TABLE OF CONTENT

INTRODUCTION

To assist businesses in managing large volumes of data, the ETL (Extract, Transform, Load) process is the best way to retrieve data often scattered across different files, transform it according to the company's needs, and centralize it in a Data Warehouse. Once there, the data is structured and ready for analysis, allowing businesses to save time and increase efficiency when making decisions.

- CONTEXT

The IT company 'ServiceSpot' has reached out to us for assistance in analyzing their call center data.

Receiving a significant volume of calls daily, this information is logged across multiple files, making it challenging to utilize in its current format.

To gain a better understanding of their service performance, they've tasked us with restructuring and centralizing this data into a single data warehouse. This will enable more straightforward and efficient analyses.

- TOOLS

To realize this project, we will use the following tools :

- MS SQL Server
- SSIS Module on VISUAL STUDIO

- DATASETS

ServiceSpot provides us with 7 .csv files.

- Four of them are information about employees/ Call's types/ Call's charges and States :

Employees Table :

>> *Site* = City + StateCD

Column Name	Description
EmployeeID	Employee unique ID
EmployeeName	Employee full name
Site	Site name where the employee is working at
ManagerName	Employee's Manager

Call_Types Table :

Column Name	Description
CallTypeID	Unique ID for Call Type
CallTypeLabel	Call type label

Call_Charges Table :

>> *Call Charges/ Min (YYYY))* = The amount of money that is charged to customers per minute, and for each year

Column Name	Description
Call Type Key	Unique ID for Call Type
Call Type	Call type label
Call Charges / Min (YYYY)	The amount of money that is charged to a customer fo minute spent on the phone, for a specific year (YYYY)

US States Table :

Column Name	Description
StateCD	2-letter state code
Name	Name of the state
Region	US region name (East, West, etc.)

- Three of them are Calls details : 1 for year 2018 , one other for year 2019 and the last one for year 2020 :

Column Name	Description
CallTimestamp	Date & Time of call
Call Type	Type of call
EmployeeID	Employee unique ID
CallDuration	Duration of call, in seconds
WaitTime	Wait time, in seconds
CallAbandoned	Was the call abandoned by the customer ? (1

- DATA PIPELINE

STEP 1 : STA
The Staging (STA) is used to store the data as we receive them. No modification will be applied.

STEP 2 : ODS
The Operational Data Store (ODS) serves as the primary location for data cleansing and normalization. Within this area, all necessary modifications are applied — including merging datasets, adding, or splitting columns, adjusting data types, and more. If any data fails to meet the set quality standards, they will be recorded in a file called "Technical_Rejects". Subsequently, all these technical rejections will be consolidated in an "Administration" (ADM) file.

STEP 3 : DWH

The Data Warehouse (DWH) is the final step. It will store all the original data that have been cleaned and restructured. It will be organized like a "Star Schema" , with one Fact Table ("Calls" in this project) and several dimension tables.

- **It is very important to respect the SSIS Convention Names all along the ETL process.**
  They can be found at this direction : https://sqlkover.com/ssis-naming-conventions-2-0/

STEP 1 : STAGING (STA)

As mentioned in the previous section, the 'Staging' Step is used to store all the datasets provided by our customer. We won't make any changes to those datasets; they will be stored as we received them.

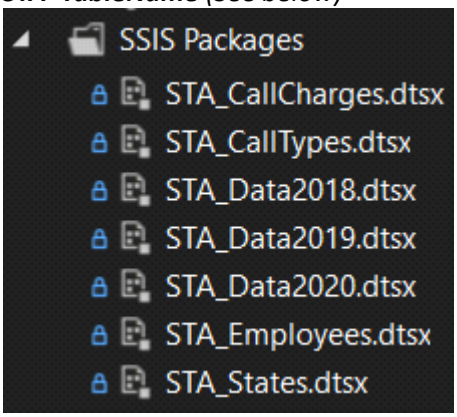
>> Please note that all .csv files have only one tab per sheet.

>> Same process will be applied for each table at this Staging Step

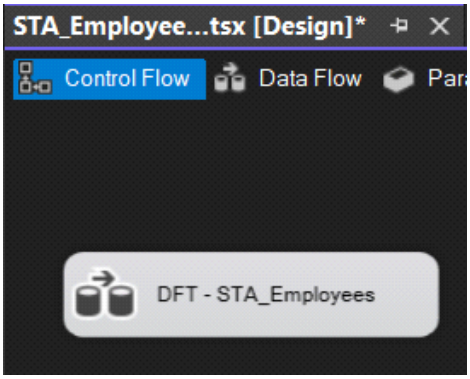
- STA PROCESS

For this section, we will take the 'Employees' Table as an example.

>> **PACKAGE CREATION** :
For each table, we will start by creating a package in SSIS. The package name will contain ***STA+TableName*** *(See below)*

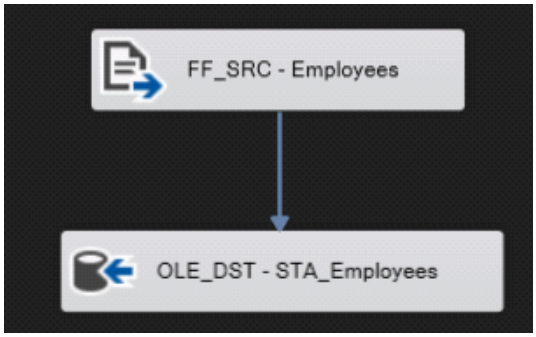


>> **DATA FLOW TASK CREATION** *(Control Flow Tab)* :
For each package, we will start by creating a label "Data Flow Task" in the tab "Control Flow"- The convention name will be ***DFT+PAckageName*** *(See below)*



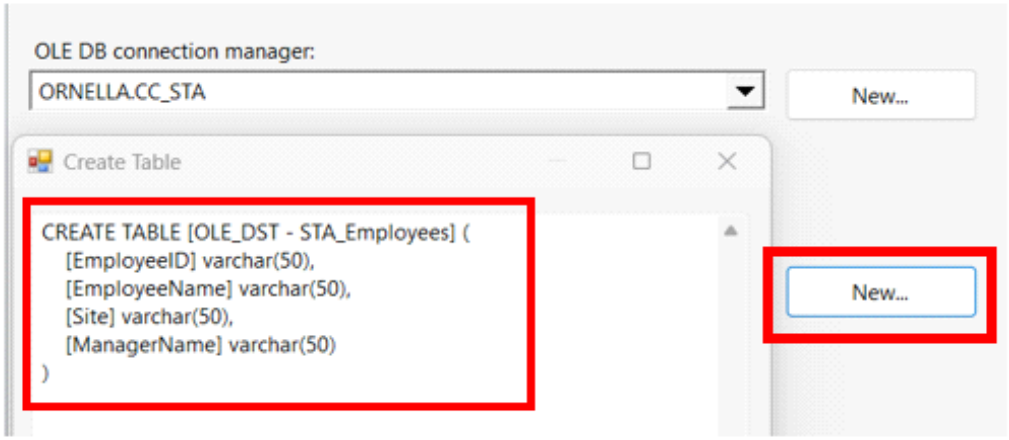
- *We then double-click on it, and we will be redirected to the "Data Flow" tab to start the storage*.

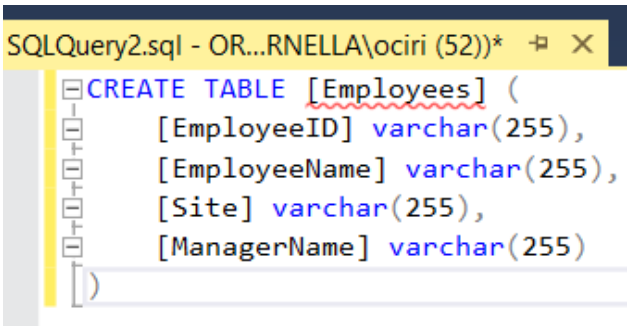
>> **FLAT FILE CREATION** *(Data Flow Tab)* :
Now we are in "Data Flow" Tab. We create the flat source file : it will be used to charge the dataset source (*.csv file as received - no modification made*). The convention name will be **FF-SRC + Table Name**



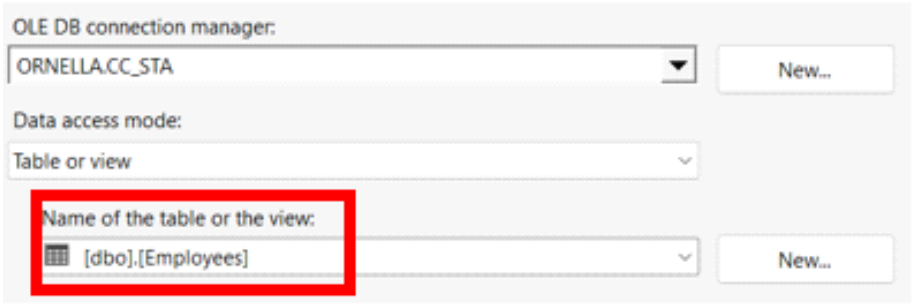
>> **OLE DB DESTINATION CREATION** *(Data Flow Tab) :*
We now drag the 'OLE DB Destination' and we will select and create our table in MS SQL Server
After selecting the correct table name (Here 'STA'), we will copy the table (Click 'New') and create it in SQL Server

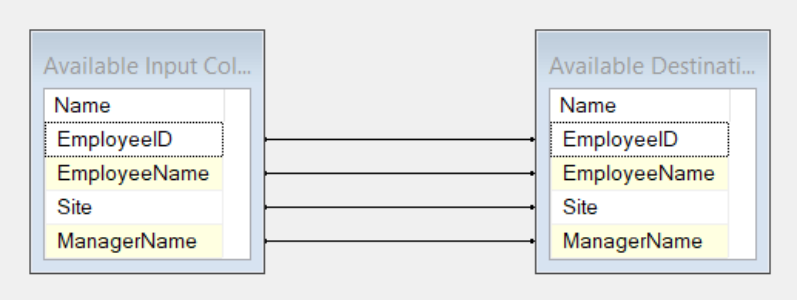
- Table will be renamed (here: Employees) and the maximum value will be changed to '255' to ensure we don't loose any date.



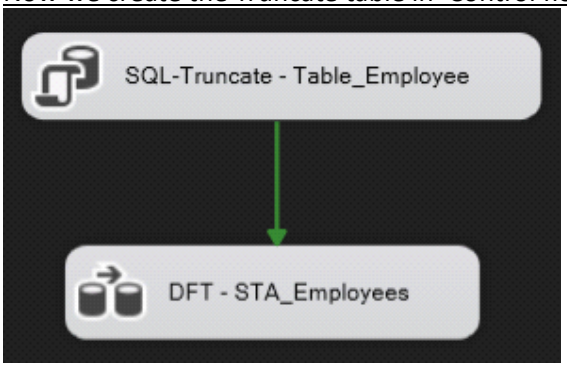


- We are then able to select the table in SSIS and verify the mapping.

