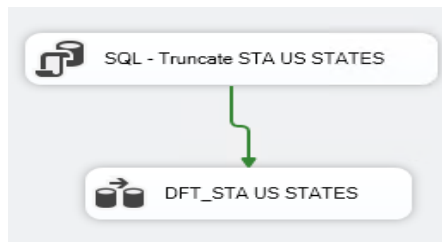
	EmployeeID	EmployeeName	Site	ManagerName
1	N772493	Onita Trojan	Spokane, WA	Deidre Robbs
2	F533051	Stormy Seller	Aurora, CO	Elsie Taplin
3	S564705	Mable Ayoub	Aurora, CO	Shala Lion
4	I281837	Latrisha Buckalew	Aurora, CO	Rana Taub
5	Y193775	Adrianna Duque	Spokane, WA	Collin Trotman
6	J632516	Keiko Daulton	Spokane, WA	Jamar Prah
7	G727038	Dolores Lundeen	Aurora, CO	Shala Lion
8	V126561	Wilbur Mohl	Jacksonville, FL	Casey Bainbridge
9	E243130	Ileen Bomstein	Jacksonville, FL	Gonzalo Lesage
10	C206355	Janeth Roesler	Spokane, WA	Miyoko Degraw

## 2. US States Table

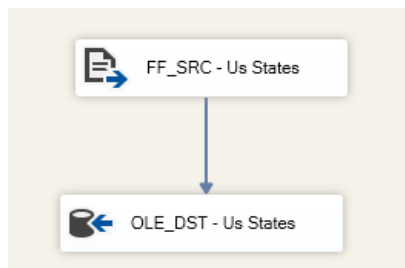
Here is an extract from the file “US STATES.csv”

	StateCD	Name	Region
1			
2	AK	Alaska	West
3	AL	Alabama	South
4	AR	Arkansas	South
5	AZ	Arizona	West
6	CA	California	West
7	CO	Colorado	West
8	CT	Connectic	Northeast
9	DC	District of	South
10	DE	Delaware	South

For this table, we don't need to add additional data. Here we just make sure to truncate the table “US STATES” before running the package:



The dataflow is an import of a flat file:



We need to create the destination table with the following command:

```
CREATE TABLE [OLE_DST - Us States] (  
    [StateCD] varchar(50),  
    [Name] varchar(50),  
    [Region] varchar(50)  
);
```

The data is then loaded into the table “US States”:

OLE DB connection manager:

localhost.A\_23\_ETL\_GroupProject\_STA

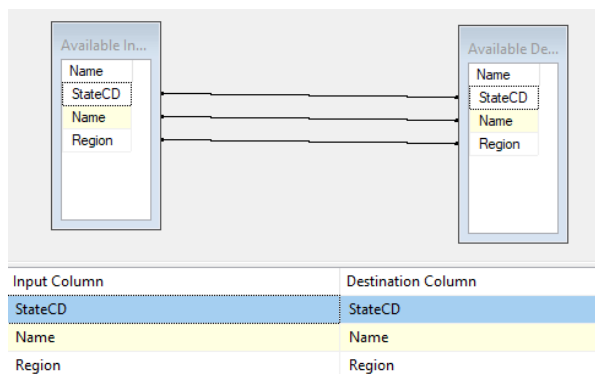
Data access mode:

Table or view

Name of the table or the view:

[dbo].[OLE\_DST - Us States]

We also make sure to change some of the names to avoid problems with accents, spaces and SQL reserved words:



Here is the first ten lines of the results:

	StateCD	Name	Region
1	AK	Alaska	West
2	AL	Alabama	South
3	AR	Arkansas	South
4	AZ	Arizona	West
5	CA	California	West
6	CO	Colorado	West
7	CT	Connecticut	Northeast
8	DC	District of Columbia	South
9	DE	Delaware	South
10	FL	Florida	South

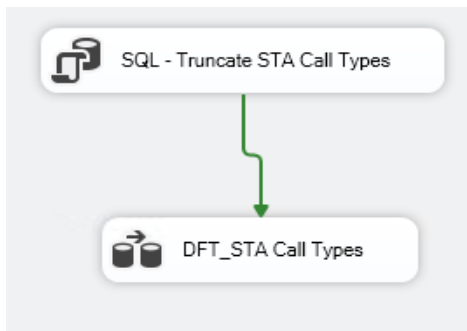
### 3. Call Types Table

Here is an extract from the file "Call Types Table.csv",

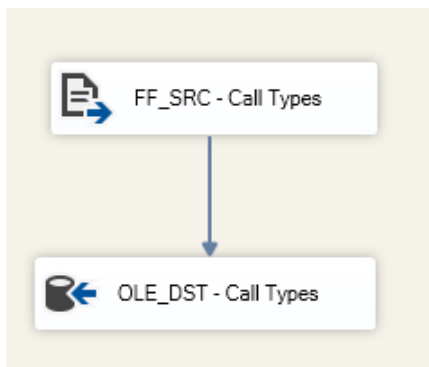
	A	B
1	CallTypeID	CallTypeLabel
2	1	Sales
3	2	Billing
4	3	Tech Support

For this table, we don't need to add additional data. Here we just make sure to truncate the table "Call Types" before running the package:





The dataflow is an import of a flat file :



We need to create the destination table with the following command:

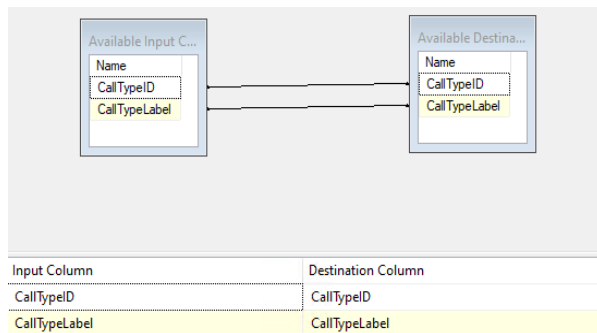
```

CREATE TABLE [OLE_DST - Call Types] (
    [CallTypeID] varchar(50),
    [CallTypeLabel] varchar(50)
)
  
```

The data is then loaded into the table “Call Types”:

The screenshot shows the 'OLE DB connection manager' dialog box. The 'OLE DB connection manager:' section has a dropdown menu set to 'localhost.A\_23\_ETL\_GroupProject\_STA'. The 'Data access mode:' section has a dropdown menu set to 'Table or view'. The 'Name of the table or the view:' section has a dropdown menu set to '[dbo].[OLE\_DST - Call Types]'. There is a small table icon to the left of the dropdown menu in the 'Name of the table or the view:' section.

We also make sure to change some of the names to avoid problems with accents, spaces and SQL reserved words:



Here are the lines of the results:

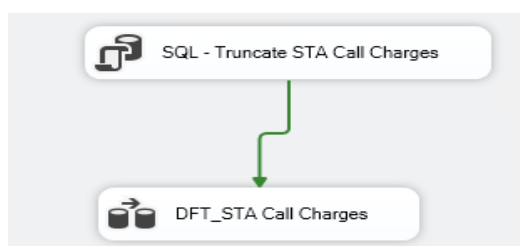
	CallTypeID	CallTypeLabel
1	1	Sales
2	2	Billing
3	3	Tech Support

## 4. Call Charges

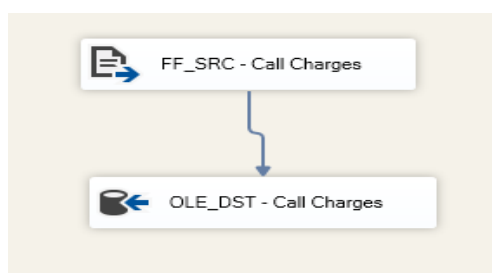
Here is an extract from the file “Call Charges Table.csv”,

Call Type	Call Charges (2018)	Call Charges (2019)	Call Charges (2020)	Call Charges (2021)
Sales	1.52 / min	1.56 / min	1.60 / min	1.71 / min
Billing	1.2 / min	1.32 / min	1.41 / min	1.45 / min
Tech Support	0.95 / min	0.98 / min	1.04 / min	1.12 / min

For this table, we don’t need to add additional data. Here we just make sure to truncate the table “Call Types” before running the package:



The dataflow is an import of a flat file :



We need to create the destination table with the following command:

```
CREATE TABLE [OLE_DST - Call Charges] (
    [Call Type ] varchar(50),
    [Call Charges (2018)] varchar(50),
    [Call Charges (2019)] varchar(50),
    [Call Charges (2020)] varchar(50),
    [Call Charges (2021)] varchar(50)
)
```

The data is then loaded into the table “Call Charges”:

OLE DB connection manager:

localhost.A\_23\_ETL\_GroupProject\_STA

Data access mode:

Table or view

Name of the table or the view:

[dbo].[OLE\_DST - Call Charges]

We also make sure to change some of the names to avoid problems with accents, spaces and SQL reserved words:

Available Input Columns:

Name
Call Type
Call Charges (2018)
Call Charges (2019)
Call Charges (2020)
Call Charges (2021)

Available Destination Columns:

Name
Call Type
Call Charges (2018)
Call Charges (2019)
Call Charges (2020)
Call Charges (2021)

Input Column	Destination Column
[Call Type ]	Call Type
Call Charges (2018)	Call Charges (2018)
Call Charges (2019)	Call Charges (2019)
Call Charges (2020)	Call Charges (2020)
Call Charges (2021)	Call Charges (2021)

Here is the lines of the results:

	Call Type	Call Charges (2018)	Call Charges (2019)	Call Charges (2020)	Call Charges (2021)
1	Sales	1.52 / min	1.56 / min	1.60 / min	1.71 / min
2	Billing	1.2 / min	1.32 / min	1.41 / min	1.45 / min
3	Tech Support	0.95 / min	0.98 / min	1.04 / min	1.12 / min

## 5. Call Data Table

Here is an extract of one file of “Data.csv”:

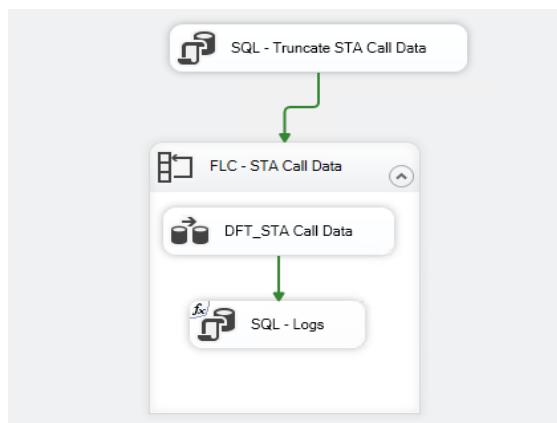
	A
1	CallTimestamp,Call Type,EmployeeID,CallDuration,WaitTime,CallAbandoned
2	5/4/2018 16:33,3,U559430,486,2,0
3	6/21/2018 18:28,3,A166733,945,0,0
4	6/21/2018 15:13,1,B971624,379,11,0
5	11/21/2018 13:02,3,U641256,1044,0,0
6	2/25/2018 13:36,1,P286634,1357,0,0
7	10/28/2018 10:04,3,M855788,570,23,0
8	12/17/2018 16:39,3,M794992,26,26,0
9	7/28/2018 19:09,2,I281837,8,8,0
10	11/6/2018 18:57,1,I192194,800,0,0

The data is separated into different files :

Nom	Modifié le	Type	Taille
Data 2018.csv	11/05/2023 09:10	Fichier CSV Micro...	1 149 Ko
Data 2019.csv	11/05/2023 09:10	Fichier CSV Micro...	1 147 Ko
Data 2020.csv	11/05/2023 09:10	Fichier CSV Micro...	1 145 Ko

To be able to use the data in the file, we need to put it into one table. This means that we need to extract the data from each data file.

To go into all data files, we use a “Foreach Loop Container” where we put the extraction dataflow inside. The data flow in the container will be able to load one files at the time. Here, we also truncate the data from the previous runs.



We use the enumerator “Foreach File Enumerator” to be able to iterate over the data files of the folder, it will loop over the files with .csv.

To keep track of the file we created a sql logs table, so we can see if every file has been extracted.

We use this expression to have all informations about the loop,

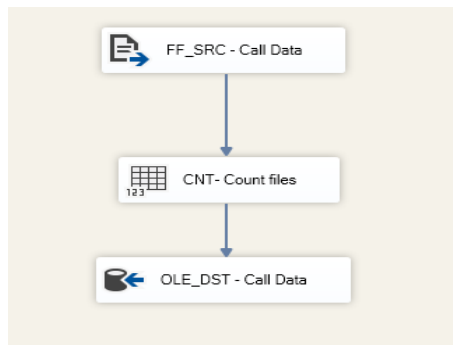
We need to create the destination table with the following command:

```
Create table FileInfo(
    Id int identity ,
    FilePath text,
    RecordCount int,
    Dated datetime
)
GO
```

Here are the lines of the results:

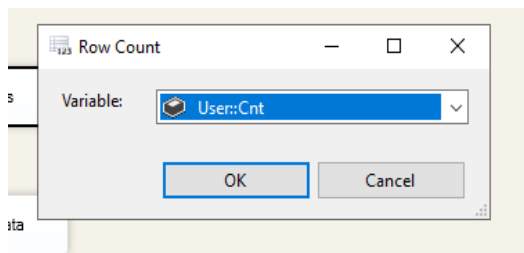
	Id	FilePath	RecordCount	Dated
1	1	C:\Users\Administrator\Desktop\Project ETL Data\...	33057	2024-05-12 11:24:50.603
2	2	C:\Users\Administrator\Desktop\Project ETL Data\...	33057	2024-05-12 11:25:10.873
3	3	C:\Users\Administrator\Desktop\Project ETL Data\...	33057	2024-05-12 11:25:31.670

Next, the dataflow is defined as follow:



To keep track of the line in all different data files we use a count file from the folder

We use the variable CNT to count the line from the folder,



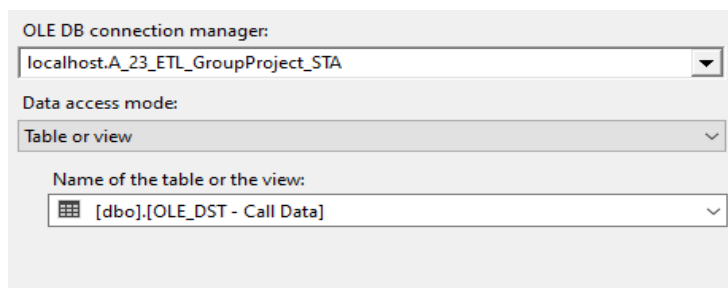
Once we have extracted the data, we can load it into our target table “Call Data” into the STA database.

First, we create the table with the following script:

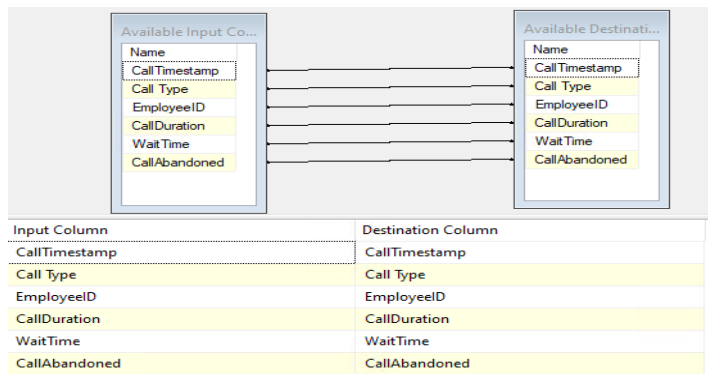
```

CREATE TABLE [OLE_DST - Call Data] (
    [CallTimestamp] varchar(50),
    [Call Type] varchar(50),
    [EmployeeID] varchar(50),
    [CallDuration] varchar(50),
    [WaitTime] varchar(50),
    [CallAbandoned] varchar(50)
)
  
```

We can now define the target destination.



And finally, we define the columns mapping as follow:



Here is the first ten lines of the results:

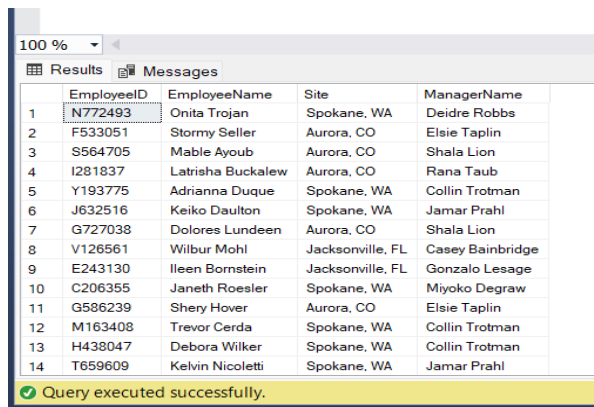
	CallTimestamp	Call Type	EmployeeID	CallDuration	WaitTime	CallAbandoned
1	5/4/2018 16:33	3	U559430	486	2	0
2	6/21/2018 18:28	3	A166733	945	0	0
3	6/21/2018 15:13	1	B971624	379	11	0
4	11/21/2018 13:02	3	U641256	1044	0	0
5	2/25/2018 13:36	1	P286634	1357	0	0
6	10/28/2018 10:04	3	M855788	570	23	0
7	12/17/2018 16:39	3	M794992	26	26	0
8	7/28/2018 19:09	2	I281837	8	8	0
9	11/6/2018 18:57	1	J192194	800	0	0
10	6/18/2018 15:32	2	F542348	651	111	0

## VII. ODS Database

Here, the role of the ODS database is to connect to the STA database and make all the transformations to clean the data.

### 1. ODS - Employee Table

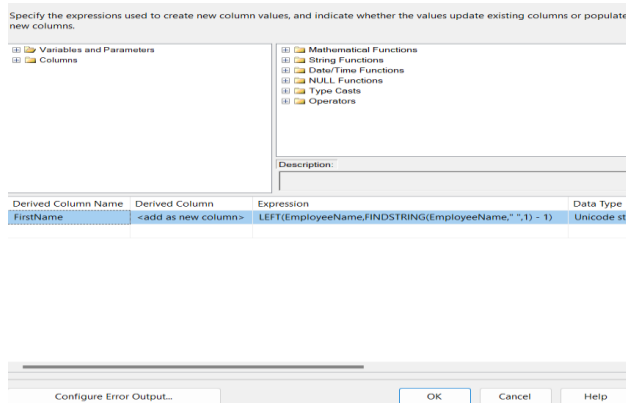
Here is an extract from the file “STA-Employee.csv” which represents our sources for the transformation:



	EmployeeID	EmployeeName	Site	ManagerName
1	N772493	Onita Trojan	Spokane, WA	Deidre Robbs
2	F533051	Stormy Seller	Aurora, CO	Elsie Taplin
3	S564705	Mable Ayoub	Aurora, CO	Shala Lion
4	I281837	Latrisha Buckalew	Aurora, CO	Rana Taub
5	Y193775	Adrianna Duque	Spokane, WA	Collin Trotman
6	J632516	Keiko Daulton	Spokane, WA	Jamar Prah
7	G727038	Dolores Lundeen	Aurora, CO	Shala Lion
8	V126561	Wilbur Mohl	Jacksonville, FL	Casey Bainbridge
9	E243130	Ileen Bornstein	Jacksonville, FL	Gonzalo Lesage
10	C206355	Janeth Roesler	Spokane, WA	Miyoko Degraw
11	G586239	Shery Hover	Aurora, CO	Elsie Taplin
12	M163408	Trevor Cerda	Spokane, WA	Collin Trotman
13	H438047	Debora Wilker	Spokane, WA	Collin Trotman
14	T659609	Kelvin Nicoletti	Spokane, WA	Jamar Prah

We made the following transformation in this table:

- A derived column transformation to extract the first name of employee.



Derived Column Name	Derived Column	Expression	Data Type
FirstName	<add as new column>	LEFT(EmployeeName,FINDSTRING(EmployeeName," ",1) - 1)	Unicode st

- A second derived column transformation to extract the last name of employee.



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Variables and Parameters  
Columns

Mathematical Functions  
String Functions  
Date/Time Functions  
NULL Functions  
Type Casts  
Operators

Description:

Derived Column Name	Derived Column	Expression	Data Type
FirstName	<add as new column>	LEFT(EmployeeName,FINDSTRING(EmployeeName," ",1) - 1)	Unicode st

Configure Error Output... OK Cancel Help

- We created the Email Address of each Employee by concatenating FirstName, LastName and Domain address which we chose as ServiceSpot.com

Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Variables and Parameters  
Columns

Mathematical Functions  
String Functions  
Date/Time Functions  
NULL Functions  
Type Casts  
Operators

Description:

Derived Column Name	Derived Column	Expression	Data Type	Length	Precision	Scale	Code Page
EmailAddress	<add as new column>	FirstName + " " + LastName + "@ServiceSpot.com"	Unicode string [DT_...]	117			

Configure Error Output... OK Cancel Help

- We separated the site column to have the SiteName and StateID

Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Variables and Parameters  
Columns

Mathematical Functions  
String Functions  
Date/Time Functions  
NULL Functions  
Type Casts  
Operators

Description:

Derived Column Name	Derived Column	Expression	Data Type	Length	Precision	Scale	Code Page
Site_ID	<add as new column>	LEFTSTRING(SiteID,1) - 1)	Unicode string [DT_...]	50			
State_ID	<add as new column>	TRIM(RIGHT(SiteID,LEN(SiteID) - FINDSTRING(SiteID," ",1)))	Unicode string [DT_...]	50			

Configure Error Output... OK Cancel Help

## - Resizing of SiteName, StateID and EmployeeID

Data Conversion Transformation Editor

Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.

Available Input Columns

- ☒ Name
- ☒ EmployeeID
- ☐ EmployeeName
- ☐ Site
- ☐ ManagerName
- ☐ FirstName
- ☐ LastName

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
State_ID	StateID_R	string [DT_STR]	10			1252 (ANSI - Latin I)
Site_D	Site_R	string [DT_STR]	50			1252 (ANSI - Latin I)
EmployeeID	EmployeeID	string [DT_STR]	20			1252 (ANSI - Latin I)

Configure Error Output... OK Cancel Help

## - Getting StateName and the region of each Employee from the STA - US STATE TABLE

Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

General

Connection

**Columns**

Advanced

Error Output

Available Input Columns

- Name
- OLE\_SRC\_STA-EmployeeID
- EmployeeName
- Site
- ManagerName
- FirstName
- LastName
- EmailAddress
- Site\_D
- State\_ID

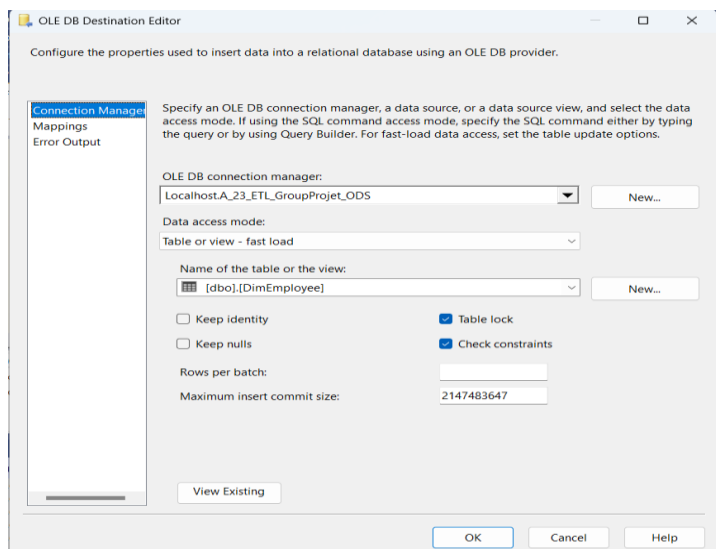
Available Lookup Columns

- ☒ Name
- ☐ StateCD
- ☒ Name
- ☒ Region

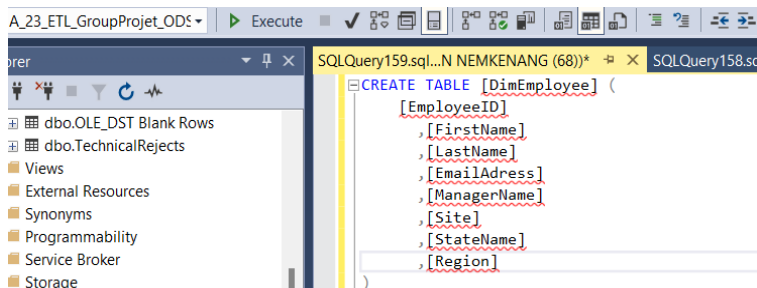
Lookup Column	Lookup Operation	Output Alias
Name	<add as new column>	StateName
Region	<add as new column>	Region

OK Cancel Help

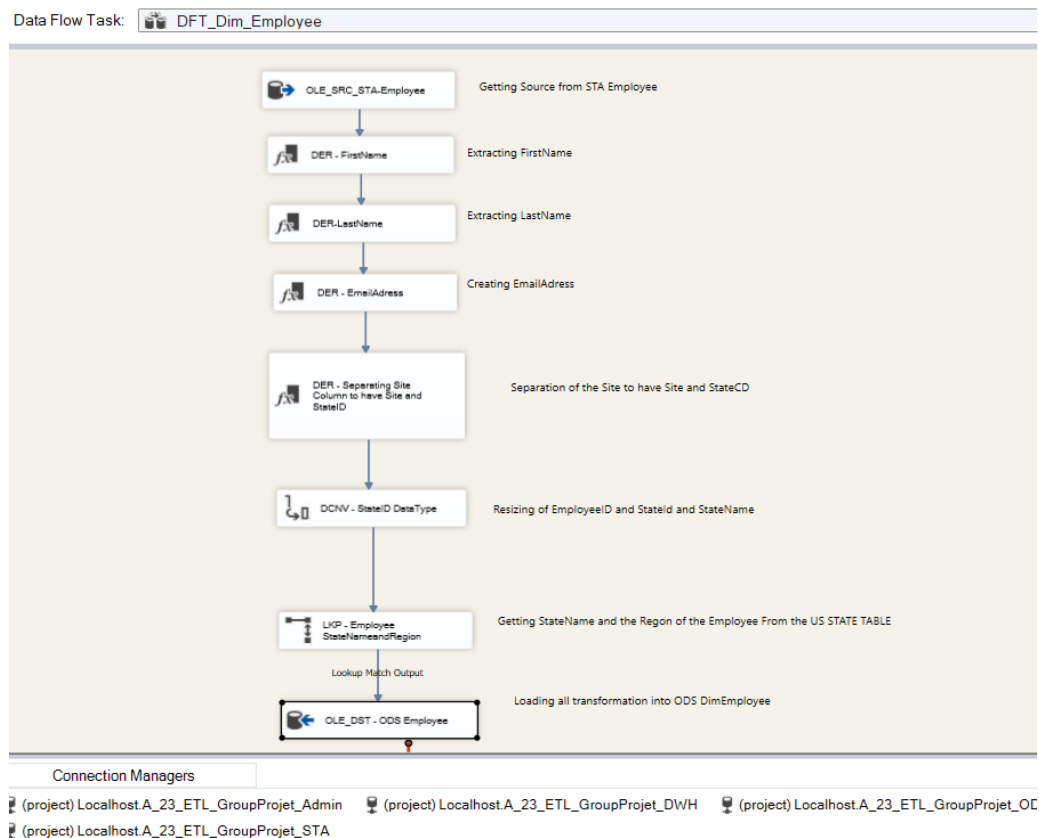
- Loading all the transformations into ODS - DimEmployee



We had to create a table with this script :



Here is the result in the Data Flow view:



## 2. ODS - Call Charges Table

Getting sources from the STA - Call Charges:

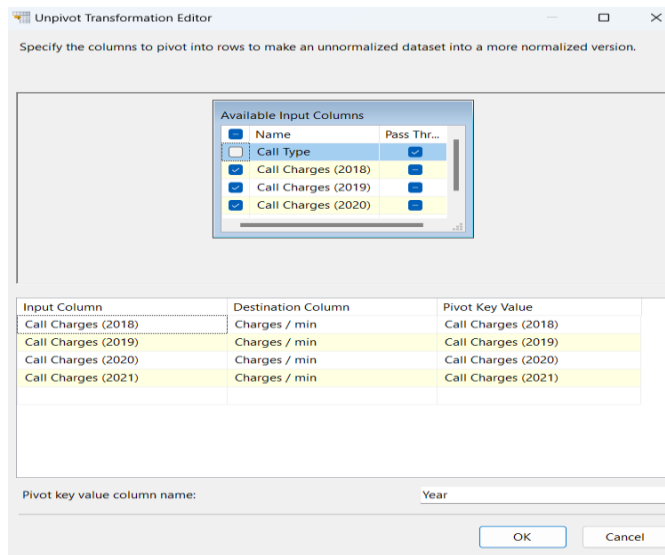
	Call Type	Call Charges (2018)	Call Charges (2019)	Call Charges (2020)	Call Charges (2021)
1	Sales	1.52 / min	1.56 / min	1.60 / min	1.71 / min
2	Billing	1.2 / min	1.32 / min	1.41 / min	1.45 / min
3	Tech Support	0.95 / min	0.98 / min	1.04 / min	1.12 / min

Before the unpivoting, we used a query in SQL to delete the empty rows in the STA - CallCharges table:

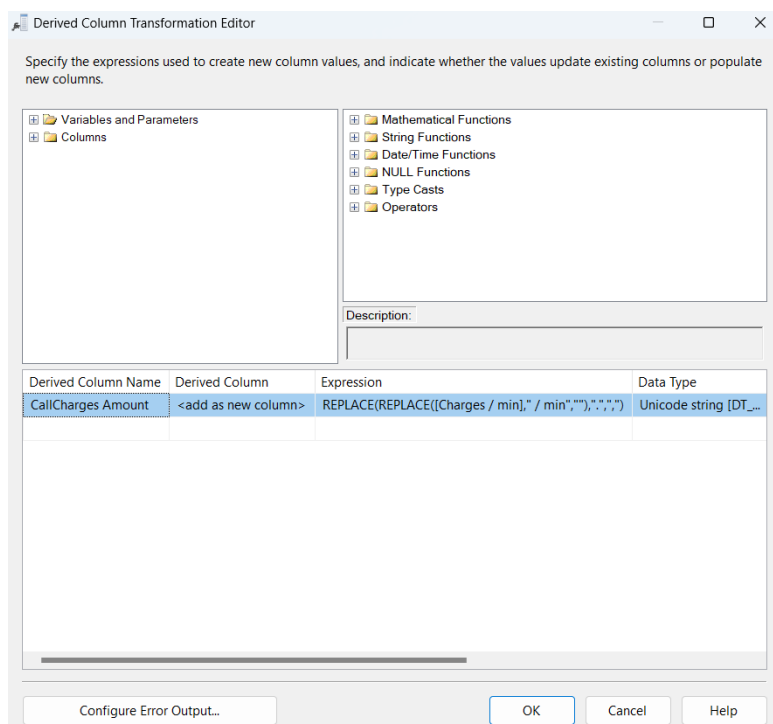
```

DELETE FROM [A_23_ETL_GroupProjet_STA].[dbo].[OLE_DST - Call Charges]
WHERE [Call Type] = ''
AND [Call Charges (2018)] = ''
AND [Call Charges (2019)] = ''
AND [Call Charges (2020)] = ''
AND [Call Charges (2021)] = ''
  
```

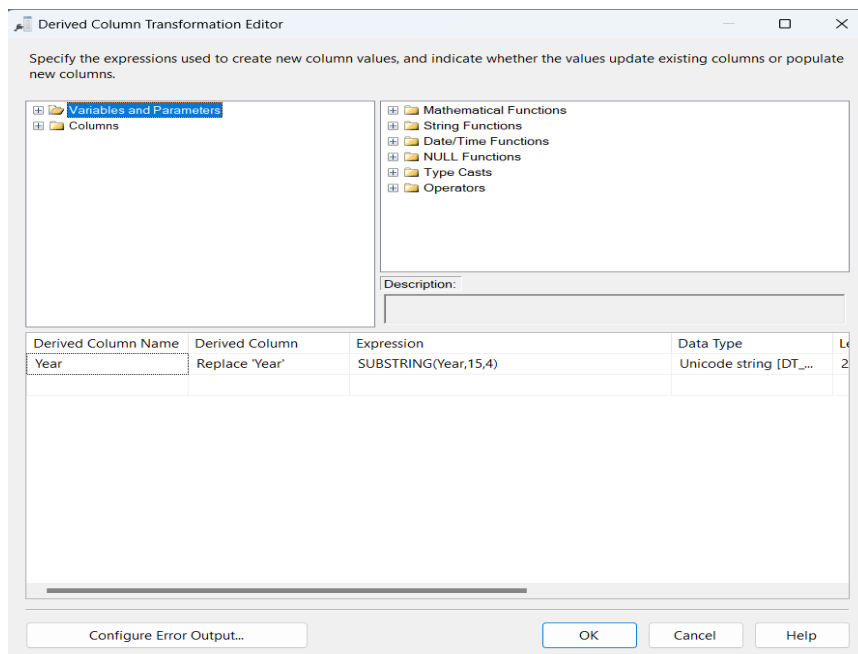
Unpivoting the STA - Call Charges table:



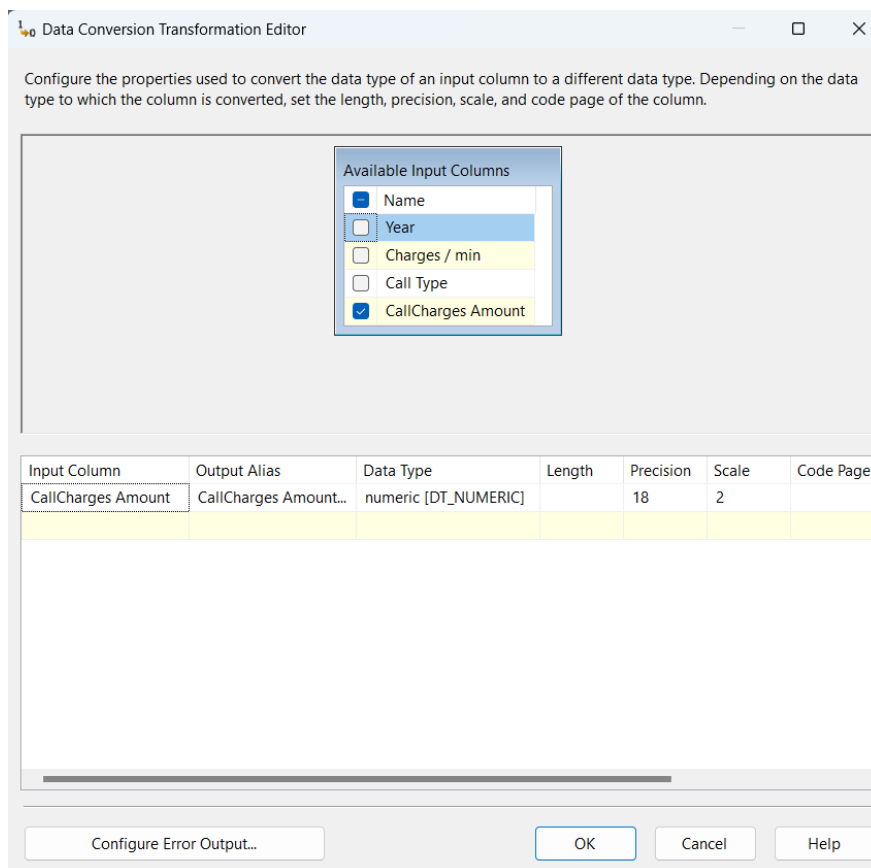
Extracting Charges Amount with the derived column:



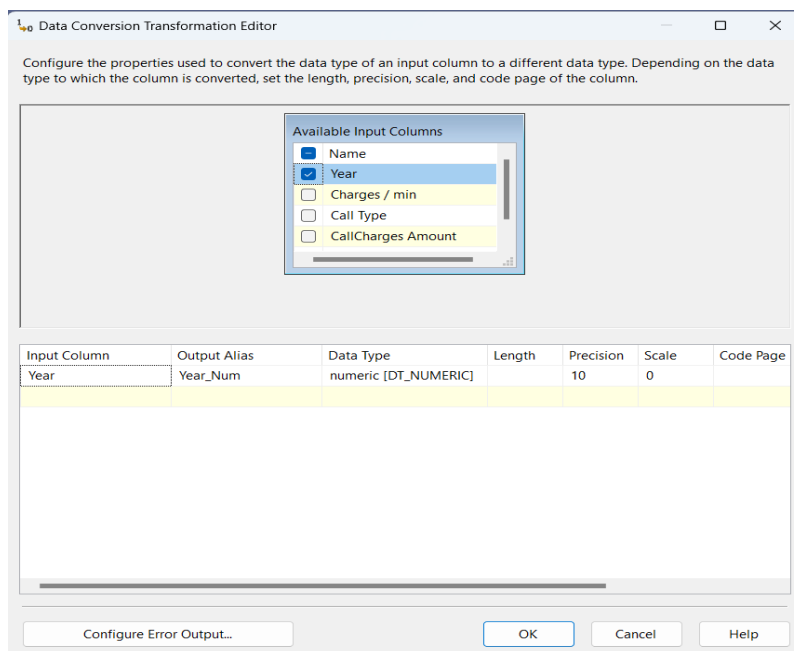
Extracting the Year with the derived column in a new column:



Converting ChargesAmount to numeric DataType:



Converting the Year STR obtained to numeric:

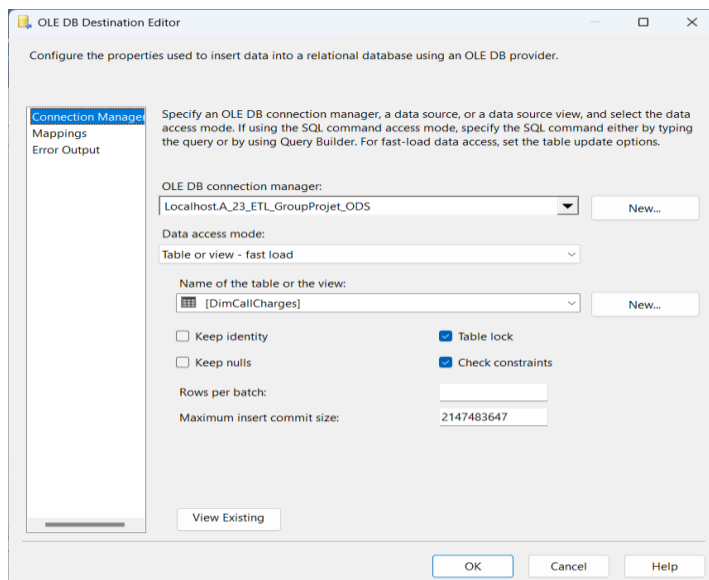


The Data Conversion Transformation Editor window is shown. It has a title bar with a logo, the text 'Data Conversion Transformation Editor', and standard window controls. Below the title bar is a descriptive text: 'Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.'