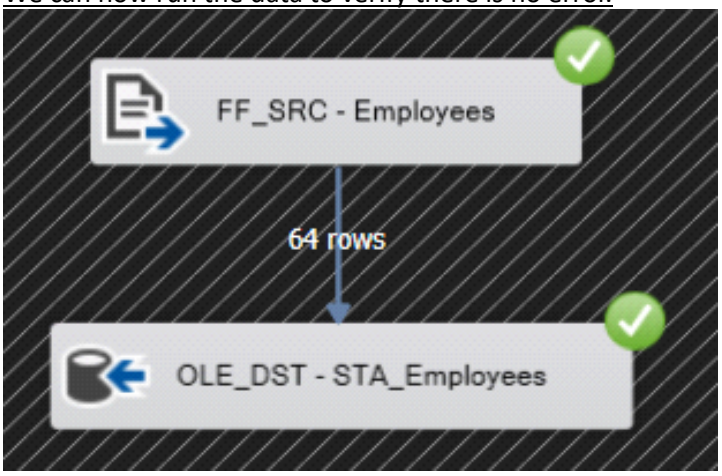




- Now we create the Truncate table in 'Control flow' Tab to avoid accumulation of data



- We can now run the data to verify there is no error.



- SQL SCRIPTS & TABLES EXTRACT

To get the following extract , we used the quick command *"Select the 1000 first lines".*

- **Employees Table**

```
CREATE TABLE [Employees] (
    [EmployeeID] varchar(255),
    [EmployeeName] varchar(255),
    [Site] varchar(255),
    [ManagerName] varchar(255)
)
```

	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

- **StateDC Table**

```
CREATE TABLE [States] (
    [StateCD] varchar(255),
    [Name] varchar(255),
    [Region] varchar(255)
)
```

	StateCD	Name	Region
1	AK	Alaska	West
2	AL	Alabama	South
3	AR	Arkansas	South
4	AZ	Arizona	West
5	CA	California	West

- **Call Types Table**

```
CREATE TABLE [CallTypes] (
    [CallTypeID] varchar(255),
    [CallTypeLabel] varchar(255)
)
```

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

- **Call Charges table**

```
CREATE TABLE [CallCharges] (
    [Call Type ] varchar(255),
    [Call Charges (2018)] varchar(255),
    [Call Charges (2019)] varchar(255),
    [Call Charges (2020)] varchar(255),
    [Call Charges (2021)] varchar(255)
)
```

	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

- **Calls_2018 Table**

```
CREATE TABLE [Data2018] (
    [CallTimestamp] varchar(255),
    [Call Type] varchar(255),
    [EmployeeID] varchar(255),
    [CallDuration] varchar(255),
    [WaitTime] varchar(255),
    [CallAbandoned] varchar(255)
)
```

	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

- **Calls_2019 Table**

```
CREATE TABLE [Data2019] (
    [CallTimestamp] varchar(255),
    [Call Type] varchar(255),
    [EmployeeID] varchar(255),
    [CallDuration] varchar(255),
    [WaitTime] varchar(255),
    [CallAbandoned] varchar(255)
)
```

	CallTimestamp	Call Type	EmployeeID	CallDuration	WaitTime	CallAbandoned
1	8/14/2019 10:31	1	T398672	1126	138	0
2	10/6/2019 9:59	2	K915074	1320	69	0
3	2/11/2019 19:28	2	G586239	1294	10	0
4	1/10/2019 12:10	1	O362956	960	0	1
5	1/4/2019 12:24	3	S274120	1026	0	0

- **Calls_2020 Table**

```
CREATE TABLE [Data2020] (
    [CallTimestamp] varchar(255),
    [Call Type] varchar(255),
    [EmployeeID] varchar(255),
    [CallDuration] varchar(255),
    [WaitTime] varchar(255),
    [CallAbandoned] varchar(255)
)
```

	CallTimestamp	Call Type	EmployeeID	CallDuration	WaitTime	CallAbandoned
1	3/29/2020 13:07	2	U559430	757	0	0
2	3/27/2020 10:00	1	Q921541	35	26	0
3	5/11/2020 11:07	2	I281837	461	0	0
4	9/23/2020 8:13	2	O362956	1133	0	0
5	9/25/2020 9:12	2	J419954	124	17	0

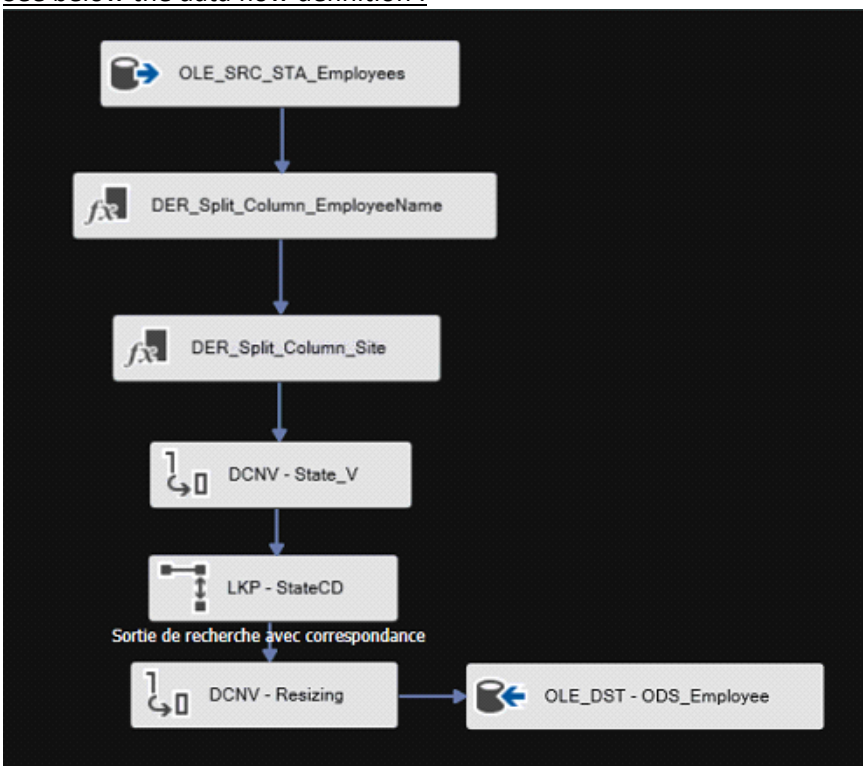
STEP 2 : OPERATIONAL DATA STORE (ODS)

In this section, we will transform the data to make them usable. We will need to clean the data and normalise them, sometimes change their types, create new columns, pivot the table …
Several changes will be operated, and errors will be centralised in an extra file.

- ODS - EMPLOYEES & STATE DC TABLES : MERGING

We decided to merge the *employee table* with the *StateCD table* : The employee table is the only one that has a relation with the StateCD table (Column “Site” vs column “StateCD”).

- See below the data flow definition :



We started by splitting the column EmployeeName into 2 columns : First_name and Last_name. We decided to keep the initial column with the full name of the customer.

First_name	<add as new co...	LEFT(EmployeeName,FINDSTRING(EmployeeName," ",1) - 1)
Last_Name	<add as new co...	TRIM(RIGHT(EmployeeName,LEN(EmployeeName) - FINDSTRING(EmployeeN

Then, to be able to make the relation between the 2 tables, we will need to split the column "Site" from the Employee Table and create two columns : "SiteCity" and 'State".

We will use the following formulas to get the 2 new columns :

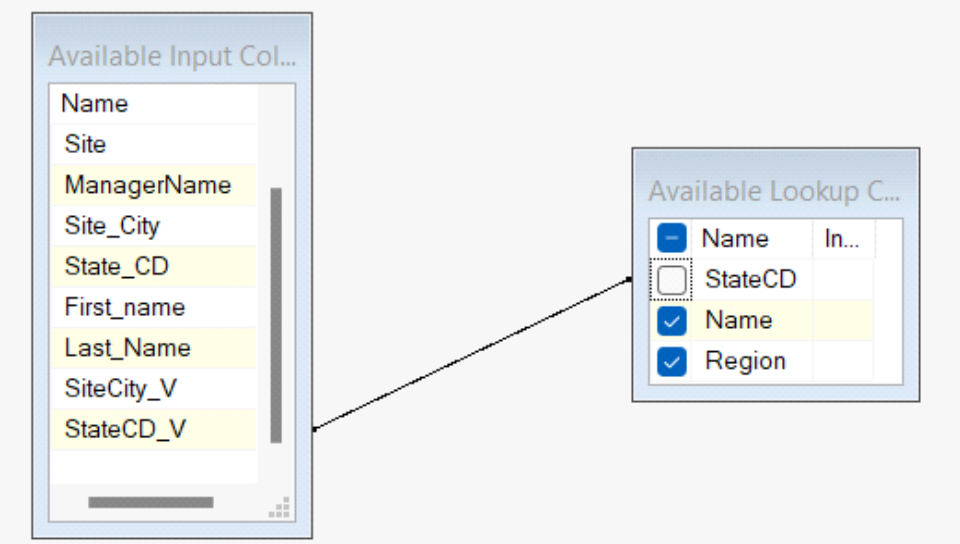
Derived Column Name	Derived Column	Expression	Data Type
SiteCity	<add as new column>	LEFT(Site,FINDSTRING(Site,",",1) - 1)	chaîne Un
StateCD	<add as new column>	TRIM(RIGHT(Site,LEN(Site) - FINDSTRING(Site,",",1)))	chaîne Un

We needed to change the data type of those new columns to be able to match them with the table StateCD *(Otherwise the match between the columns 'StateCD' was not possible)*

- *Chaine Unicode [DT_WSTR] to chaîne [DT_STR]*

Input Column	Output Alias	Data Type	Length
SiteCity	SiteCity_V	chaîne [DT_STR]	255
StateCD	StateCD_V	chaîne [DT_STR]	255

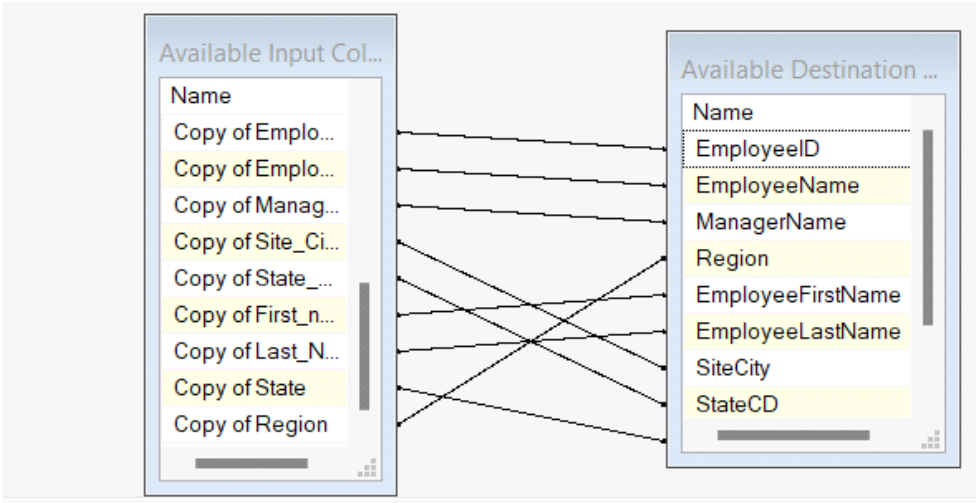
We are now able to create a "LookUp" and merge the two tables. We will link the common field (*StateCD*), and select all the other ones (*Name, Region*).