There is a section titled 'Available Input Columns' with a list of columns: 'Name' (checked), 'Year' (checked), 'Charges / min' (unchecked), 'Call Type' (unchecked), and 'CallCharges Amount' (unchecked). Below this is a table with the following columns: 'Input Column', 'Output Alias', 'Data Type', 'Length', 'Precision', 'Scale', and 'Code Page'. The first row of the table is highlighted in yellow and contains the following values: 'Year', 'Year\_Num', 'numeric [DT\_NUMERIC]', an empty 'Length' cell, '10' for 'Precision', '0' for 'Scale', and an empty 'Code Page' cell.

At the bottom of the window are four buttons: 'Configure Error Output...', 'OK', 'Cancel', and 'Help'.

Loading all the transformation into ODS - DimCallCharges:



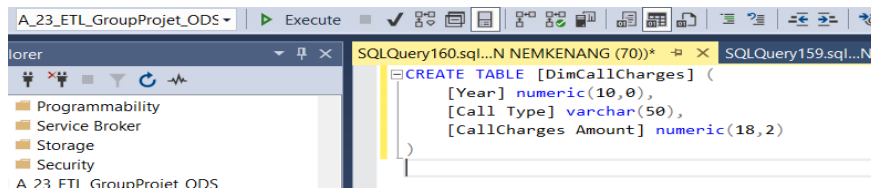
The OLE DB Destination Editor window is shown. It has a title bar with a logo, the text 'OLE DB Destination Editor', and standard window controls. Below the title bar is a descriptive text: 'Configure the properties used to insert data into a relational database using an OLE DB provider.'

On the left side, there is a 'Connection Manager' pane with a tree view showing 'Mappings' and 'Error Output'. The main area contains the following settings:

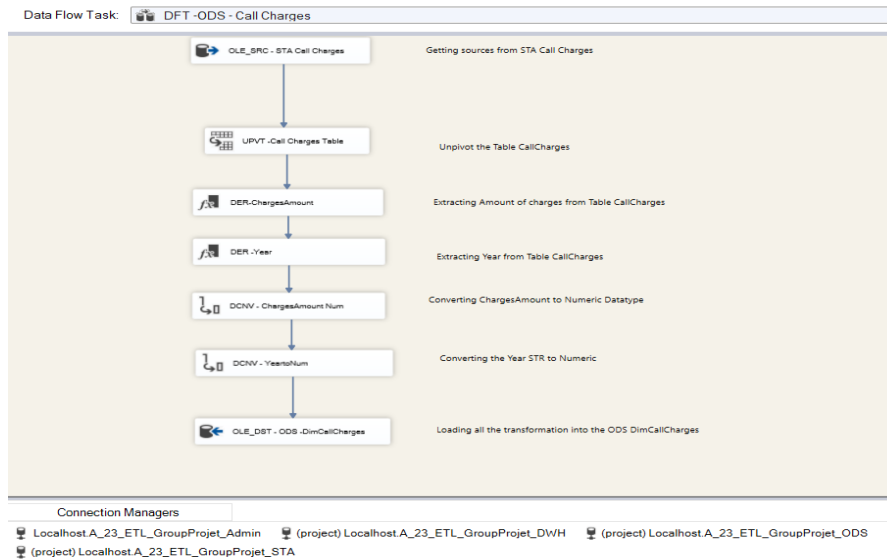
- 'OLE DB connection manager:' dropdown menu set to 'Localhost\_A\_23\_ETL\_GroupProjet\_ODS' with a 'New...' button.
- 'Data access mode:' dropdown menu set to 'Table or view - fast load'.
- 'Name of the table or the view:' dropdown menu set to '[DimCallCharges]' with a 'New...' button.
- Checkboxes for 'Keep identity' (unchecked), 'Keep nulls' (unchecked), 'Table lock' (checked), and 'Check constraints' (checked).
- 'Rows per batch:' text box with the value '2147483647'.
- 'Maximum insert commit size:' text box with the value '2147483647'.
- A 'View Existing' button.

At the bottom of the window are three buttons: 'OK', 'Cancel', and 'Help'.

For that we created the destination table in the ODS database using this query:



Here is the result in the Data Flow view:



### 3. ODS - Call Data Table

Getting the source from the STA - CallData, here is an extract from the file:

Results		Messages					
	CallTimestamp	Call Type	EmployeeID	CallDuration	WaitTime	CallAbandoned	
1	5/4/2018 16:33	3	U559430	486	2	0	
2	6/21/2018 18:28	3	A166733	945	0	0	
3	6/21/2018 15:13	1	B971624	379	11	0	
4	11/21/2018 13:02	3	U641256	1044	0	0	
5	2/25/2018 13:36	1	P286634	1357	0	0	
6	10/28/2018 10:04	3	M855788	570	23	0	
7	12/17/2018 16:39	3	M794992	26	26	0	
8	7/28/2018 19:09	2	I281837	8	8	0	
9	11/6/2018 18:57	1	J192194	800	0	0	
10	6/18/2018 15:32	2	F542348	651	111	0	
11	5/14/2018 16:24	2	J632516	229	10	0	
12	6/2/2018 12:57	3	U194381	467	23	0	
13	6/20/2018 12:01	3	G586239	1290	7	0	
14	12/20/2018 16:27	3	N772493	504	0	0	

Query executed successfully.



## Converting CallTimeStamp to Date:

The Data Conversion Transformation Editor window is shown with the following configuration:

**Available Input Columns:**

- ☒ Name
- ☒ CallTimeStamp
- ☐ CallTypeID
- ☐ EmployeeID
- ☐ CallDuration
- ☐ WaitTime

**Table:**

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
CallTimeStamp	CallTimeStamp_Date	date [DT_DATE]				

**Buttons:** Configure Error Output..., OK, Cancel, Help

## Converting CallDuration to INT :

The Data Conversion Transformation Editor window is shown with the following configuration:

**Available Input Columns:**

- ☒ Name
- ☐ CallTimeStamp
- ☐ CallTypeID
- ☐ EmployeeID
- ☒ CallDuration
- ☐ WaitTime
- ☐ CallAbandoned

**Table:**

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
CallDuration	CallDuration_Int	four-byte signed integ...				

**Buttons:** Configure Error Output..., OK, Cancel, Help

## Converting WaitTime to INT:

The screenshot shows the 'Data Conversion Transformation Editor' window. At the top, a text box explains: 'Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.'

Below this is a large grey area containing a list box titled 'Available Input Columns'. The list includes: Name (checked), CallTimestamp (highlighted), CallTypeID, EmployeeID, CallDuration, WaitTime (checked), and CallAbandoned.

Below the list box is a table with the following columns: Input Column, Output Alias, Data Type, Length, Precision, Scale, and Code Page.

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
WaitTime	WaitTime_Int	four-byte signed integ...				

At the bottom of the window are four buttons: 'Configure Error Output...', 'OK', 'Cancel', and 'Help'.

## Converting AbandonedCall to INT:

The screenshot shows the 'Data Conversion Transformation Editor' window. At the top, a text box explains: 'Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.'

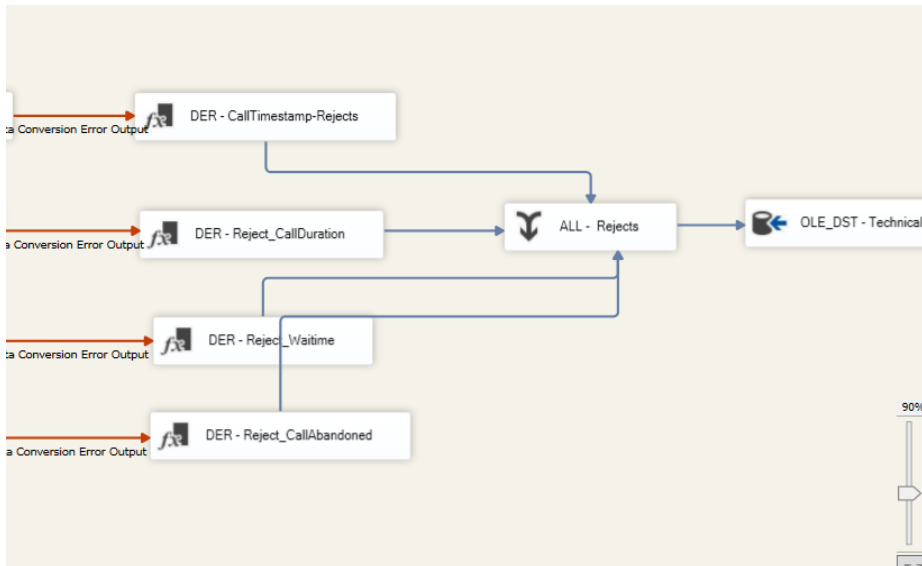
Below this is a large grey area containing a list box titled 'Available Input Columns'. The list includes: Name (checked), CallTimestamp (highlighted), CallTypeID, EmployeeID, CallDuration, WaitTime, and CallAbandoned (checked).

Below the list box is a table with the following columns: Input Column, Output Alias, Data Type, Length, Precision, Scale, and Code Page.

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
CallAbandoned	CallAbandoned_Int	four-byte signed integ...				

At the bottom of the window are four buttons: 'Configure Error Output...', 'OK', 'Cancel', and 'Help'.

Pushing all the potential rejects by creating derived columns and sending them to the Admin Database in the table TechnicalRejects:



Adding the Service Legal Agreement status in a new column :

Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

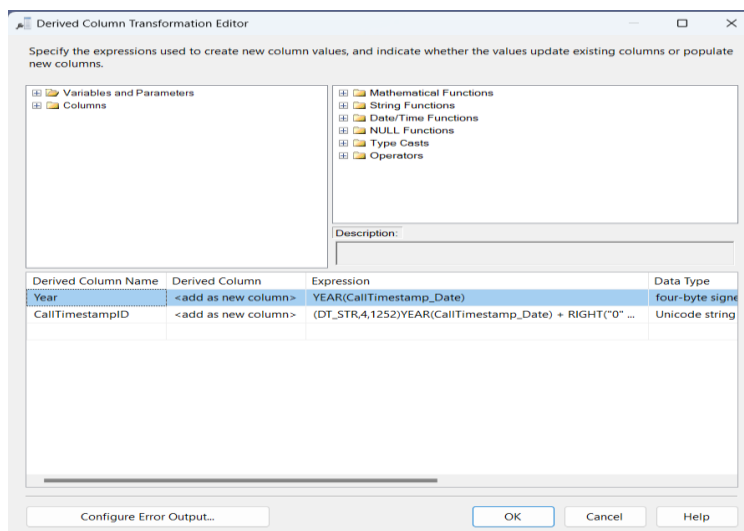
Variables and Parameters  
Columns

Mathematical Functions  
String Functions  
Date/Time Functions  
NULL Functions  
Type Casts  
Operators

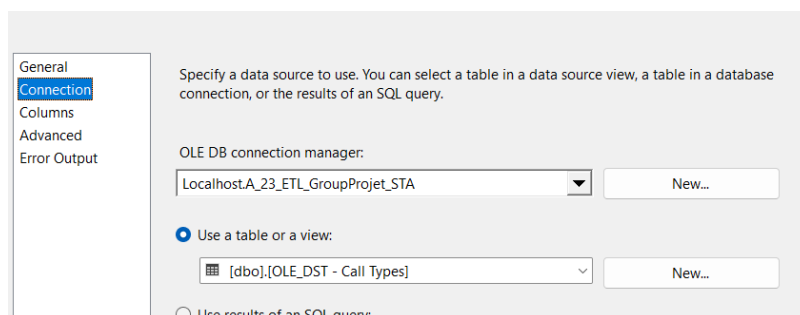
Description:

Derived Column Name	Derived Column	Expression	Data Type
SLA	<add as new column>	(WaitTime_Int < 35) ? "Within SLA" : "Outside SLA"	Unicode string [DT_...

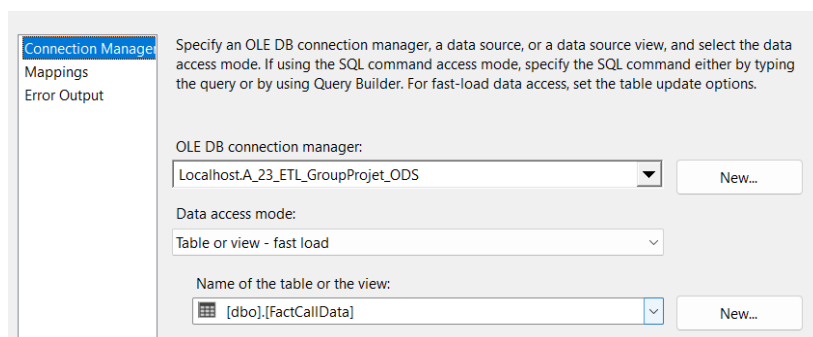
Creation of two new columns to compute the CallTimeStamp Identifier and Year of Call :



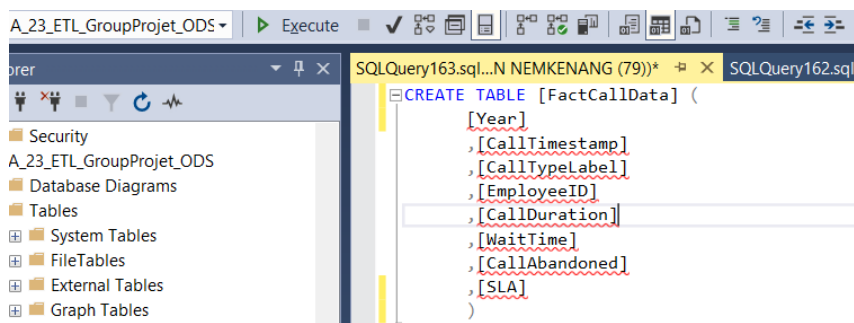
Getting the CallTypesLabel from the STA - Call Types table :



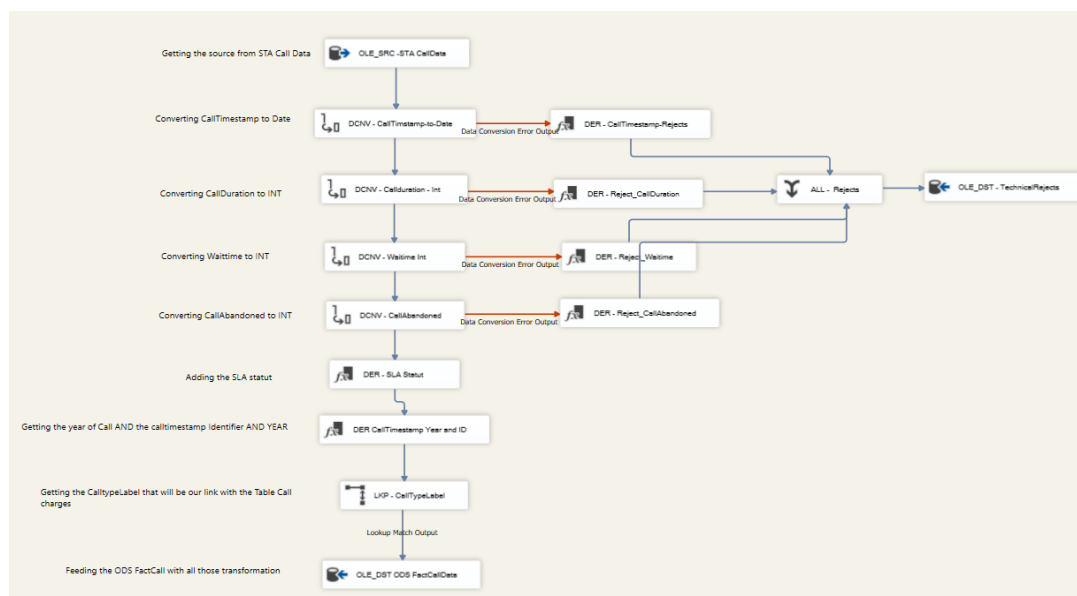
Loading all the transformation into the ODS - FactCallData :



For that we created the destination table in the ODS database using this query:



Here is the result in the Data Flow view:



## VIII. DWH Database

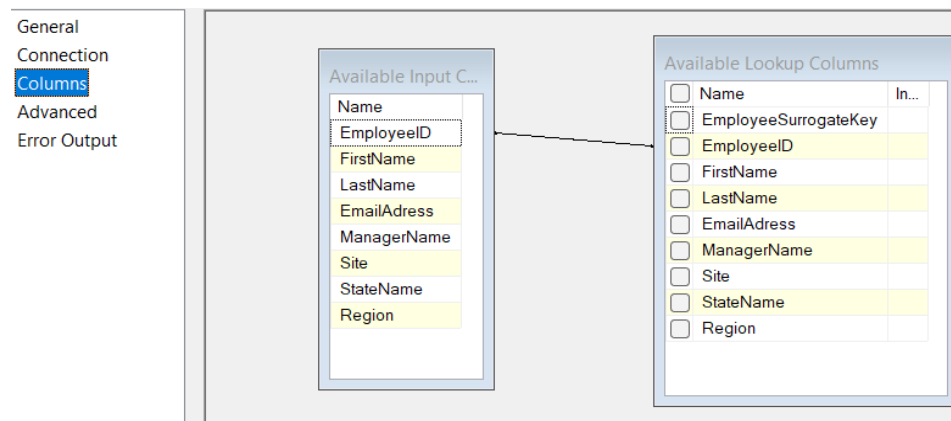
Here, the role of the DWH Database is to connect to the ODS Database and see if there are any changes, update it and push it to the DWH Database.

### 4. DimEmployee

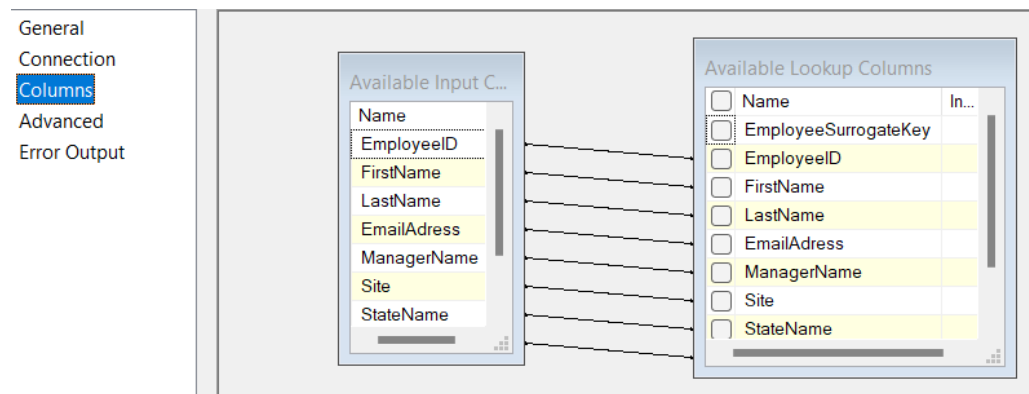
Getting the ODS - Employee table, here is an extract of the table:

	EmployeeID	FirstName	LastName	EmailAddress	ManagerName	Site	StateName	Region
1	N772493	Onita	Trojan	Onita.Trojan@ServiceSpot.com	Deidre Robbs	Spokane	Washington	West
2	F533051	Stormy	Seller	Stormy.Seller@ServiceSpot.com	Elsie Taplin	Aurora	Colorado	West
3	S564705	Mable	Ayoub	Mable.Ayoub@ServiceSpot.com	Shala Lion	Aurora	Colorado	West
4	I281837	Latrisha	Buckalew	Latrisha.Buckalew@ServiceSpot.com	Rana Taub	Aurora	Colorado	West
5	Y193775	Adrianna	Duque	Adrianna.Duque@ServiceSpot.com	Collin Trotman	Spokane	Washington	West
6	J632516	Keiko	Daulton	Keiko.Daulton@ServiceSpot.com	Jamar Prah	Spokane	Washington	West
7	G727038	Dolores	Lundeen	Dolores.Lundeen@ServiceSpot.com	Shala Lion	Aurora	Colorado	West
8	V126561	Wilbur	Mohl	Wilbur.Mohl@ServiceSpot.com	Casey Bainbridge	Jacksonville	Florida	South
9	E243130	Ileen	Bornstein	Ileen.Bornstein@ServiceSpot.com	Gonzalo Lesage	Jacksonville	Florida	South
10	C206355	Janeth	Roesler	Janeth.Roesler@ServiceSpot.com	Miyoko Degraw	Spokane	Washington	West