Next step is to normalize and resize the data :

Input Column	Output Alias	Data Type	Length
EmployeeID	Copy of EmployeeID	chaîne Unicode [DT_W...	10
EmployeeName	Copy of EmployeeName	chaîne Unicode [DT_W...	50
ManagerName	Copy of ManagerName	chaîne Unicode [DT_W...	50
Site_City	Copy of Site_City	chaîne Unicode [DT_W...	50
State_CD	Copy of State_CD	chaîne Unicode [DT_W...	10
First_name	Copy of First_name	chaîne Unicode [DT_W...	50
Last_Name	Copy of Last_Name	chaîne Unicode [DT_W...	50
State	Copy of State	chaîne Unicode [DT_W...	50
Region	Copy of Region	chaîne Unicode [DT_W...	50

We can now create the final mapping and call the first line of the new Employee Table



- Mapping :

Input Column	Destination Column
Copy of EmployeeID	EmployeeID
Copy of EmployeeName	EmployeeName
Copy of First_name	EmployeeFirstName
Copy of Last_Name	EmployeeLastName
Copy of ManagerName	ManagerName
Copy of Region	Region
Copy of Site_City	SiteCity
Copy of State	StateName
Copy of State_CD	StateCD

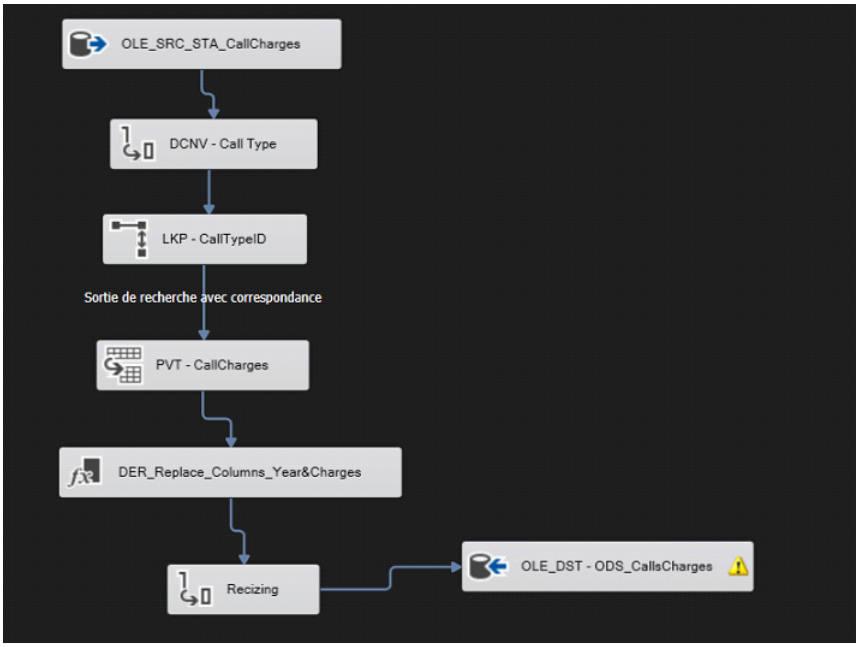
- SQL Script & First lines query :

```
CREATE TABLE [Employee] (
    [EmployeeID] nvarchar(10),
    [EmployeeName] nvarchar(50),
    [EmployeeFirstName] nvarchar(50),
    [EmployeeLastName] nvarchar(50),
    [ManagerName] nvarchar(50),
    [SiteCity] nvarchar(50),
    [StateCD] nvarchar(10),
    [StateName] nvarchar(50),
    [Region] nvarchar(50)
)
```

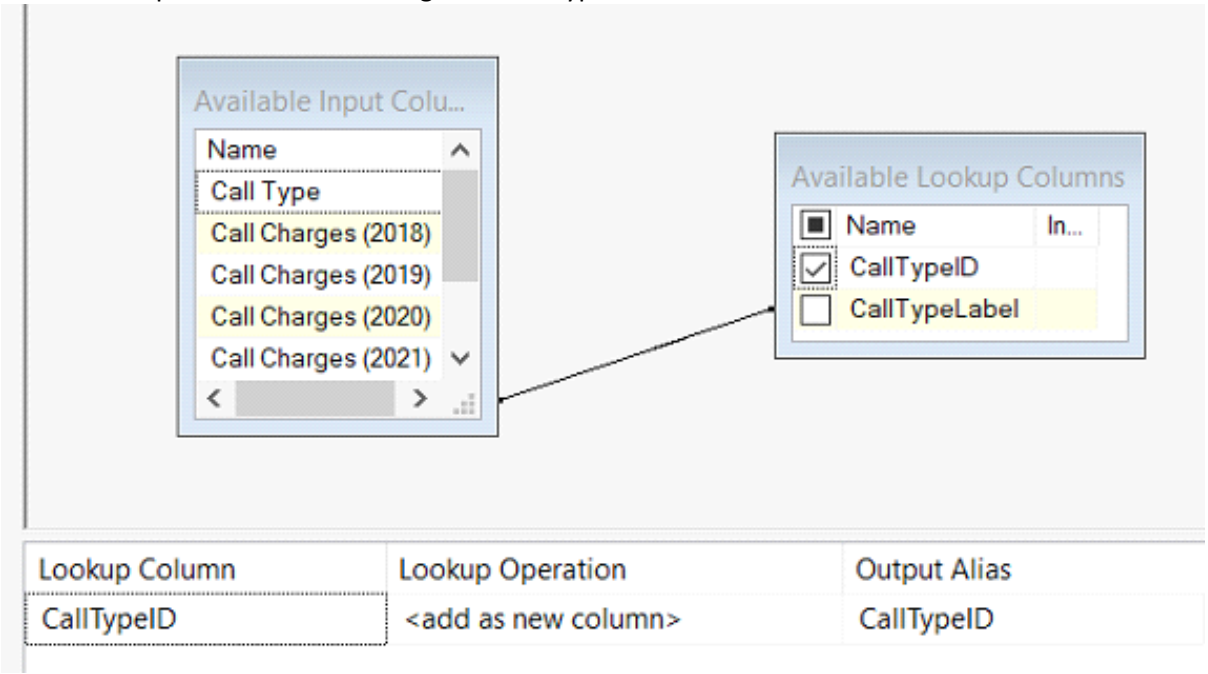
	EmployeeID	EmployeeName	EmployeeFirstName	EmployeeLastName	ManagerName	SiteCity	StateC
1	N772493	Onita Trojan	Onita	Trojan	Deidre Robbs	Spokane	WA
2	F533051	Stormy Seller	Stormy	Seller	Elsie Taplin	Aurora	CO
3	S564705	Mable Ayoub	Mable	Ayoub	Shala Lion	Aurora	CO
4	I281837	Latrisha Buckalew	Latrisha	Buckalew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

- ODS - TYPE & CHARGES TABLES

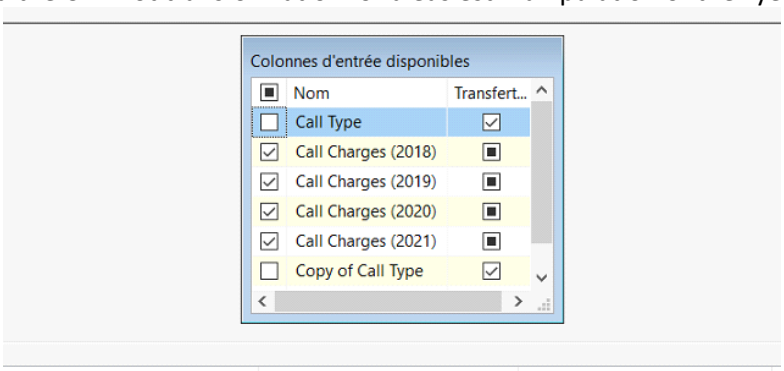
Package :



We decided to Lookup the tables Data Charges et DataType :



We used the UnPivot transformation for a easiest manipulation of the “years” :



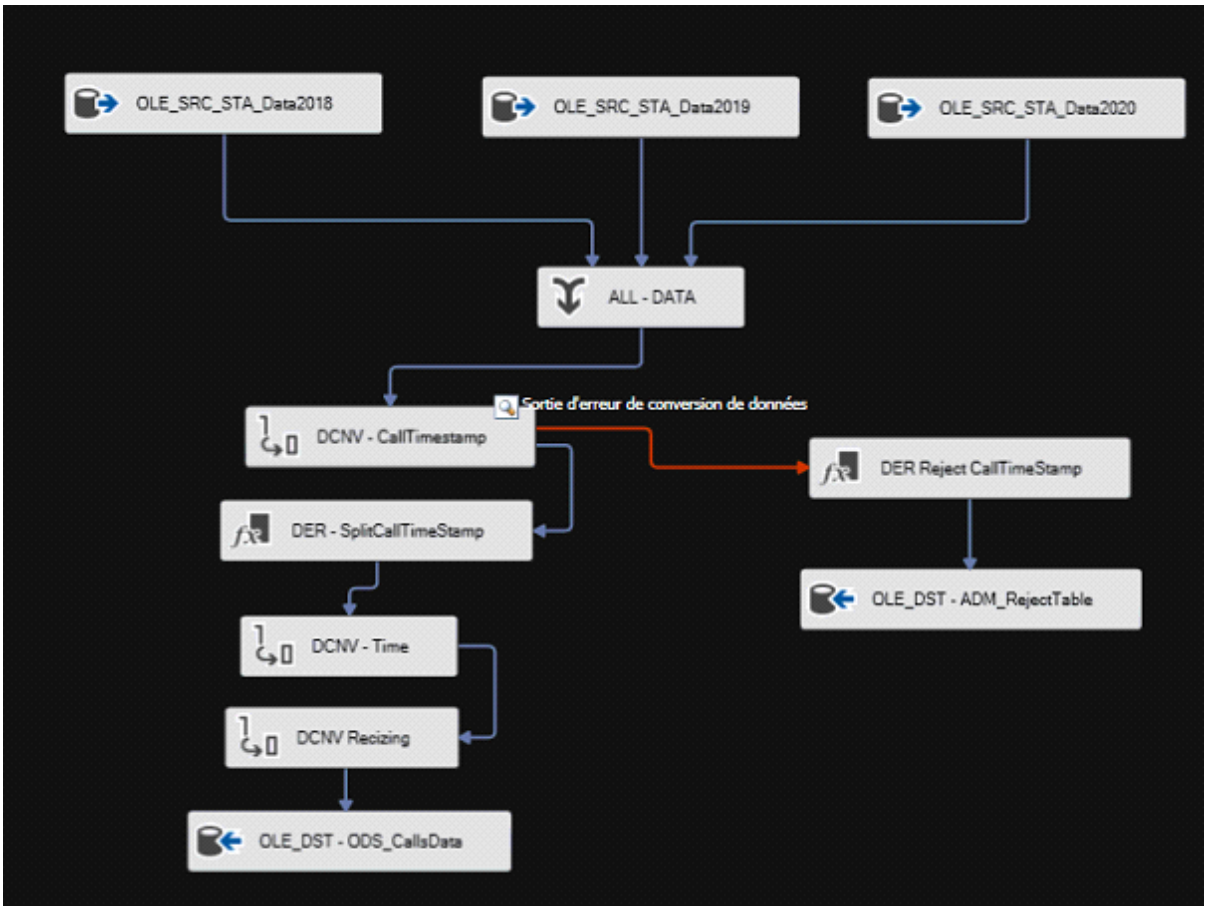
Input Column	Destination Column	Pivot Key Value
Call Charges (2018)	Charges	Call Charges (2018)
Call Charges (2019)	Charges	Call Charges (2019)
Call Charges (2021)	Charges	Call Charges (2021)
Call Charges (2020)	Charges	Call Charges (2020)

Columns “Year” and “charges” have been replaced :

Derived Column Name	Derived Column	Expression
Year	Remplacer 'Year'	REPLACE((RIGHT(Year,5)),")"," ")
Charges	Remplacer 'Charges'	REPLACE(Charges,"/ min"," ")

- ODS - CALLS DATA

  Package :



We have linked all the data (*year 2018, year 2019, year 2020*) in a union ALL

Union All Transformation Editor

Configure the properties used to merge multiple inputs into one output by creating mappings between columns.

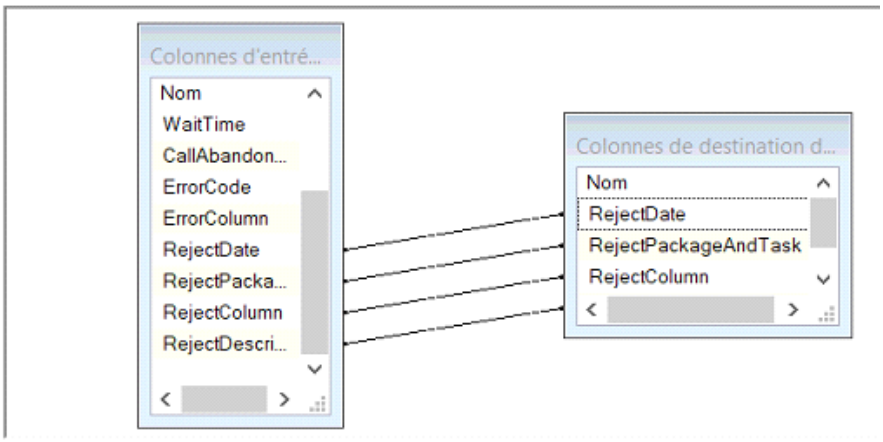
Output Column Name	Unir tout entrée 1	Entrée Union All 1
CallTimestamp	CallTimestamp	CallTimestamp
Call Type	Call Type	Call Type
EmployeeID	EmployeeID	EmployeeID
CallDuration	CallDuration	CallDuration
WaitTime	WaitTime	WaitTime
CallAbandoned	CallAbandoned	CallAbandoned

A reject table has been created for the CallTimeStamps columns:

Derived Column Name	Derived Column	Expression	Data Type
RejectDate	<ajouter comme nou...	GETDATE()	horodateur base de
RejectPackageAndTask	<ajouter comme nou...	(DT_WSTR,100)@[System::PackageName] + " "...	chaîne Unicode [DT_
RejectColumn	<ajouter comme nou...	CallTimestamp	chaîne Unicode [DT_
RejectDescription	<ajouter comme nou...	"The value " + CallTimestamp + " is not a vali...	chaîne Unicode [DT_

This reject tab is linked to the ADM Tab where a '**Technicals reject** " file has been created :

```
CREATE TABLE[TechnicalsReject](
    [RejectDate] [datetime] NULL,
    [RejectPackageAndTask] [nvarchar](201) NULL,
    [RejectColumn] [nvarchar](255) NULL,
    [RejectDescription] [nvarchar](286) NULL
)
```



Colonne d'entrée	Colonne de destination
RejectDate	RejectDate
RejectPackageAndTask	RejectPackageAndTask
RejectColumn	RejectColumn
RejectDescription	RejectDescription

We splitted the CallTimeStamps into a **Date & Time** columns for a better visualisation :

Derived Column Name	Derived Column	Expression	Da
Date	<ajouter comme nou...	(DT_DBDATE)CallTimestamp	da
Time	<ajouter comme nou...	(DT_DBTIME)CallTimestamp	he

- SQL Script & First lines query :

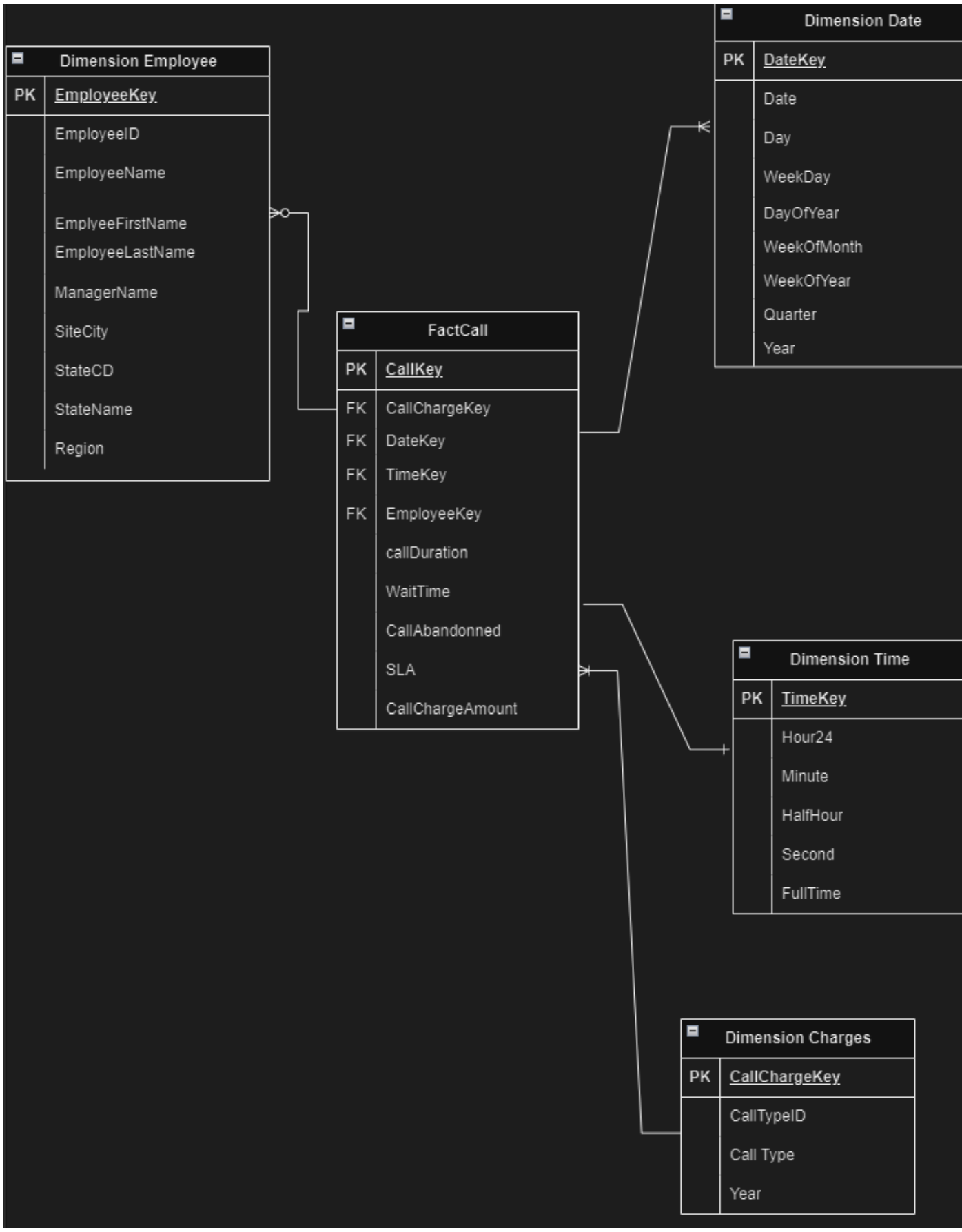
```
CREATE TABLE [CallsData](
    [CallTimestamp] [nvarchar](50) NULL,
    [Call Type] [int] NULL,
    [EmployeeID] [nvarchar](50) NULL,
    [CallDuration] [int] NULL,
    [WaitTime] [int] NULL,
    [CallAbandoned] [int] NULL,
    [Date] [date] NULL,
    [Time] [time](7) NULL
```

	CallTimestamp	Call ...	EmployeeID	CallDuration	Wai...	Call...	Date	Time
1	2019-09-19 11:41:00	3	M163408	813	4	0	2019-09-19	11:41:00...
2	2019-05-20 16:50:00	2	D774655	32	0	0	2019-05-20	16:50:00...
3	2019-11-06 13:09:00	3	N772493	381	0	0	2019-11-06	13:09:00...
4	2019-05-08 09:25:00	3	E778362	797	8	0	2019-05-08	09:25:00...
5	2019-11-06 11:36:00	3	N772493	1060	28	0	2019-11-06	11:36:00...

STEP 3 : DATA WAREHOUSE (DWH)

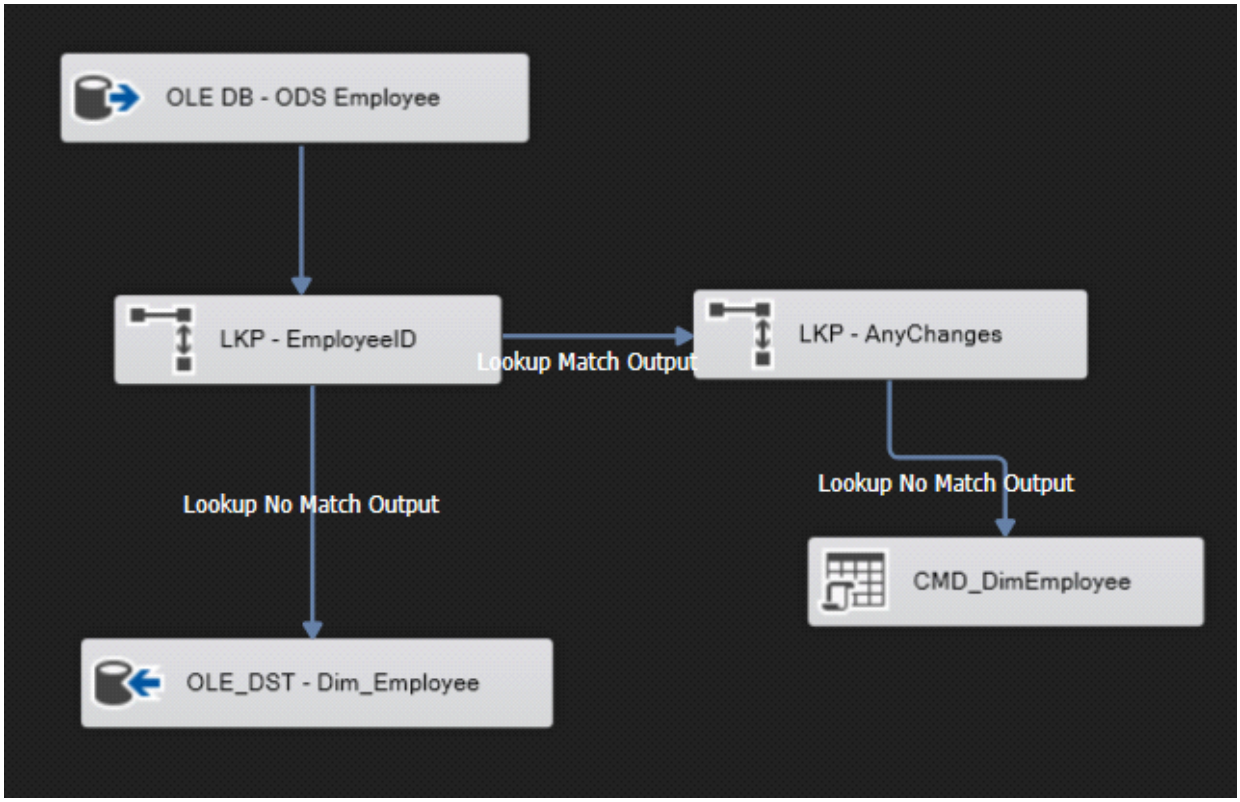
- STAR SCHEMA

In this step, we created a Star Schema to help define the Dimension tables and the Fact table.

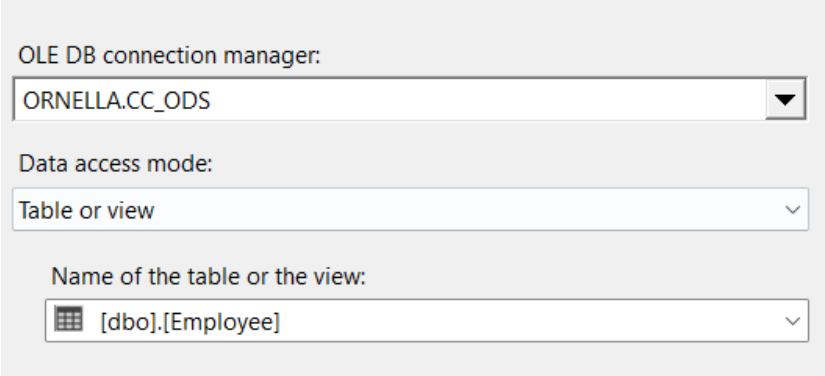

Dimension Employee
PK
EmployeeKey
EmployeeID
EmployeeName
EmplyeeFirstName
EmployeeLastName
ManagerName
SiteCity
StateCD
StateName
Region
FactCall
PK
CallKey
FK
CallChargeKey
FK
DateKey
FK
TimeKey
FK
EmployeeKey
callDuration
WaitTime
CallAbandonned
SLA
CallChargeAmount
Dimension Date
PK
DateKey
Date
Day
WeekDay
DayOfYear
WeekOfMonth
WeekOfYear
Quarter
Year
Dimension Time
PK
TimeKey
Hour24
Minute
HalfHour
Second
FullTime
Dimension Charges
PK
CallChargeKey
CallTypeID
Call Type
Year

- DIM EMPLOYEE TABLE

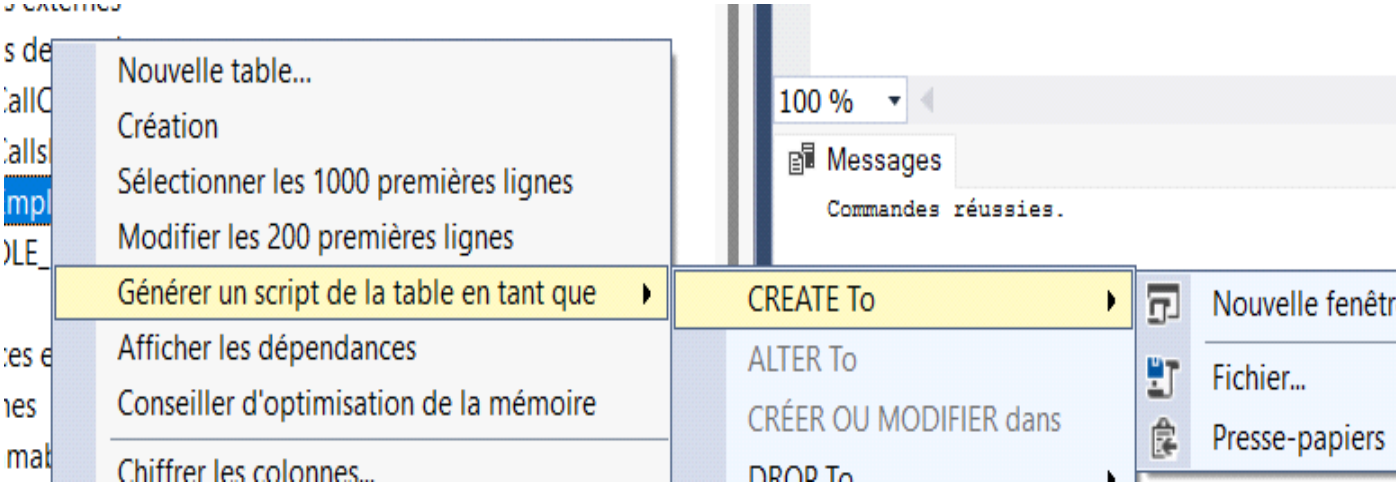
We proceed now with the dimension tables creation, starting with the **DimEmployee**



We start by creating the OLE DB Source by uploading the ODS_Employee table :



Then create the LookUp table : We will created the DimEmployee table in SQL Server from the ODS_Emploee Table :



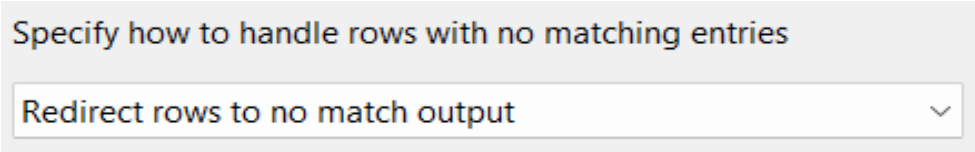
We proceed with the modification of the table :

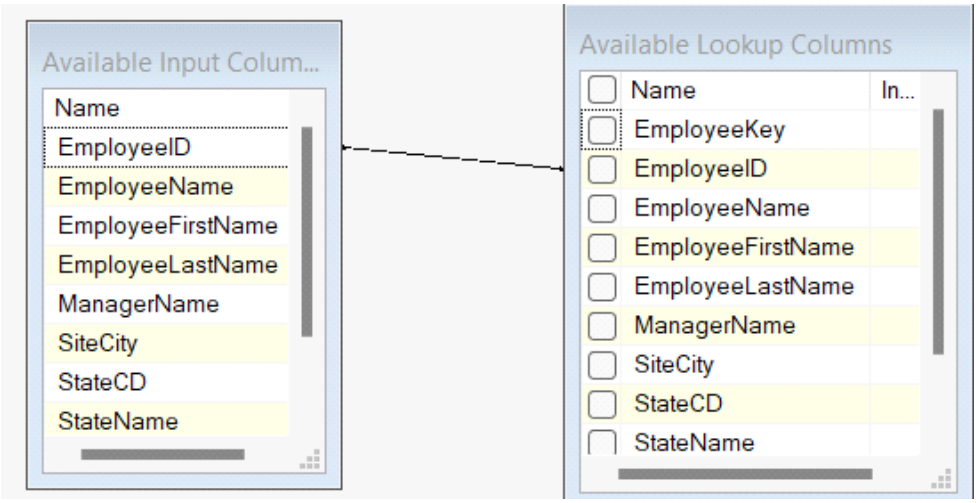
- Change the Data Warehouse
- Rename the table
- Create the table Key (= EmployeeKey)
- Execute + Refresh

```
USE [CC_DWH]
GO

/****** Object:  Table [dbo].[Employee]    Script Date: 15/10/2023
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimEmployee](
    [EmployeeKey] INT PRIMARY KEY IDENTITY (1,1),
    [EmployeeID] [nvarchar](10) NULL,
    [EmployeeName] [nvarchar](50) NULL,
    [EmployeeFirstName] [nvarchar](50) NULL,
    [EmployeeLastName] [nvarchar](50) NULL,
    [ManagerName] [nvarchar](50) NULL,
    [SiteCity] [nvarchar](50) NULL,
    [StateCD] [nvarchar](10) NULL,
    [StateName] [nvarchar](50) NULL,
    [Region] [nvarchar](50) NULL
) ON [PRIMARY]
GO
```

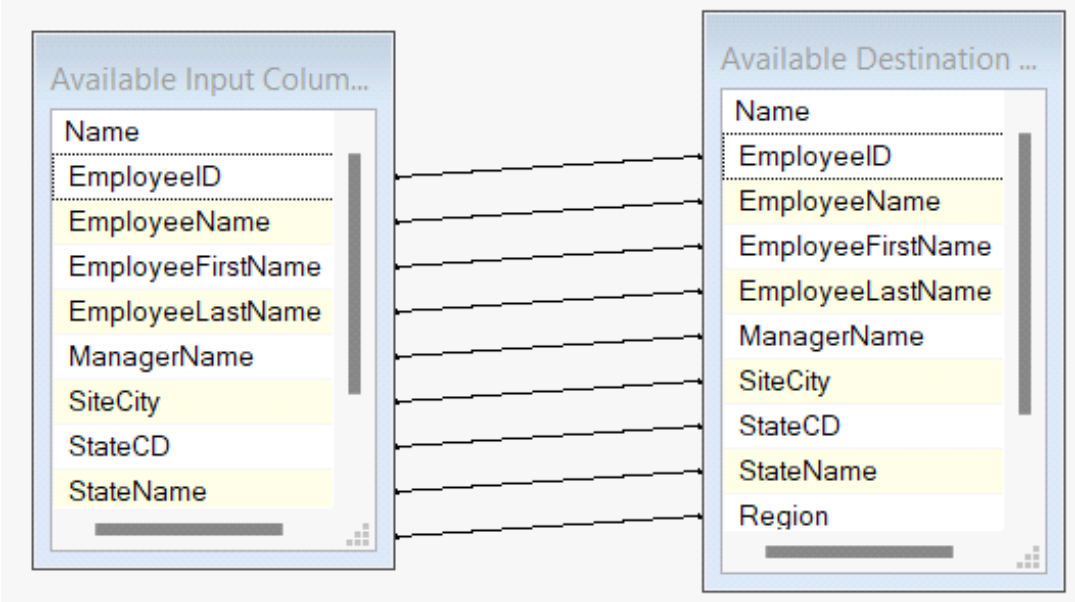
Back to SSIS :

We select "*Redirect row to no match output*" , then select *DWH / dimEmployee Table* - and we link the "EmployeeID"



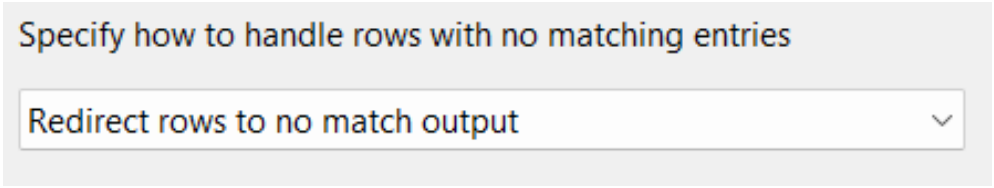


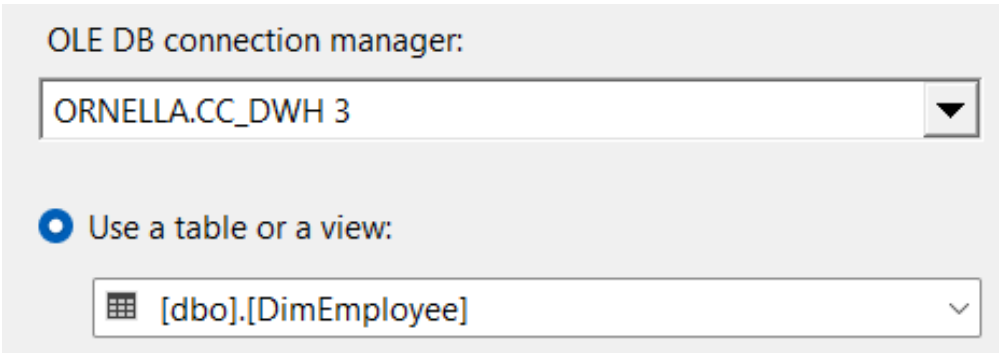
We create the OLE DB Destination and we link the LookUp to it with "*No match Output*" - and we link all the field :

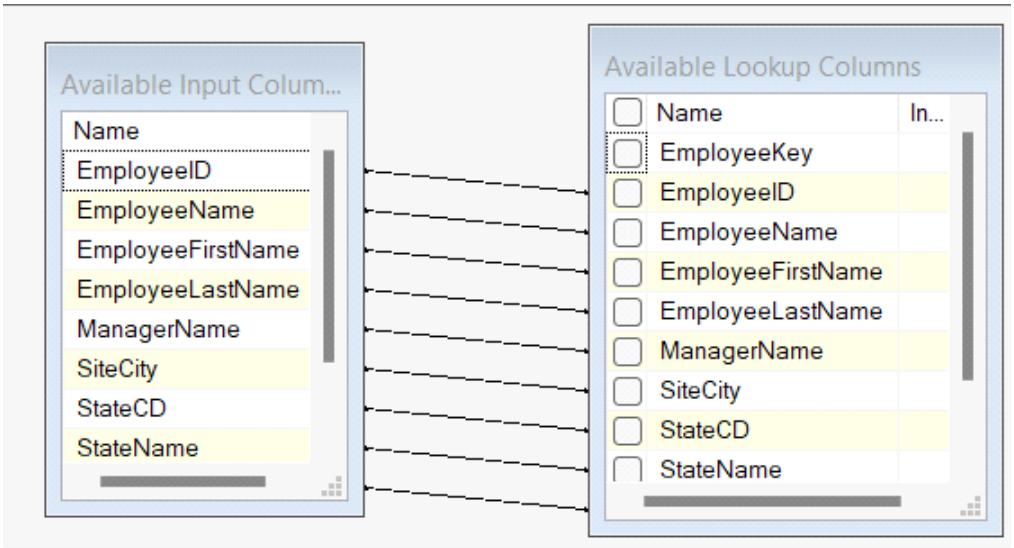


Then we create a new lookUp called “AnyChanges” from the Tab “LKP_EmployeeID”.
This will be useful in case there are new changes in the employees data.

In this lookUp, we still select “*Redirect row to no match output*” , “*DWH / dimEmployee” Table* - and we verify that all fields are correctly linked (Except the EmployeeKey)







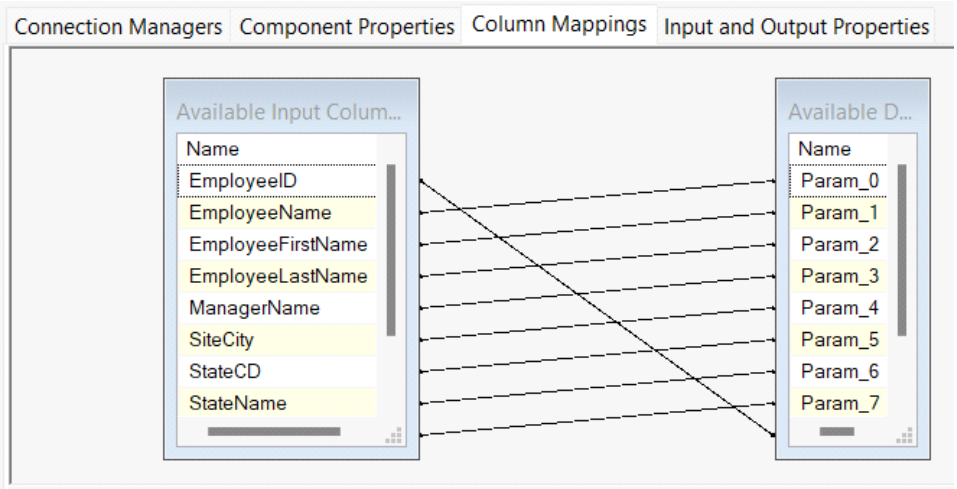
And finally, we create the “Command” Tab : We will create an **update script** to retrieve any changes :

String value:

```
UPDATE [dbo].[DimEmployee]
  SET [EmployeeName] = ?
     ,[EmployeeFirstName] = ?
     ,[EmployeeLastName] = ?
     ,[ManagerName] = ?
     ,[SiteCity] = ?
     ,[StateCD] = ?
     ,[StateName] = ?
     ,[Region] = ?
 WHERE [EmployeeID] = ?
```

SqlCommand | UPDATE [dbo].[DimEmployee] SET [E

We then map the data with the parameter (EmployeeID has to be the last one):



Input Column	Destination Column
EmployeeName	Param_0
EmployeeFirstName	Param_1
EmployeeLastName	Param_2
ManagerName	Param_3
SiteCity	Param_4
StateCD	Param_5
StateName	Param_6
Region	Param_7
EmployeeID	Param_8

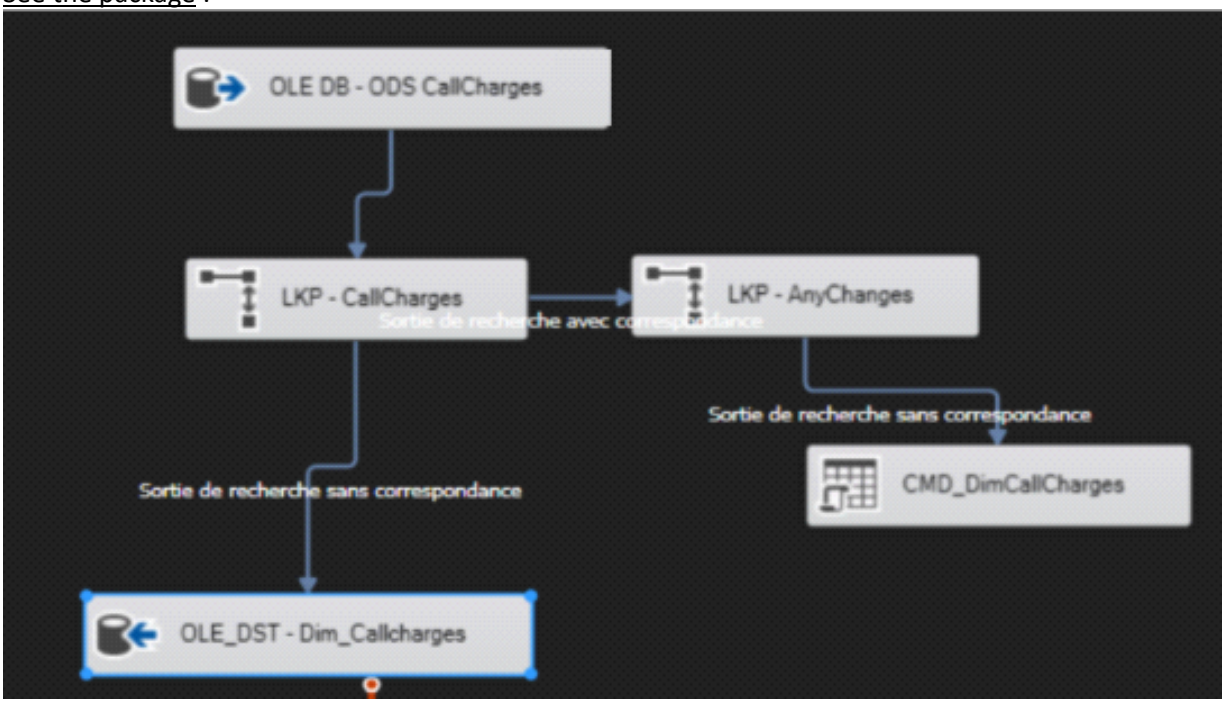
Here is an extract of the DimEmployee :

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

- DIM CHARGES TABLE

For this one, we followed the same steps as per the DimEmployee Table.

See the package :



See below the SQL script :

```
USE [CC_DWH]
GO

/****** Object:  Table [dbo].[CallCharges]    Script Date: 15/10/2023 16:45:57 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[DimCallCharges](
    [Year] [nvarchar](10) NULL,
    [Call Type] [nvarchar](50) NULL,
    [Charges] [nvarchar](10) NULL,
    [CallTypeID] [nvarchar](255) NULL
) ON [PRIMARY]
GO
```

See below an extract of the table :

	CallsChargesKey	CallTypeId	Call Type	Charges	Year
1	25	1	Sales	1.52	2018
2	26	1	Sales	1.56	2019
3	27	1	Sales	1.60	2020
4	28	1	Sales	1.71	2021
5	29	2	Billing	1.2	2018

- DIM DATE TABLE

For this DimDate Table , we will create it directly in the SQL SERVER.

Script :

```
CREATE TABLE [dbo].[DimDate](
    [DateKey] [int] NOT NULL,
    [Date] [date] NOT NULL,
    [Day] [tinyint] NOT NULL,
    [DaySuffix] [char](2) NOT NULL,
    [Weekday] [tinyint] NOT NULL,
    [WeekDayName] [varchar](10) NOT NULL,
    [WeekDayName_Short] [char](3) NOT NULL,
    [WeekDayName_FirstLetter] [char](1) NOT NULL,
    [DOWInMonth] [tinyint] NOT NULL,
    [DayOfYear] [smallint] NOT NULL,
    [WeekOfMonth] [tinyint] NOT NULL,
    [WeekOfYear] [tinyint] NOT NULL,
    [Month] [tinyint] NOT NULL,
    [MonthName] [varchar](10) NOT NULL,
    [MonthName_Short] [char](3) NOT NULL,
    [MonthName_FirstLetter] [char](1) NOT NULL,
    [Quarter] [tinyint] NOT NULL,
    [QuarterName] [varchar](6) NOT NULL,
    [Year] [int] NOT NULL,
    [MMYYYY] [char](6) NOT NULL,
    [MonthYear] [char](7) NOT NULL,
    [IsWeekend] [bit] NOT NULL,
)
```

Extract :

	DateKey	Date	Day	DaySuffix	Weekday	WeekDayName	WeekDayName_Short	WeekDayNar
1	20180101	2018-01-01	1	st	1	lundi	LUN	l
2	20180102	2018-01-02	2	nd	2	mardi	MAR	m
3	20180103	2018-01-03	3	rd	3	mercredi	MER	m
4	20180104	2018-01-04	4	th	4	jeudi	JEU	i

- DIM TIME TABLE

Same for the DimTime Table

Script :

```
CREATE TABLE [dbo].[DimTime](
    [TimeKey] [int] NOT NULL,
    [Hour24] [int] NULL,
    [Hour24ShortString] [varchar](2) NULL,
    [Hour24MinString] [varchar](5) NULL,
    [Hour24FullString] [varchar](8) NULL,
    [Hour12] [int] NULL,
    [Hour12ShortString] [varchar](2) NULL,
    [Hour12MinString] [varchar](5) NULL,
    [Hour12FullString] [varchar](8) NULL,
    [AmPmCode] [int] NULL,
    [AmPmString] [varchar](2) NOT NULL,
    [Minute] [int] NULL,
    [MinuteCode] [int] NULL,
    [MinuteShortString] [varchar](2) NULL,
    [MinuteFullString24] [varchar](8) NULL,
    [MinuteFullString12] [varchar](8) NULL,
    [HalfHour] [int] NULL,
    [HalfHourCode] [int] NULL,
    [HalfHourShortString] [varchar](2) NULL,
    [HalfHourFullString24] [varchar](8) NULL,
    [HalfHourFullString12] [varchar](8) NULL,
    [Second] [int] NULL,
    [SecondShortString] [varchar](2) NULL,
    [FullTimeString24] [varchar](8) NULL,
    [FullTimeString12] [varchar](8) NULL,
    [FullTime] [time](7) NULL
    )
```

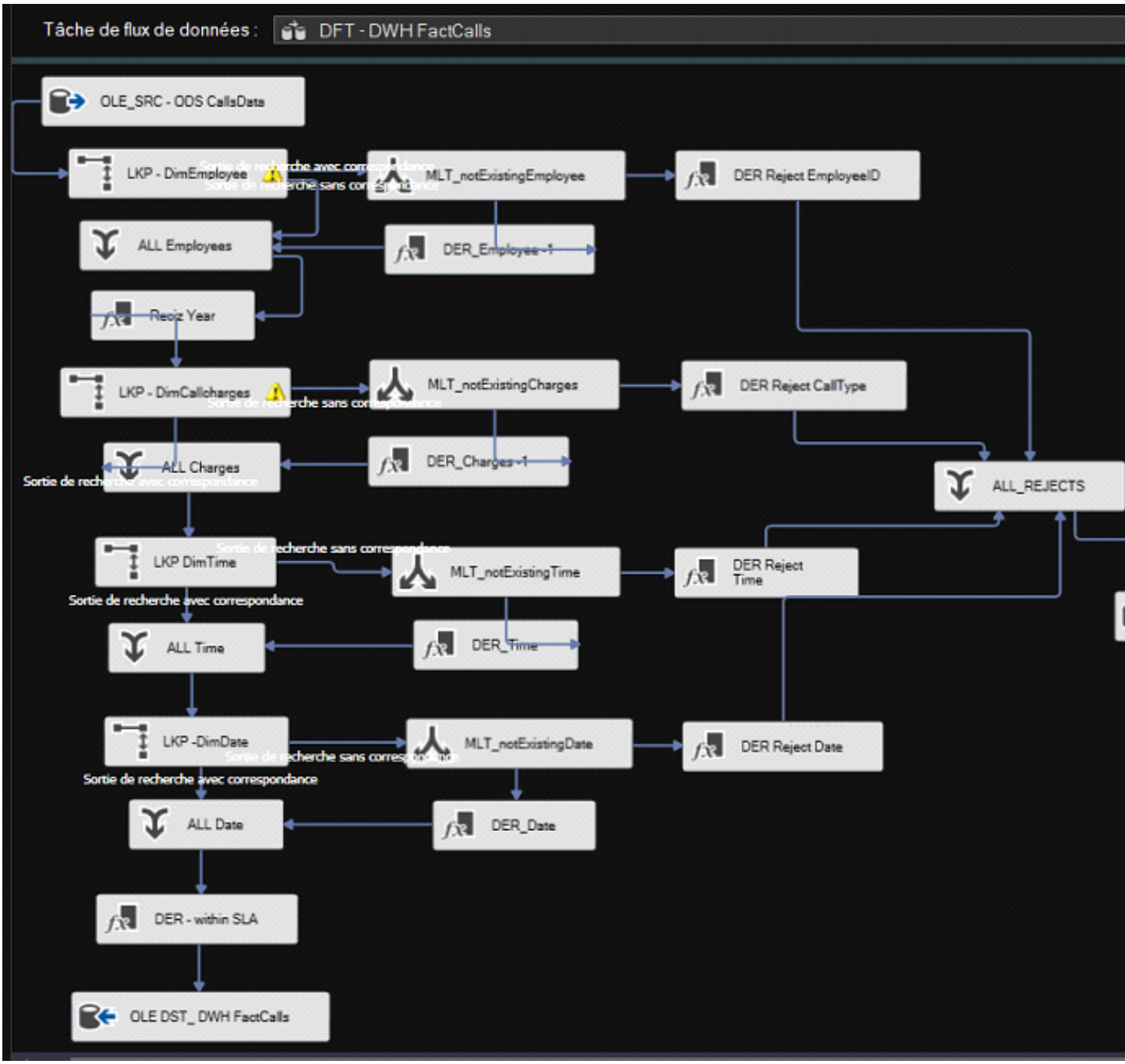
Table :

	TimeKey	Hour...	Hour...	Hour24...	Hour24...	Hour12	Hour...	Hour12...	Hour12Ful...	Am...	A
1	0	0	00	00:00	00:00:00	0	00	00:00	00:00:00	0	A
2	1	0	00	00:00	00:00:00	0	00	00:00	00:00:00	0	A
3	2	0	00	00:00	00:00:00	0	00	00:00	00:00:00	0	A
4	3	0	00	00:00	00:00:00	0	00	00:00	00:00:00	0	A
5	4	0	00	00:00	00:00:00	0	00	00:00	00:00:00	0	A

- FACT CALLS TABLE

We now create the Fact table where we will centralize all the technical Key and retrieve all the errors.

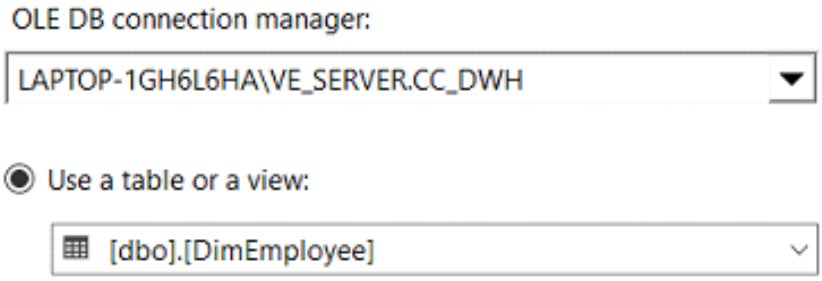
See the package :

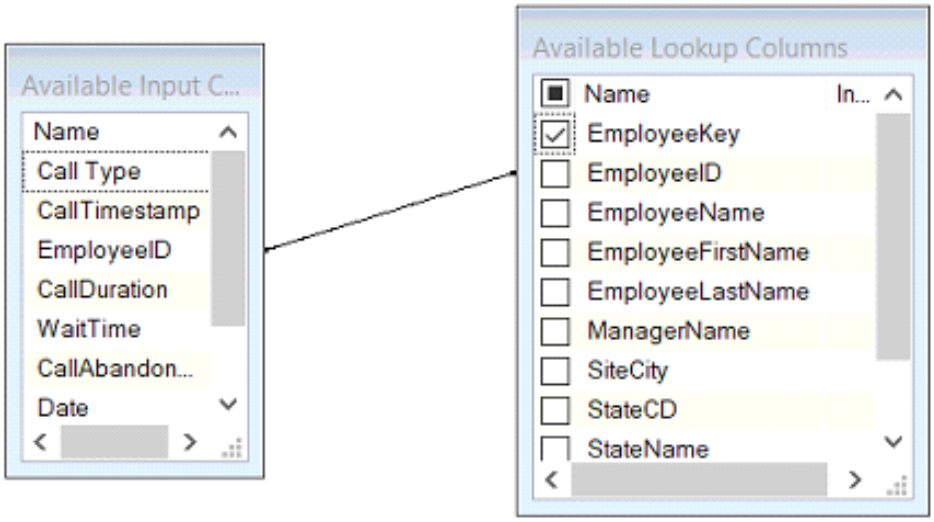


First, We will create a LookUp to retrieve each Dimension Tables and we will link the Primary Key and check the Table Key

=> Four in total : Dim Employee/ DimCharges/ DimDate/ DimTime) .

Example with DimEmployee Table (Same process for each DimTab) :





Then we will create a “Within SLA” tab, where we will retrieve all the calls that have been answered in the first 35 sec : a “1” will appear when the wait time is respected, a “0” when it is not.

Derived Column Name	Derived Column	Expression
within SLA	<ajouter comme nou...	WaitTime < 35 ? 1 : 0

Once all the LookUp and the Within SLA columns are created, we create the destination OLE DB and the FactCall Table in SQL Server - The table retrieve all the tableKeys :

```
CREATE TABLE [dbo].[FactCall](
[CallKey] [int] IDENTITY(1,1) NOT NULL,
    [Call Type] [int] NULL,
    [EmployeeKey1] [int] NULL,
    [CallsChargesKey1] [int] NULL,
    [DateKey] [int] NULL,
    [TimeKey] [int] NULL,
    [Year(Date)] [nvarchar](20) NULL,
    [CallDuration] [int] NULL,
    [WaitTime] [int] NULL,
    [within SLA] [int] NULL,
    [CallAbandoned] [int] NULL,


)
```

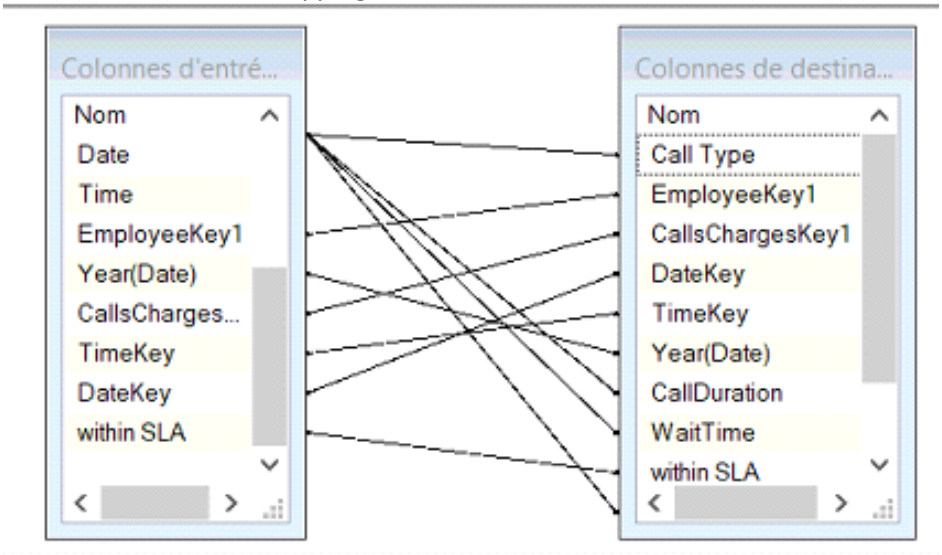
SQL Query :

Résultats | Messages

	CallKey	Call Type	EmployeeKey1	CallsChargesKey1	DateKey	TimeKey	Year(Date)	CallDuration	WaitTime	within
1	593851	3	12	34	20190919	114100	2019	813	4	1
2	593852	2	59	30	20190520	165000	2019	32	0	1
3	593853	3	1	34	20191106	130900	2019	381	0	1
4	593854	3	62	34	20190508	92500	2019	797	8	1
5	593855	3	1	34	20191106	113600	2019	1060	28	1
6	593856	2	42	30	20190104	184000	2019	728	3	1
7	593857	1	32	26	20190701	104000	2019	532	0	1
8	593858	2	41	30	20191027	155200	2019	1358	26	1
9	593859	1	62	26	20190821	112000	2019	272	19	1
10	593860	2	25	30	20190816	83100	2019	1084	12	1

Exécution de requête réussie. LAPTOP-1GH6L6HA\VE_SERVER (... LAPTOP-1GH6L6HA\A

In SSIS , we check the mapping :



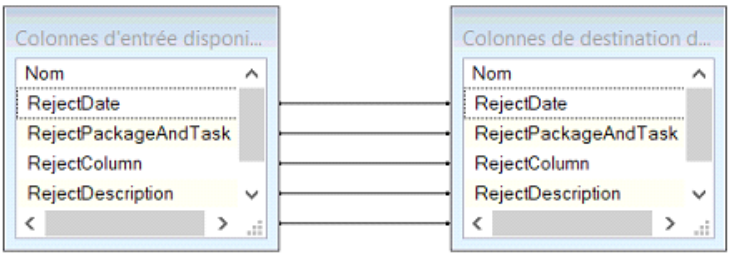
Each LookUp tab will have his reject table. An intermediate “Multidiffusion” Tab , will be created to redirect all the error to the “Reject table” and then re-inject them to the flow specifying “-1”

Example with DimEmployee Table (Same process for each DimTab) :

Derived Column Name	Derived Column	Expression	Data Type
RejectDate	<ajouter comme nou...	GETDATE()	horodateur b
RejectPackageAndTask	<ajouter comme nou...	(DT_WSTR,100)@[System::PackageName] + " "...	chaîne Unico
RejectColumn	<ajouter comme nou...	EmployeeID	chaîne Unico
RejectDescription	<ajouter comme nou...	"The value " + EmployeeID + " is not a valid [...	chaîne Unico

All the rejects will be grouped in an Aggregate Tab - A “Functional Rejects” Table will be created in the ADM file :


```
CREATE TABLE "FunctionalRejects" (
    "RejectDate" datetime,
    "RejectPackageAndTask" nvarchar(201),
    "RejectColumn" nvarchar(50),
    "RejectDescription" nvarchar(90),
    "nb_rejects" numeric(20,0)
)
```



Colonne d'entrée	Colonne de destination
RejectDate	RejectDate
RejectPackageAndTask	RejectPackageAndTask
RejectColumn	RejectColumn
RejectDescription	RejectDescription
COUNT ALL	nb_rejects

**** USE CASE ****

Once we have deployed the data warehouse, we can start to query it for making analysis reports. For instance, we can generate a view that can then be used in a BI tool to report on Call monitoring by employee

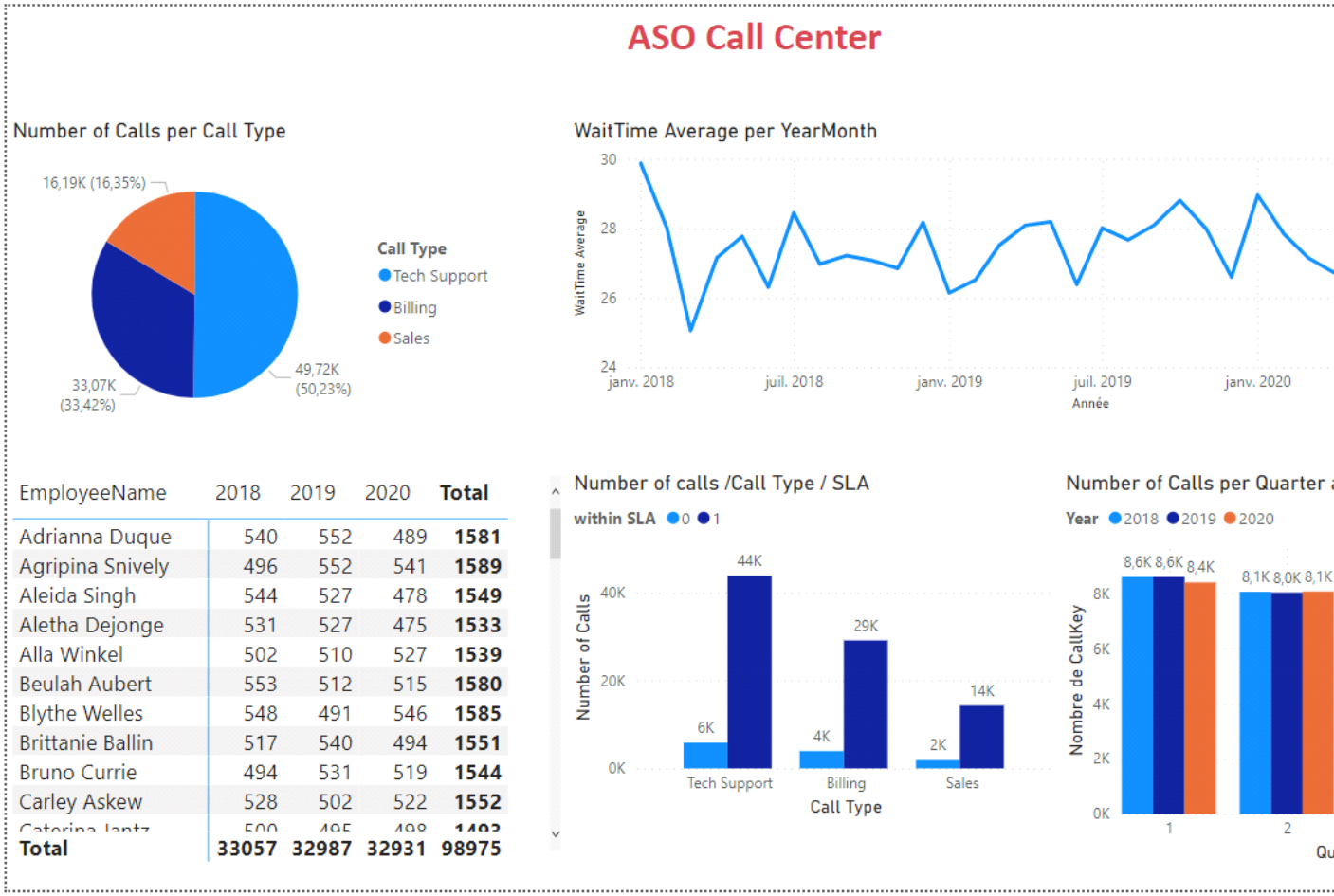
The query aggregates call information by employee, calculating the total number of calls, the number of calls with SLA, average call duration, and average call waiting time, all grouped by employee, state and year.

```sql
Create View AG_CALLS as
   select EmployeeName,StateName,count (CallKey) Number_of_calls
   ,sum (case when FT.[within SLA]=1 then 1 else 0 end ) as Calls
   ,sum(CallDuration)/count (CallKey) as call_average,
   sum(WaitTime)/count (CallKey) as call_wait_Time_Average,
Year

from

 DimEmployee  DimEmp
left join FactCall FT on (FT.EmployeeKey1=DimEmp.EmployeeKey)
left join DimDate DT on (DT.DateKey=FT.DateKey)
group by EmployeeName,StateName,Year
order by EmployeeName
```

Here is the first ten lines of the results of the facts view “AG_CAlls”:

EmployeeName	StateName	Number_of_calls	Calls_with_SLA	call_average	call_wait_Time_Av
Adrianna Duque	Washington	552	500	762	23
Adrianna Duque	Washington	489	439	751	24
Adrianna Duque	Washington	540	471	713	30
Agripina Snively	Colorado	541	488	725	25
Agripina Snively	Colorado	552	491	760	26
Agripina Snively	Colorado	496	437	779	28
Aleida Singh	Colorado	478	419	740	27
Aleida Singh	Colorado	544	483	753	25
Aleida Singh	Colorado	527	465	747	26
Aletha Dejonge	Colorado	527	454	732	29

And we have made a power bi rapport for our CallCenter using our DWH database



CONCLUSION

In summary, the establishment of our data warehouse for the call center has been a crucial step in improving our understanding of the business operations. However, this project has not been without its challenges, especially in the use of SSIS, the tool we used for our ETL processes. One of the main difficulties was harmonizing data types from various sources, which required a significant effort to standardize and clean the data.

We also encountered challenges during data transformation with SSIS, particularly when splitting columns for more efficient use. This demanded meticulous error correction during the execution of the ETL process.

Furthermore, we initially considered creating a "Reject Table" to manage rejected data during the splitting of the "EmployeeName" and "Site" columns in ODS_Employees. However, despite our efforts, we were unable to make it function satisfactorily, leading to its removal.

Despite these obstacles, we successfully implemented a functional data warehouse that now allows us to analyze call center activity in greater depth. These challenges have deepened our understanding of best data management practices, especially with SSIS, and have helped us better comprehend the implications of data collection, transformation, and loading.