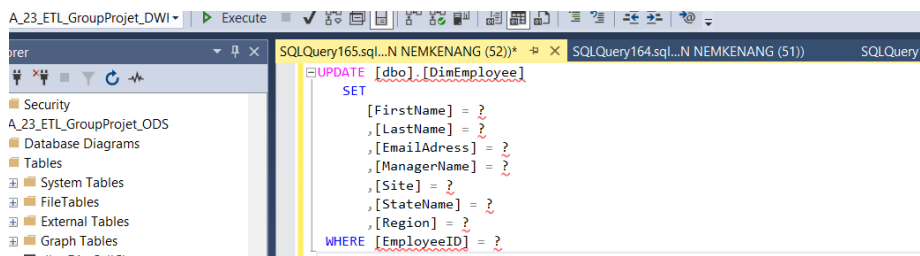
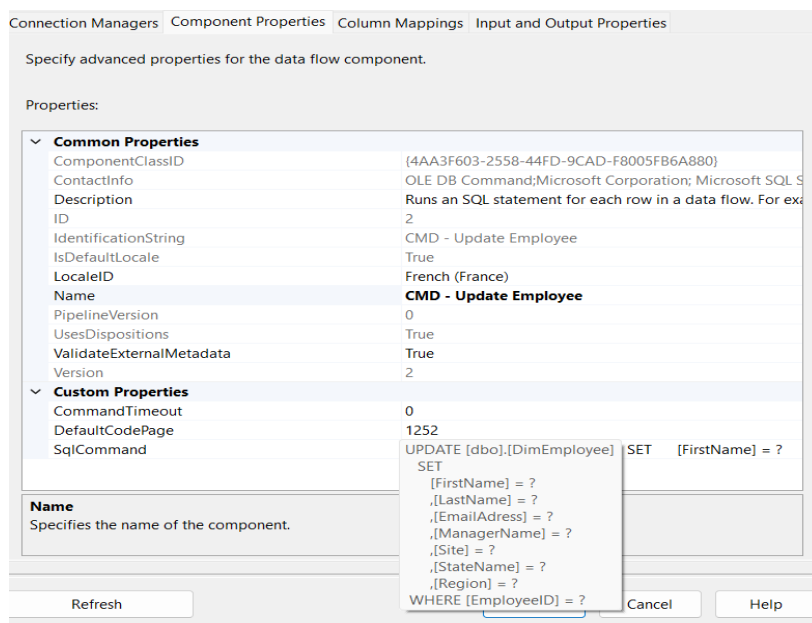
Lookup the EmployeeID which represents the connection between the ODS - Employee table and DWH - Employee table:



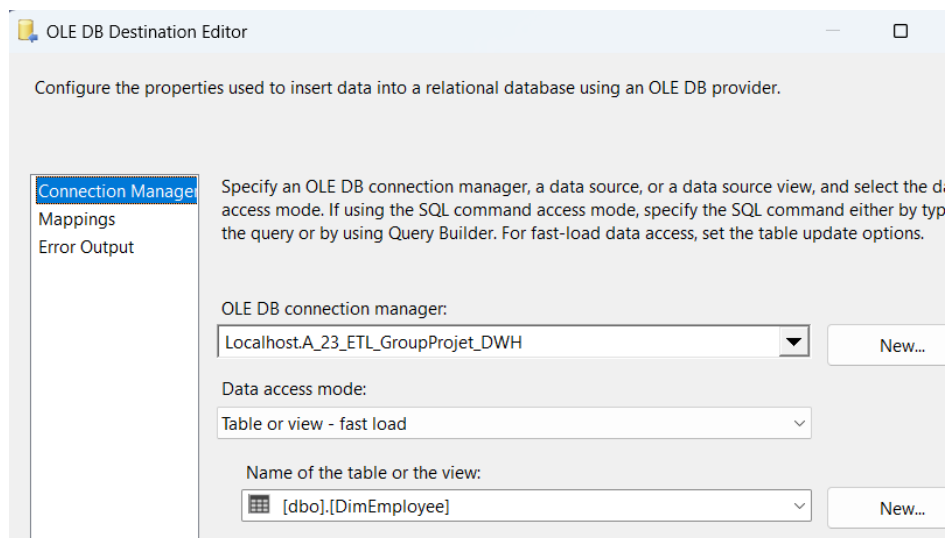
For the Match Output Lookup, we made a second Lookup to see if there are any changes:



In case of changes, we made an Update :



Loading all the transformation into the DWH - DimEmployee:



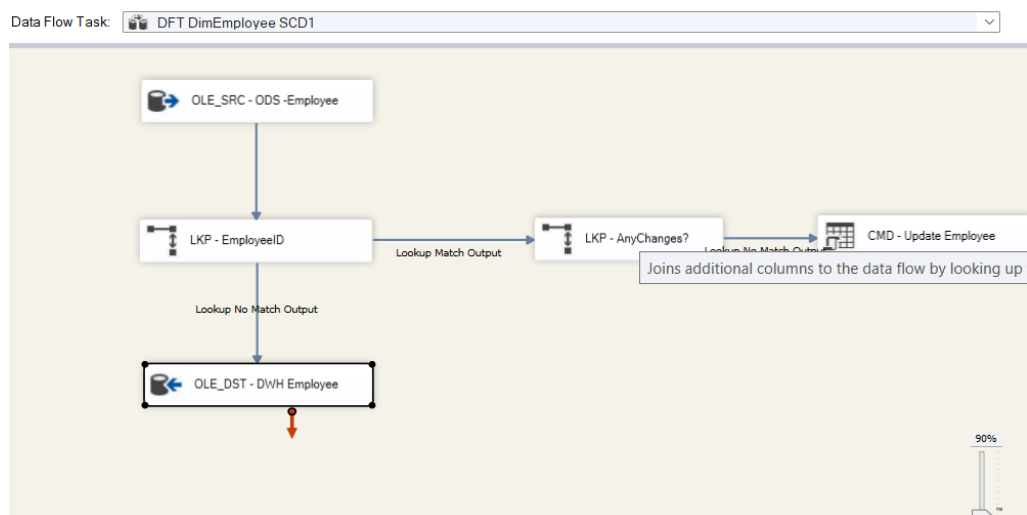
For that we created the destination table in the DWH database using this query :

```

CREATE TABLE [dbo].[DimEmployee](
    [EmployeeSurrogateKey] [int] IDENTITY(1,1) NOT NULL,
    [EmployeeID] [varchar](50) NULL,
    [FirstName] [nvarchar](50) NULL,
    [LastName] [nvarchar](50) NULL,
    [EmailAddress] [nvarchar](117) NULL,
    [ManagerName] [varchar](50) NULL,
    [Site] [varchar](50) NULL,
    [StateName] [varchar](50) NULL,
    [Region] [varchar](50) NULL,
)

```

Here is the result in the Data Flow view:



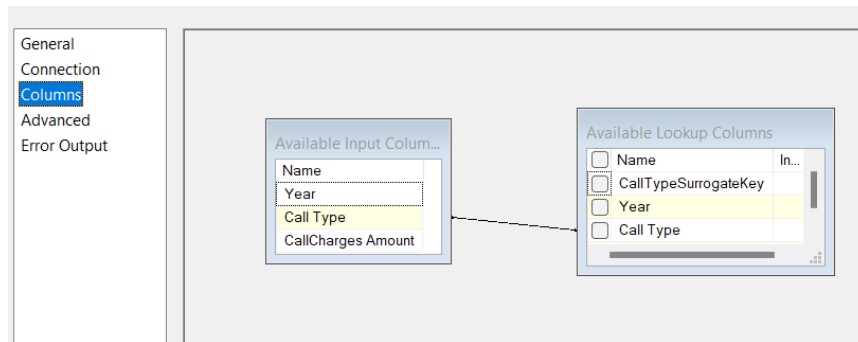
## 5. DimCallCharges

Getting the ODS - DimCallCharges table, here is an extract of the table:

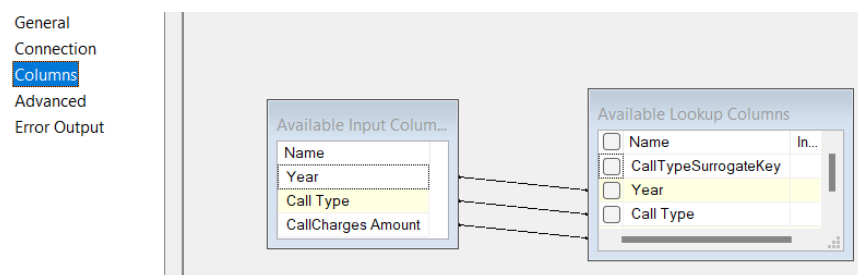
Results		Messages	
	Year	Call Type	CallCharges Amount
1	2018	Sales	1.52
2	2019	Sales	1.56
3	2020	Sales	1.60
4	2021	Sales	1.71
5	2018	Billing	1.20
6	2019	Billing	1.32
7	2020	Billing	1.41
8	2021	Billing	1.45
9	2018	Tech Support	0.95
10	2019	Tech Support	0.98
11	2020	Tech Support	1.04
12	2021	Tech Support	1.12



Lookup the Call Types which represents the connection between the ODS - DimCallCharges table and DWH - DimCallCharges:



For the Match Output Lookup, we made a second Lookup to see if there are any changes:



In case of changes, we made an Update:

Connection Managers | Component Properties | Column Mappings | Input and Output Properties

Specify advanced properties for the data flow component.

Properties:

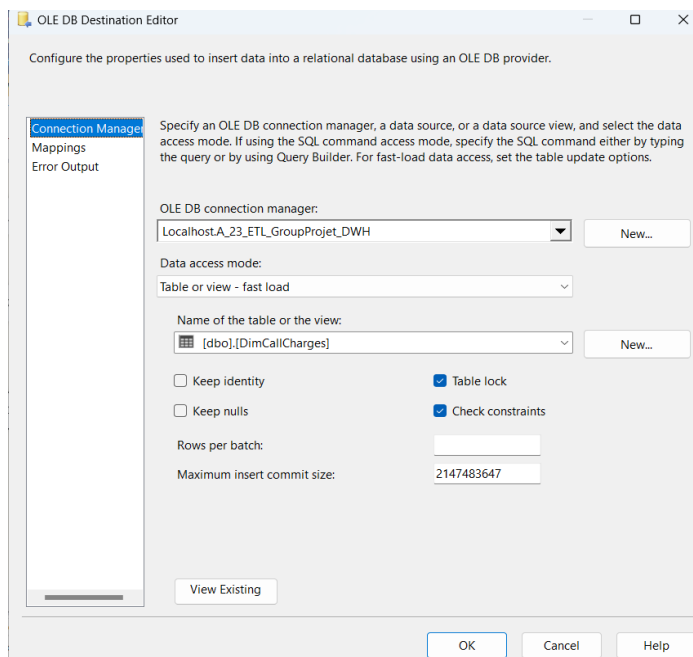
Common Properties	
ComponentClassID	{4AA3F603-2558-44FD-9CAD-F8005FB6A880}
ContactInfo	OLE DB Command;Microsoft Corporation; Microsoft SQL S
Description	Runs an SQL statement for each row in a data flow. For ex
ID	2
IdentificationString	CMD - Update Callcharges
IsDefaultLocale	True
LocaleID	French (France)
Name	<b>CMD - Update Callcharges</b>
PipelineVersion	0
UsesDispositions	True
ValidateExternalMetadata	True
Version	2
Custom Properties	
CommandTimeout	0
DefaultCodePage	1252
SqlCommand	UPDATE [dbo].[DimCallCharges] SET [Year] = ? ,[CallC SET [Year] = ? ,[CallCharges Amount] = ? WHERE [Call Type] = ?

A\_23\_ETL\_GroupProjet\_DWI | Execute | SQLQuery170.sql...N NEMKENANG (75)\* | SQLQuery169

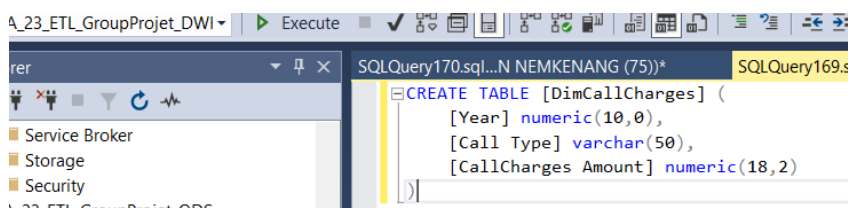
```
UPDATE [dbo].[DimCallCharges]
SET [Year] = ?
,[CallCharges Amount] = ?
WHERE [Call Type] = ?
```

Service Broker | Storage | Security | A\_23\_ETL\_GroupProjet\_ODS

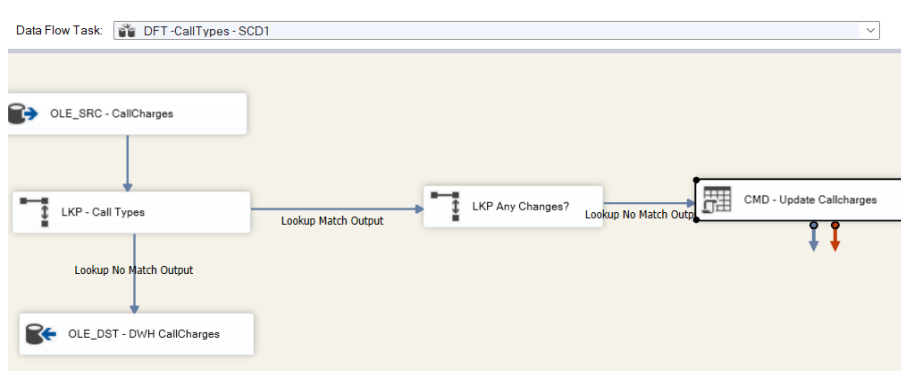
Loading all the transformation into the DWH - Call Charges:



For that we created the destination table in the DWH database using this query:



Here is the result in the Data Flow view:



## 6. Fact Table Call Data

Getting source from ODS - Call Data, here is an extract of the table:

	CallTimestampID	Year	CallTimestamp	CallTypeLabel	EmployeeID	CallDuration	WaitTime	CallAbandoned	SLA
1	2018031319300001280	2018	2018-03-13 19:30:00.000	Sales	J173690	1280	203	0	Outside SLA
2	2018072919450000412	2018	2018-07-29 19:45:00.000	Billing	D411745	412	146	0	Outside SLA
3	201808110830000011	2018	2018-08-11 08:30:00.000	Billing	J549755	11	11	0	Within SLA
4	2018092018090000992	2018	2018-09-20 18:09:00.000	Billing	K399501	992	25	0	Within SLA
5	2018050916060000545	2018	2018-05-09 16:06:00.000	Sales	L736768	545	24	0	Within SLA
6	2018100313120001397	2018	2018-10-03 13:12:00.000	Tech Support	C206355	1397	0	0	Within SLA
7	201808071500000073	2018	2018-08-07 15:00:00.000	Billing	A475155	73	0	0	Within SLA
8	2018083017150001426	2018	2018-08-30 17:15:00.000	Tech Support	X991580	1426	12	0	Within SLA
9	201803281012000913	2018	2018-03-28 10:12:00.000	Billing	Y193775	913	5	0	Within SLA
10	2018112017340000892	2018	2018-11-20 17:34:00.000	Sales	J192194	892	0	0	Within SLA
11	201808030947000600	2018	2018-08-03 09:47:00.000	Billing	D411745	600	0	0	Within SLA
12	201811151725000486	2018	2018-11-15 17:25:00.000	Tech Support	N620104	486	19	0	Within SLA
13	201804161941000690	2018	2018-04-16 19:41:00.000	Tech Support	T398672	690	7	0	Within SLA

Lookup EmployeeID in DimEmployee to retrieve EmployeeSurrogateKey:

The screenshot shows the 'Columns' tab of the Lookup Transformation Editor. On the left, the 'Available Input Columns' list includes Name, CallTimestamp, CallTypeLabel, EmployeeID, CallDuration, WaitTime, CallAbandoned..., SLA, and CallTimestamp... A line connects 'EmployeeID' to the 'Available Lookup Columns' list on the right. In the 'Available Lookup Columns' list, 'EmployeeSurrogateKey' is selected. Below the lists, the 'Lookup Column' is 'EmployeeSurrogateKey', the 'Lookup Operation' is '<add as new column>', and the 'Output Alias' is 'EmployeeSurrogateKey'.

For the Match Output Lookup, we made a second Lookup to retrieve CallTypeSurrogateKey:

The screenshot shows the 'Columns' tab of the Lookup Transformation Editor. On the left, the 'Available Input Columns' list includes Name, CallTimestamp, CallTypeLabel, EmployeeID, CallDuration, WaitTime, CallAbandon..., SLA, CallTimestamp..., and Year. A line connects 'CallTypeLabel' to the 'Available Lookup Columns' list on the right. In the 'Available Lookup Columns' list, 'CallTypeSurrogateKey' is selected. Below the lists, the 'Lookup Column' is 'CallTypeSurrogateKey', the 'Lookup Operation' is '<add as new column>', and the 'Output Alias' is 'CallTypeSurrogateKey'.

We added a Derived Column to convert CallTimeStamps to DateType:

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

**Variables and Parameters**

Columns

**Mathematical Functions**

String Functions

Date/Time Functions

NULL Functions

Type Casts

Operators

Description:

Derived Column Name	Derived Column	Expression	Data Type
CallTimestamp_Date	<add as new column>	(DT_DBDATE)CallTimestamp	database date [DT_D...

Lookup DimDate to retrieve DateKey :

General

Connection

**Columns**

Advanced

Error Output

Available Input C...

Name
CallTimestamp
CallTypeLabel
EmployeeID
CallDuration
WaitTime
CallAbandon...
SLA
CallTimestam...
Year

Available Lookup C...

Name	In...
<input checked="" type="checkbox"/> DateKey	
<input type="checkbox"/> Date	
<input type="checkbox"/> Day	
<input type="checkbox"/> DaySuffix	
<input type="checkbox"/> Weekd...	
<input type="checkbox"/> WeekD...	
<input type="checkbox"/> WeekD...	
<input type="checkbox"/> WeekD...	
<input type="checkbox"/> DOWIn...	

Lookup Column	Lookup Operation	Output Alias
DateKey	<add as new column>	DateKey

Loading all the transformation into the DWH - FactCallData:

Configure the properties used to insert data into a relational database using an OLE DB provider.

**Connection Manager**  
Mappings  
Error Output

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager:

Data access mode:

Name of the table or the view:

☐ Keep identity
☒ Table lock

☐ Keep nulls
☒ Check constraints

For that we created the destination table in the DWH database using this query:

```
SQLQuery172.sql...N NEMKENANG (68))* × SQLQuery171.sql...N NEMKENANG (52)) SQ
CREATE TABLE [FactCallData] ( [CallTimestampID]
, [Year]
, [DateKey]
, [CallTimestamp]
, [CallTypeSurKey]
, [EmployeeSurKey]
, [CallDuration]
, [WaitTime]
, [CallAbandoned]
, [SLA]
```

Here is the result in the Data Flow view:

