PROJECT REPORT

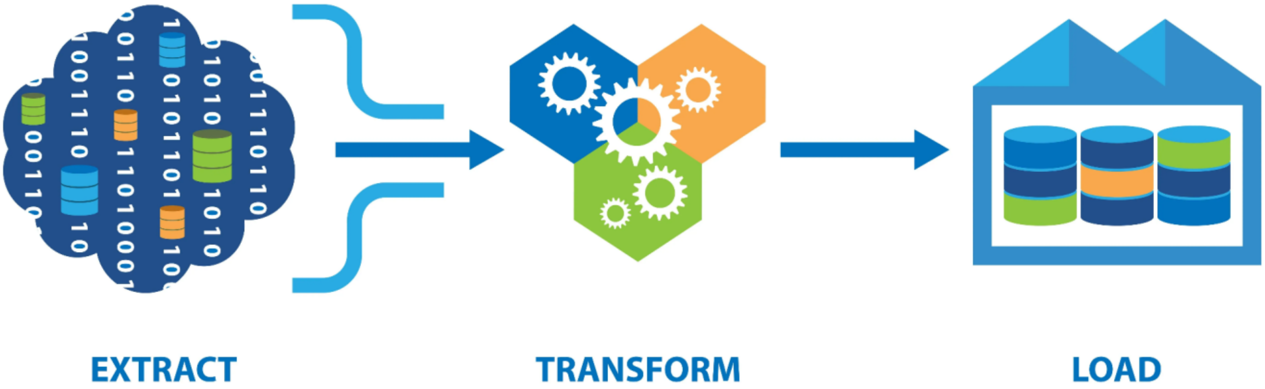
CALL CENTER ANALYSIS

🎧

Cohort: A23

Course: Data Warehousing & ETL

Instructor: Mr. Nelson Gabriel Lopez Chirinos



Team

Ken CHEVALIER,	DA	Sophia	ken.chevallier@edu.dsti.institute
Archana KHANDELWAL,	DA	Paris	archana.khandelwal@edu.dsti.institute
Chakib DOUADI	DE	Paris,	chakib.douadi@edu.dsti.institute
Alexandre BENAMENYO	DA	Paris,	alexandre.benamenyo@edu.dsti.institute
Longyin COUPPEY	DA	Paris,	longyin.couppey@edu.dsti.institute

Table of contents

1. Introduction

To be able to use and extract value from available data, it first needs to be integrated into an IT system. This imply that all the data coming from various sources need 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 historical data of a call center in the US. For this, we will use SQL Server and SSIS.

1.1 Working Environment and Tools

The table outlines the working environment and tools utilized in the project.

SQL Server Management Studio	To create tables and query the data warehouse
VisualStudio 2022 Community (with SSDT, SSIS)	To create each package for data warehousing
Git	To collaborate and manage versioning
SQL Server Configuration Manager	To create SQL Server alias

1.2 Principles of Modeling and Design Choices

For the modeling of our ETL process, we will follow the traditional architecture of data warehouse, with 3 steps:

- STA: Staging

- ODS: Operational Data Storage

- DWH: Data Warehouse

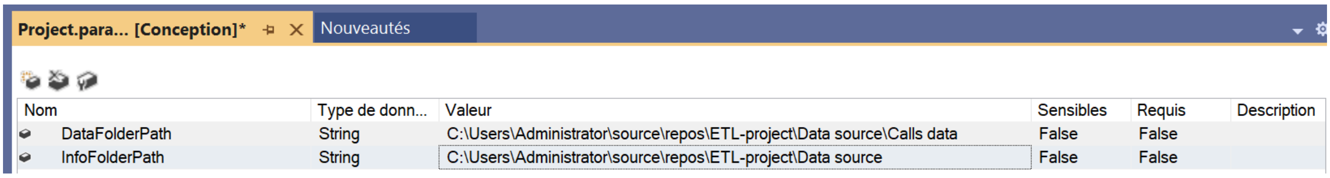
1.3 Data

The data for this project consists of several .csv files representing the general information of the company (employee list, call types) and the details of the calls handled (organized by year).

- Call Charges.csv: contains charging rates per year and call type.
- Call Types.csv: contains list of call types (Sales, Billing, Tech Support).
- Employees.csv: list of company employees.
- US States.csv: list of the 52 US States with states Code Table and Region.
- Data 2018.csv: lists all the calls handled on 2018 with several details (callTimestamp, call type, employeeID, waitingTime).
- Data 2019.csv: same for 2019.
- Data 2020.csv: same for 2020.

To make sure the project in Visual Studio can be reproduced by anyone we defined project variables for the folders containing the csv, so that one would need to just modify these variables to run the project:

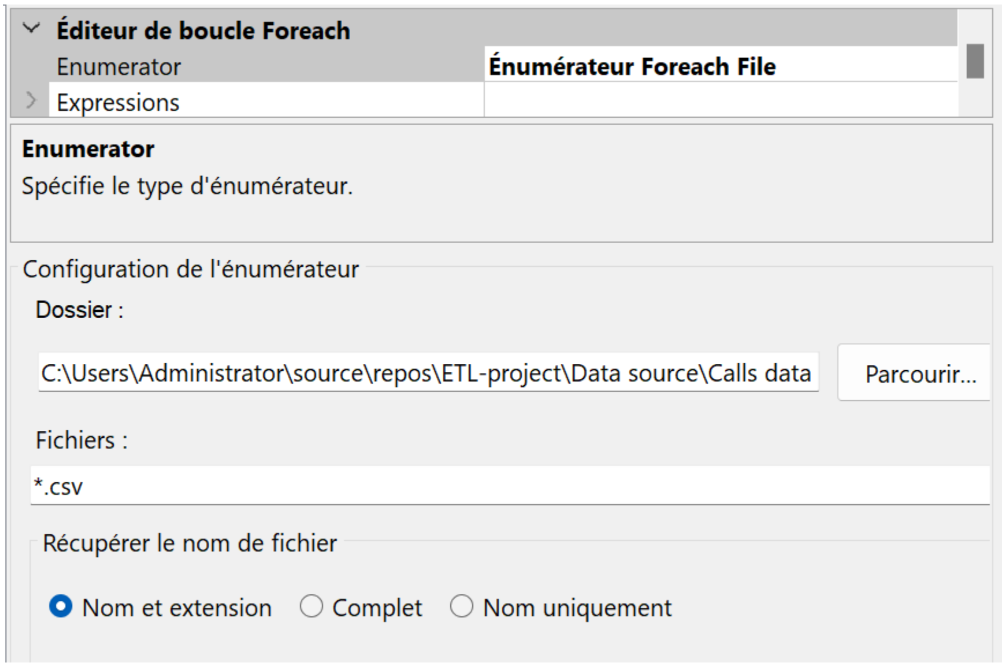
- InfoFolderPath: folder containing Employees, Call Types, Call Charges and US States.
- DataFolderPath: folder containing Data 2018, Data 2019 and Data 2020.



Nom	Type de donn...	Valeur	Sensibles	Requis	Description
DataFolderPath	String	C:\Users\Administrator\source\repos\ETL-project\Data source\Calls data	False	False	
InfoFolderPath	String	C:\Users\Administrator\source\repos\ETL-project\Data source	False	False	

When a user clones the project repository, the data files will be retrieved and stored under C:\Users\<user_name>\source\repos\ETL-project\Data source (above example is for Administrator), so the values need to be updated accordingly.

Additionally, the STA phase contains a For Each Loop Container, which needs a proper file path to be configured, so this value also may need to be modified.


Éditeur de boucle Foreach

Enumerator: **Énumérateur Foreach File**

Expressions

Enumerator

Spécifie le type d'énumérateur.

Configuration de l'énumérateur

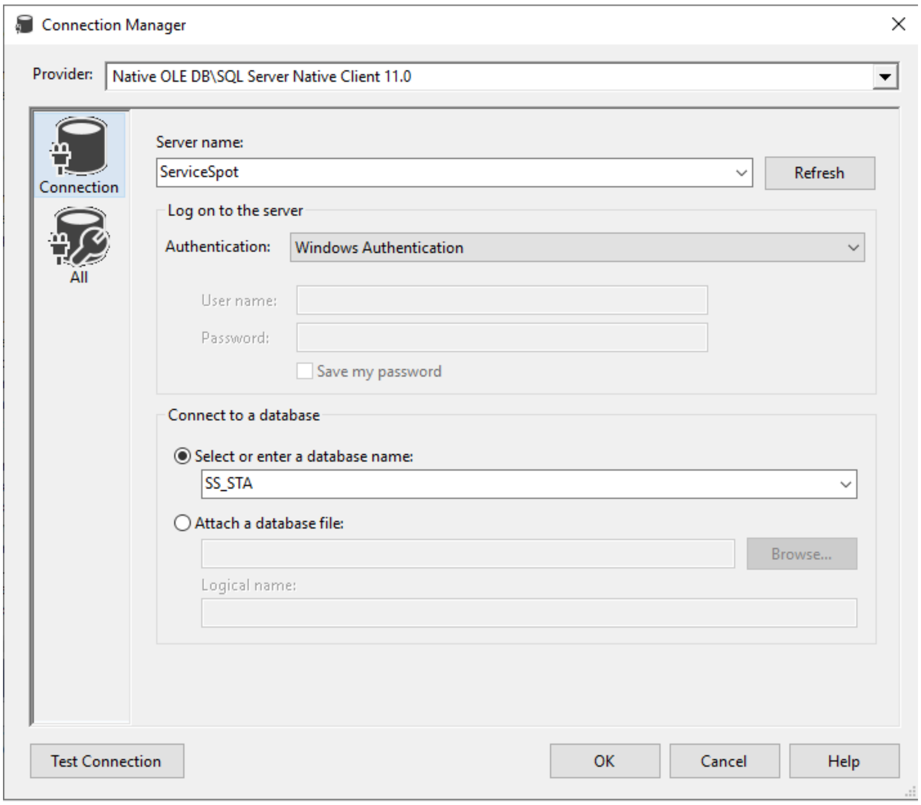
Dossier : C:\Users\Administrator\source\repos\ETL-project\Data source\Calls data

Fichiers : *.csv

Récupérer le nom de fichier

● Nom et extension ○ Complet ○ Nom uniquement

Finally, to be able to use anyone’s own SQL Server on local PC, we defined an alias, “ServiceSpot”, to our local SQL Server.



Connection Manager

Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: ServiceSpot

Log on to the server

Authentication: Windows Authentication

User name:

Password:

□ Save my password

Connect to a database

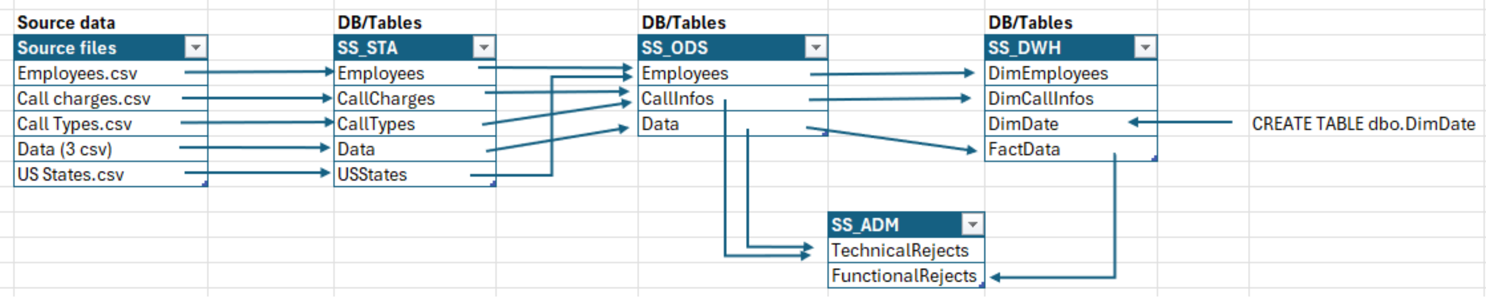
● Select or enter a database name: SS_STA

○ Attach a database file:

Logical name:

1.4 Pipeline design

The data pipeline is designed to be as following:



Source data
Source files
Employees.csv
Call charges.csv
Call Types.csv
Data (3 csv)
US States.csv

DB/Tables
SS_STA
Employees
CallCharges
CallTypes
Data
USStates

DB/Tables
SS_ODS
Employees
CallInfos
Data

DB/Tables
SS_DWH
DimEmployees
DimCallInfos
DimDate
FactData

CREATE TABLE dbo.DimDate

SS_ADM
TechnicalRejects
FunctionalRejects

We created the following databases:

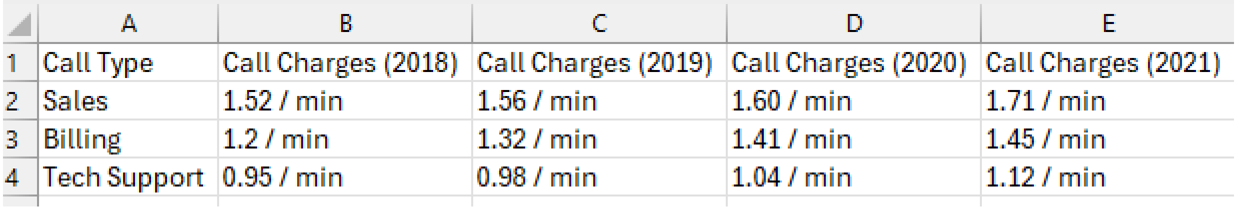
- SS_STA
- SS_ODS
- SS_DWH
- SS_ADM

Each database contains the tables listed in the pipeline model.

2. Staging

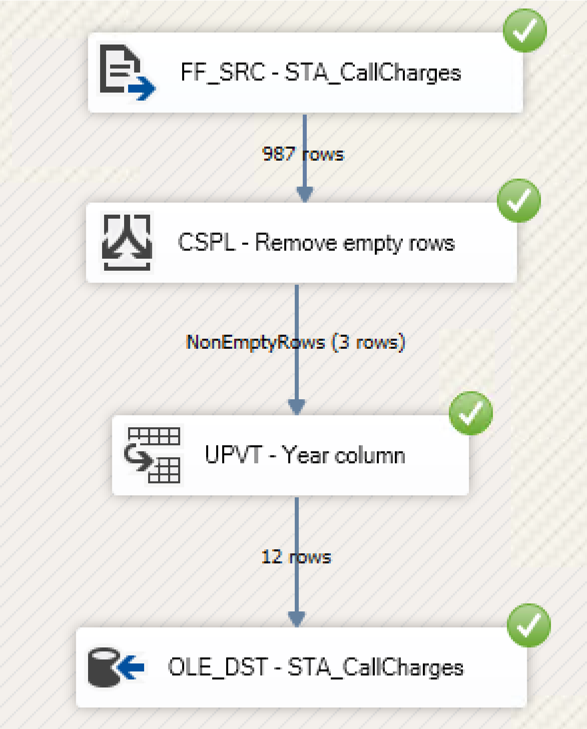
For the Staging phase, each information csv file (Employee.csv, Call Charges.csv, Call Types.csv, US States.csv) is imported to a corresponding table in SS_STA database.

The “Call Charges.csv” contains information that is in a human readable format, but not easy to read for a machine, as it mixes charge (price) information and year in the same column.

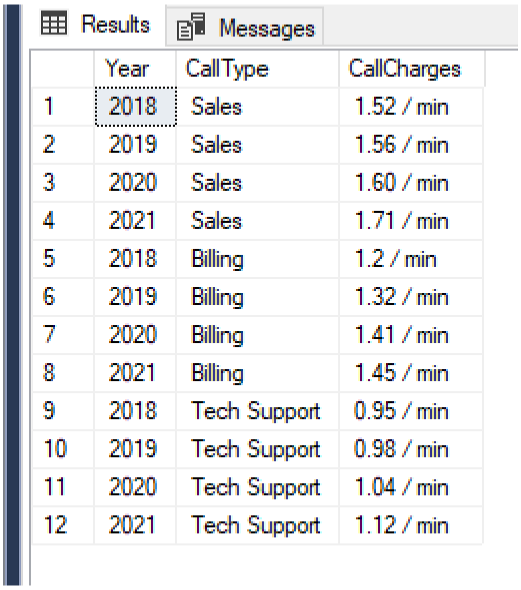


	A	B	C	D	E
1	Call Type	Call Charges (2018)	Call Charges (2019)	Call Charges (2020)	Call Charges (2021)
2	Sales	1.52 / min	1.56 / min	1.60 / min	1.71 / min
3	Billing	1.2 / min	1.32 / min	1.41 / min	1.45 / min
4	Tech Support	0.95 / min	0.98 / min	1.04 / min	1.12 / min

So we need to unpivot the table to make it more readable. Additionally, the file actually contains many empty rows, which we need to remove. So the data flow task for the STA_CallCharges is done this way:



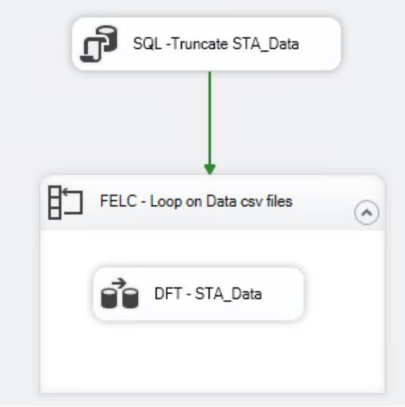
We first remove empty rows (984) and unpivot the remaining rows, to get the following result:

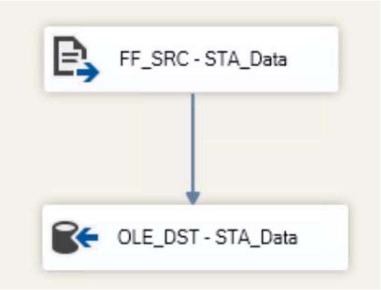


	Year	CallType	CallCharges
1	2018	Sales	1.52 / min
2	2019	Sales	1.56 / min
3	2020	Sales	1.60 / min
4	2021	Sales	1.71 / min
5	2018	Billing	1.2 / min
6	2019	Billing	1.32 / min
7	2020	Billing	1.41 / min
8	2021	Billing	1.45 / min
9	2018	Tech Support	0.95 / min
10	2019	Tech Support	0.98 / min
11	2020	Tech Support	1.04 / min
12	2021	Tech Support	1.12 / min

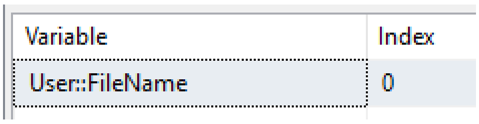
We now have Year and CallCharges in separated columns, which will be easier to process in the next stages.

For the 3 Data csv files containing call details (Data 2018.csv, Data 2019.csv and Data 2020.csv), we chose to regroup and stage them in 1 single table called Data. To do this we used a foreach loop container.



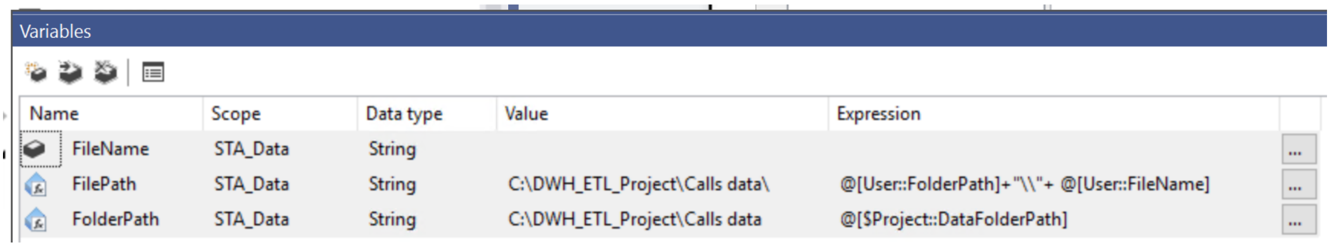


With a Foreach File enumerator with a variable mapping:



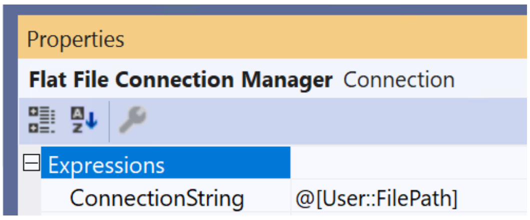
Variable	Index
User::FileName	0

The variables used are the following:



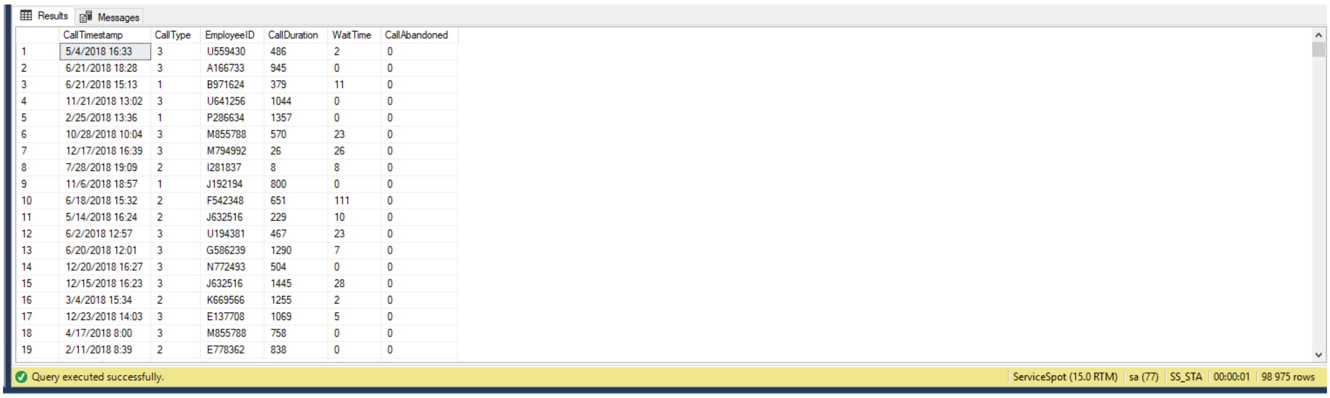
Name	Scope	Data type	Value	Expression
FileName	STA_Data	String		
FilePath	STA_Data	String	C:\DWH_ETL_Project\Calls data\	@[User::FolderPath]+"\\"+ @[User::FileName]
FolderPath	STA_Data	String	C:\DWH_ETL_Project\Calls data	@[$Project::DataFolderPath]

With the Flat File Connection Manager configured with the ConnectionString:



ConnectionString @[User::FilePath]

This way, we obtain a table dbo.Data containing 98975 rows, for all the calls of 2018, 2019 and 2020 combined:



	CallTimestamp	CallType	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
15	12/15/2018 16:23	3	J632516	1445	28	0
16	3/4/2018 15:34	2	K669566	1255	2	0
17	12/23/2018 14:03	3	E137708	1069	5	0
18	4/17/2018 8:00	3	M855788	758	0	0
19	2/11/2018 8:39	2	E778362	838	0	0

Query executed successfully. | ServiceSpot (15.0 RTM) | sa (77) | SS_STA | 00:00:01 | 98 975 rows

For the US States, Employees and Call Types csv files, we simply took the source files and created their tables in SS_STA database.

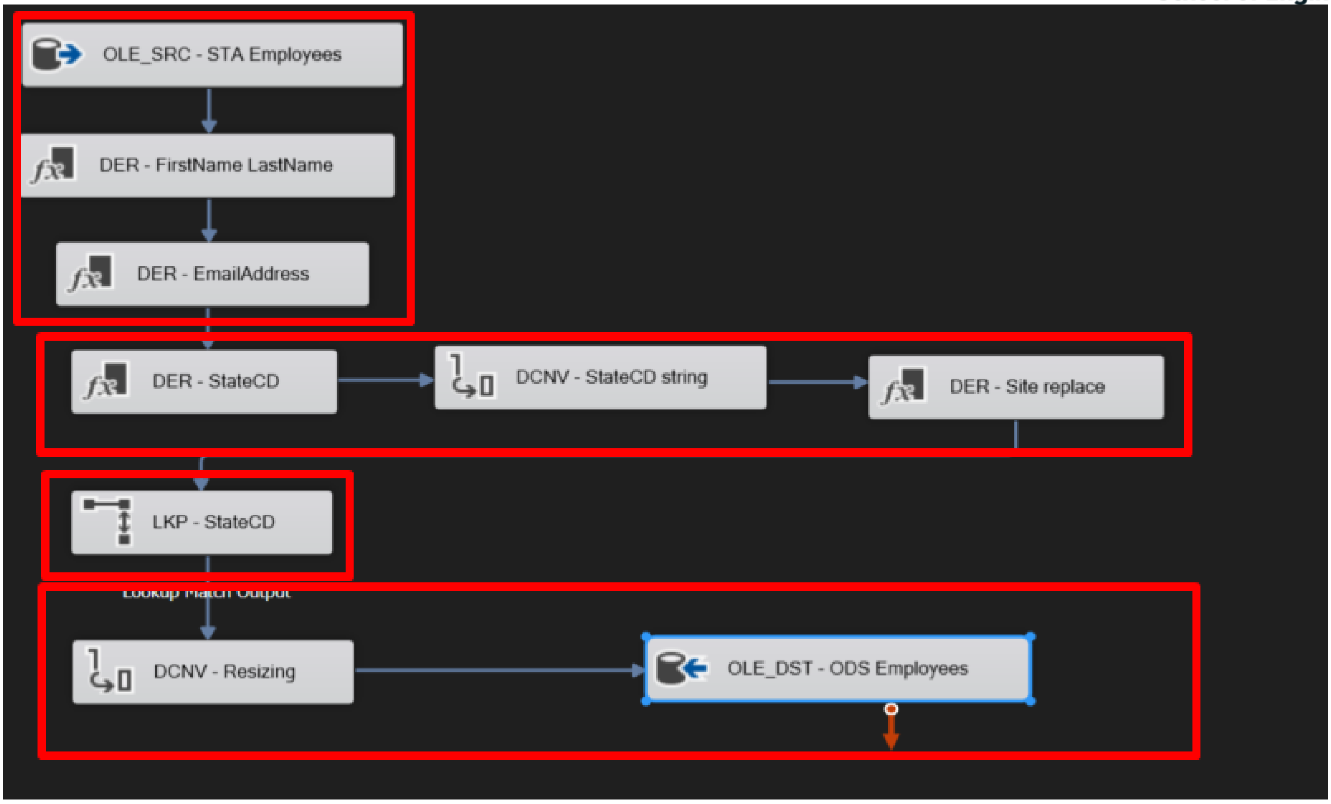
3. Operational Data Store

The second step of the pipeline is to load usable data into the Operational Data Store.

This procedure means that we need to clean and standardize data. We also need to take care of data that don’t respect quality standards and by directing as technical rejects.

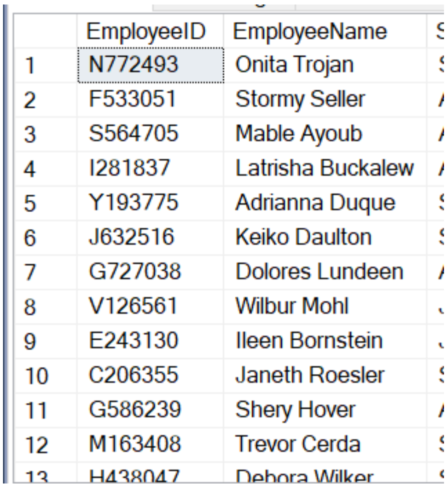
3.1 Employees Table

The data flow looks like this:

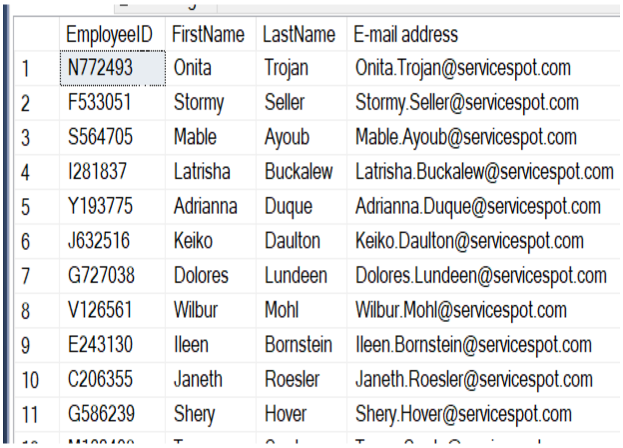


For the first segment, we split the column “Employee Name” into 2 new columns: FirstName and LastName. Then, we use these 2 columns to create another one named “EmailAdress”.

The result looks like this:



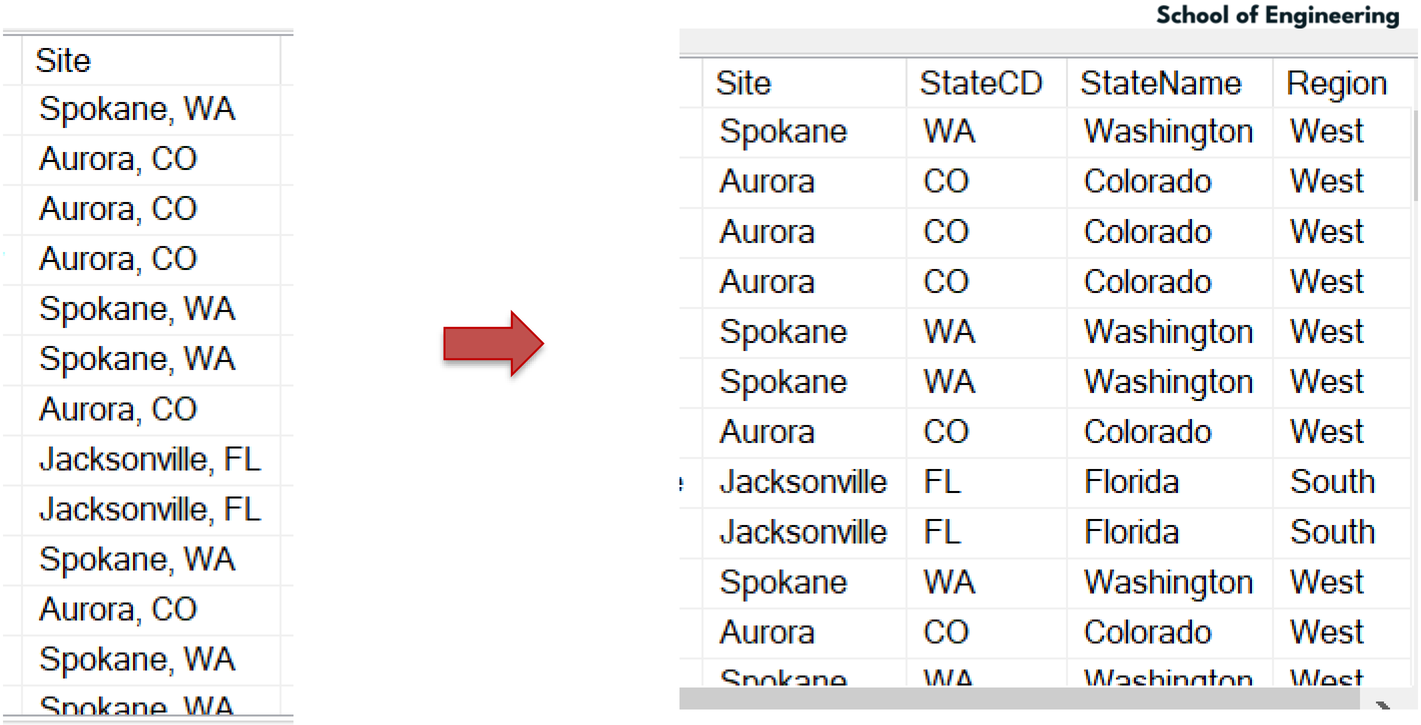
	EmployeeID	EmployeeName
1	N772493	Onita Trojan
2	F533051	Stormy Seller
3	S564705	Mable Ayoub
4	I281837	Latrisha Buckalew
5	Y193775	Adrianna Duque
6	J632516	Keiko Daulton
7	G727038	Dolores Lundeen
8	V126561	Wilbur Mohl
9	E243130	Ileen Bornstein
10	C206355	Janeth Roesler
11	G586239	Shery Hover
12	M163408	Trevor Cerda
13	H438047	Debora Wilker



	EmployeeID	FirstName	LastName	E-mail address
1	N772493	Onita	Trojan	Onita.Trojan@servicespot.com
2	F533051	Stormy	Seller	Stormy.Seller@servicespot.com
3	S564705	Mable	Ayoub	Mable.Ayoub@servicespot.com
4	I281837	Latrisha	Buckalew	Latrisha.Buckalew@servicespot.com
5	Y193775	Adrianna	Duque	Adrianna.Duque@servicespot.com
6	J632516	Keiko	Daulton	Keiko.Daulton@servicespot.com
7	G727038	Dolores	Lundeen	Dolores.Lundeen@servicespot.com
8	V126561	Wilbur	Mohl	Wilbur.Mohl@servicespot.com
9	E243130	Ileen	Bornstein	Ileen.Bornstein@servicespot.com
10	C206355	Janeth	Roesler	Janeth.Roesler@servicespot.com
11	G586239	Shery	Hover	Shery.Hover@servicespot.com

For the second segment, we split the column “Site” to create “StateCD” which is a column with only the CD. After that, “Site column” only contain the Site’s name.

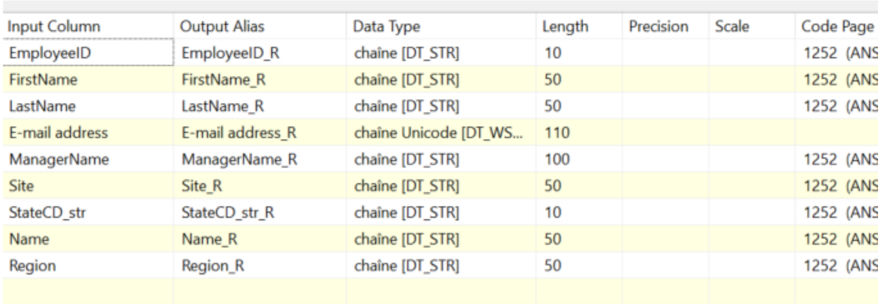
Then we used “StateCd” to lookup for 2 new colums which are present in the table “USStates” in the **STA database**. These columns are “StateName” and “Region”.



Site
Spokane, WA
Aurora, CO
Aurora, CO
Aurora, CO
Spokane, WA
Spokane, WA
Aurora, CO
Jacksonville, FL
Jacksonville, FL
Spokane, WA
Aurora, CO
Spokane, WA
Spokane, WA

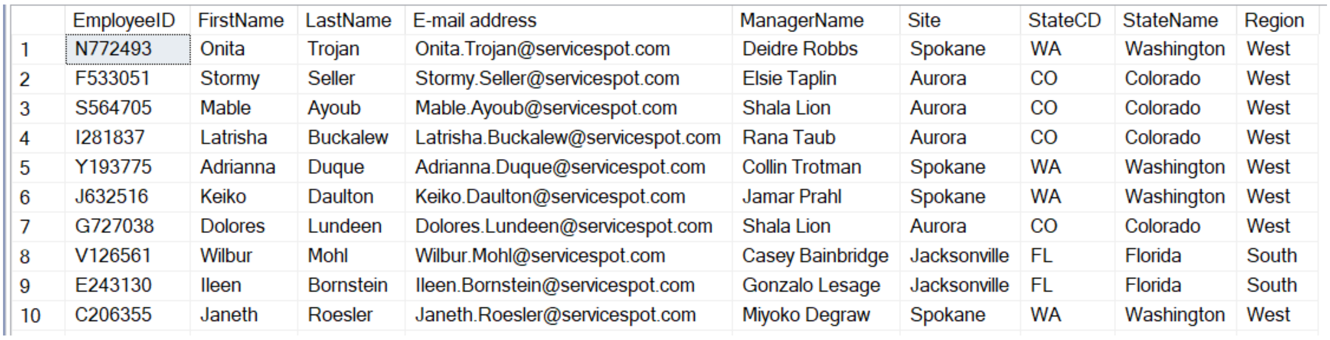
Site	StateCD	StateName	Region
Spokane	WA	Washington	West
Aurora	CO	Colorado	West
Aurora	CO	Colorado	West
Aurora	CO	Colorado	West
Spokane	WA	Washington	West
Spokane	WA	Washington	West
Aurora	CO	Colorado	West
Jacksonville	FL	Florida	South
Jacksonville	FL	Florida	South
Spokane	WA	Washington	West
Aurora	CO	Colorado	West
Spokane	WA	Washington	West

In the last segment, we resized some columns (cf. screenshot below) and we created a new table in the **ODS database** named “Employees”.



Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
EmployeeID	EmployeeID_R	chaine [DT_STR]	10			1252 (ANS
FirstName	FirstName_R	chaine [DT_STR]	50			1252 (ANS
LastName	LastName_R	chaine [DT_STR]	50			1252 (ANS
E-mail address	E-mail address_R	chaine Unicode [DT_WS...	110			
ManagerName	ManagerName_R	chaine [DT_STR]	100			1252 (ANS
Site	Site_R	chaine [DT_STR]	50			1252 (ANS
StateCD_str	StateCD_str_R	chaine [DT_STR]	10			1252 (ANS
Name	Name_R	chaine [DT_STR]	50			1252 (ANS
Region	Region_R	chaine [DT_STR]	50			1252 (ANS

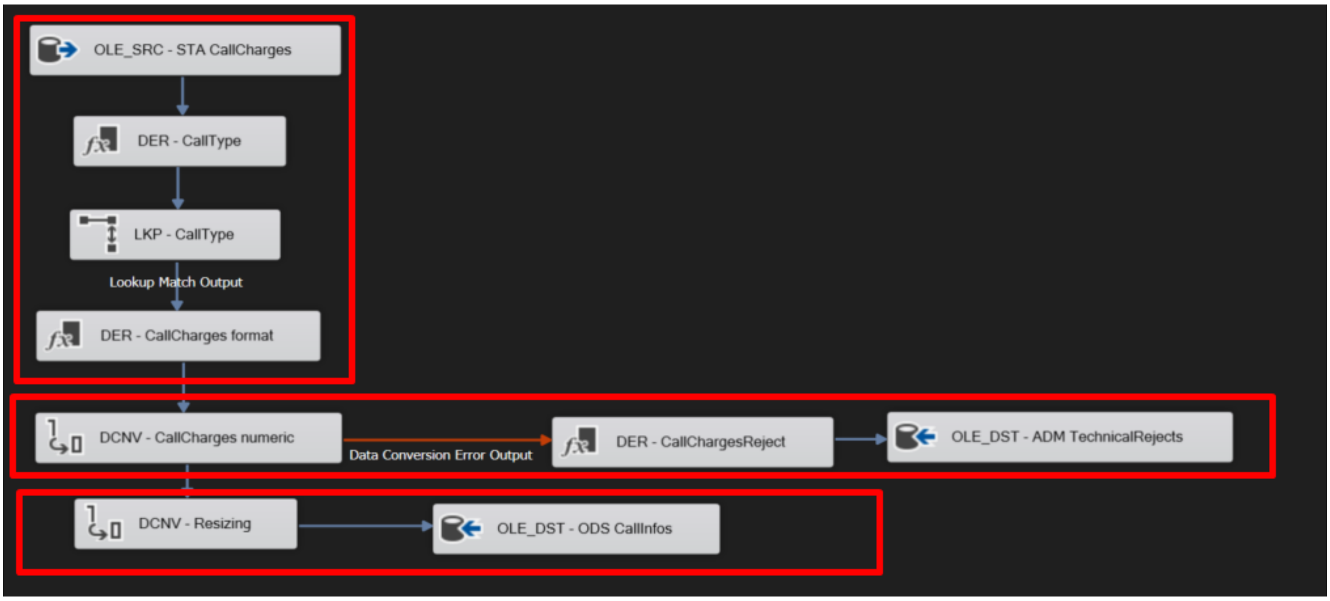
This is what the table looks like:



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

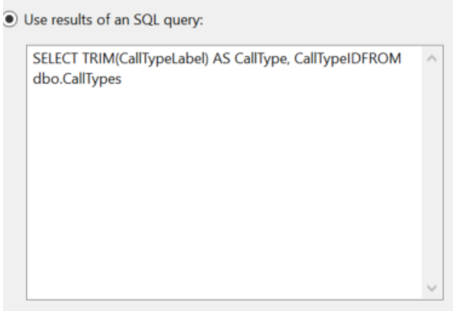
3.2 CallInfos Table

The data flow looks like this:



In the first segment, we get the data from CallCharges table from the STA database. Then, we performed a trim on the column "CallType". Indeed, there was space in some values and because of that not every value matched in the lookup.

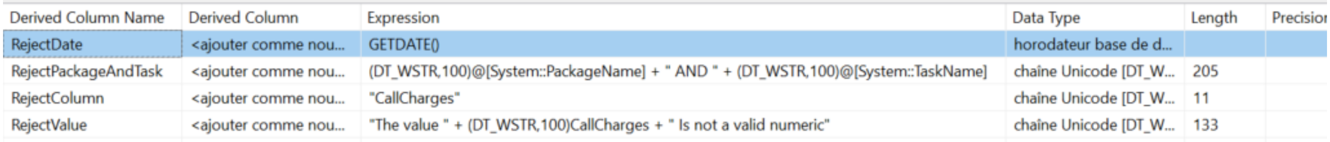
We used a lookup to get the "CallTypeID" from the CallTypes table from STA database. We directly used a query in the lookup to perform a trim on its CallType column.



● Use results of an SQL query:

```
SELECT TRIM(CallTypeLabel) AS CallType, CallTypeIDFROM
dbo.CallTypes
```

In the second segment, we converted "CallCharges" column into a numeric type. We check if this column has the right type. If not, we redirect the values to the **Technical Rejects table** in **ADM database**. This is how we created this latest:



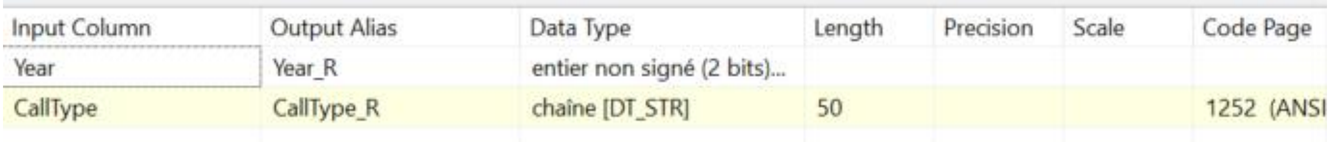
Derived Column Name	Derived Column	Expression	Data Type	Length	Precisior
RejectDate	<ajouter comme nou...	GETDATE()	horodateur base de d...		
RejectPackageAndTask	<ajouter comme nou...	(DT_WSTR,100)@[System::PackageName] + " AND " + (DT_WSTR,100)@[System::TaskName]	chaîne Unicode [DT_W...	205	
RejectColumn	<ajouter comme nou...	"CallCharges"	chaîne Unicode [DT_W...	11	
RejectValue	<ajouter comme nou...	"The value " + (DT_WSTR,100)CallCharges + " Is not a valid numeric"	chaîne Unicode [DT_W...	133	

If there is an error, we insert the following elements in the Technical Rejects table:

- The date of the error
- An error message

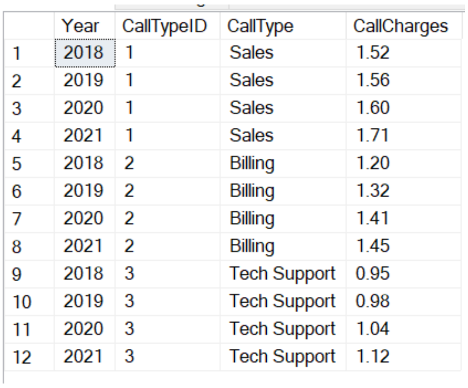
- The package and task causing the error
- The rejected value

In the last segment, we resized some columns (cf. screenshot below) and created **CallInfos table** in the **ODS database**.



Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
Year	Year_R	entier non signé (2 bits)...				
CallType	CallType_R	chaîne [DT_STR]	50			1252 (ANSI

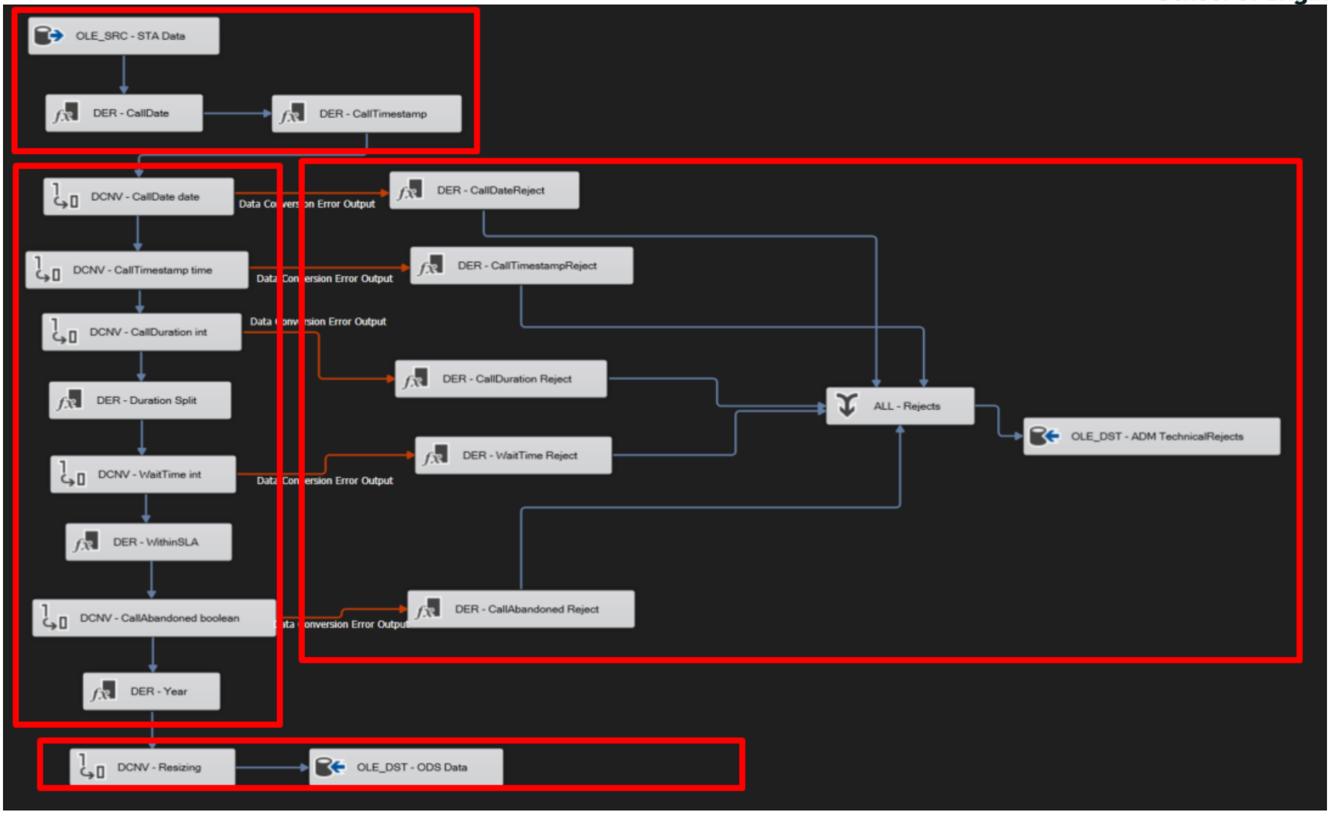
This is what the table looks like:



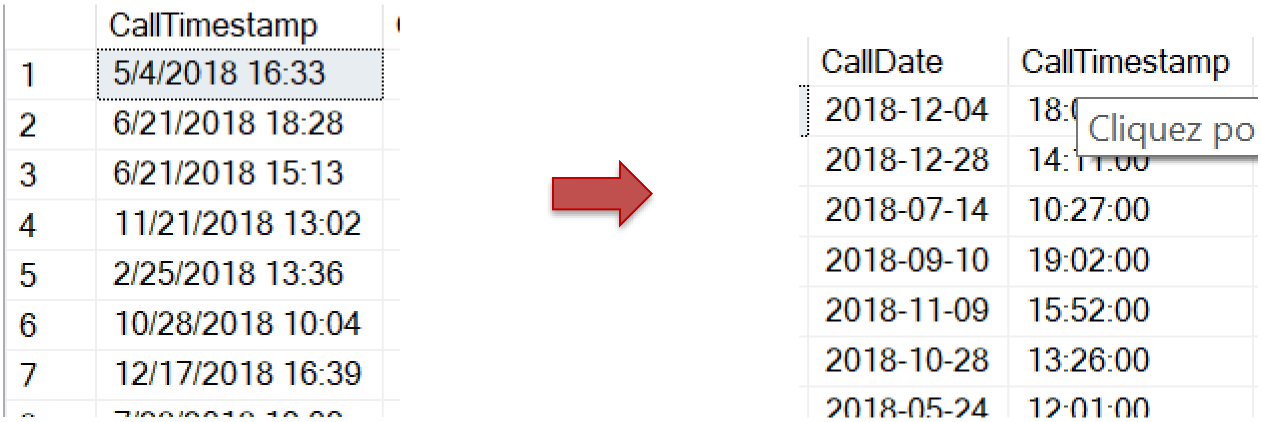
	Year	CallTypeID	CallType	CallCharges
1	2018	1	Sales	1.52
2	2019	1	Sales	1.56
3	2020	1	Sales	1.60
4	2021	1	Sales	1.71
5	2018	2	Billing	1.20
6	2019	2	Billing	1.32
7	2020	2	Billing	1.41
8	2021	2	Billing	1.45
9	2018	3	Tech Support	0.95
10	2019	3	Tech Support	0.98
11	2020	3	Tech Support	1.04
12	2021	3	Tech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

3.3 Data table

The data flow looks like this:


OLE_SRC - STA Data
DER - CallDate
DER - CallTimestamp
DCNV - CallDate date
Data Conversion Error Output
DER - CallDateReject
DCNV - CallTimestamp time
Data Conversion Error Output
DER - CallTimestampReject
DCNV - CallDuration int
Data Conversion Error Output
DER - CallDuration Reject
DER - Duration Split
ALL - Rejects
OLE_DST - ADM TechnicalRejects
DCNV - WaitTime int
Data Conversion Error Output
DER - WaitTime Reject
DER - WithinSLA
DCNV - CallAbandoned boolean
Data Conversion Error Output
DER - CallAbandoned Reject
DER - Year
DCNV - Resizing
OLE_DST - ODS Data

In the first segment, we get the data from the **Data table** in the **STA database**. Then, we used the “CallTimestamp” to create 2 columns, “CallDate” and “CallTimeStamp”. The first one contains the date and the second the time. We did it because it’s best practice to separate the “time” data (hour minute second) from the “date” data.

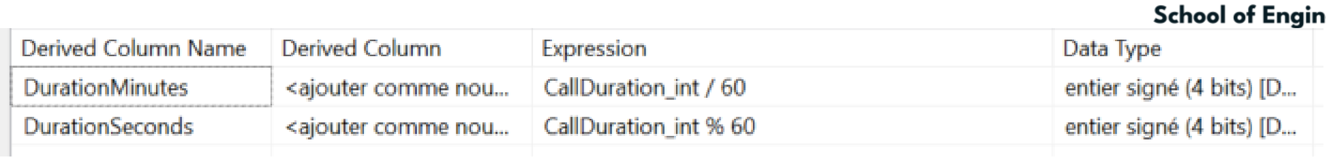


	CallTimestamp
1	5/4/2018 16:33
2	6/21/2018 18:28
3	6/21/2018 15:13
4	11/21/2018 13:02
5	2/25/2018 13:36
6	10/28/2018 10:04
7	12/17/2018 16:39

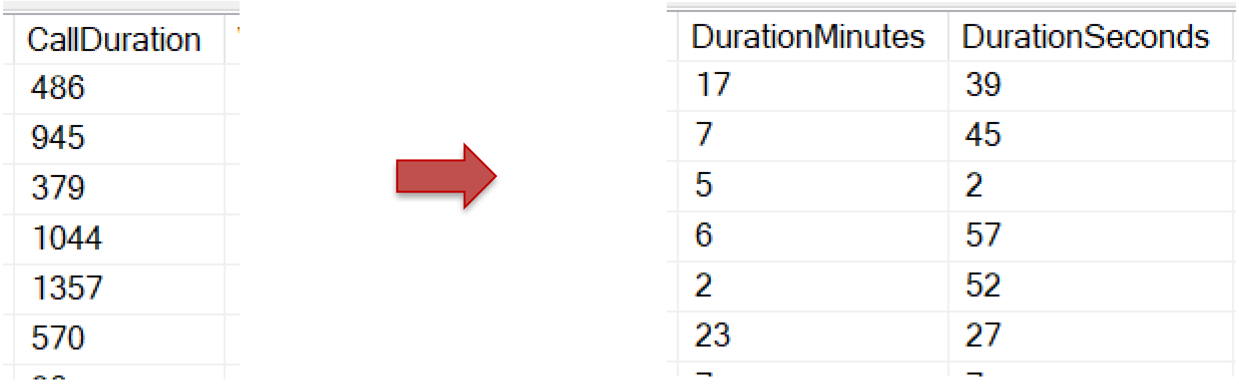
CallDate	CallTimestamp
2018-12-04	18:(
2018-12-28	14:11:00
2018-07-14	10:27:00
2018-09-10	19:02:00
2018-11-09	15:52:00
2018-10-28	13:26:00
2018-05-24	12:01:00

In the second segment (middle left), we did some conversion and enriched the data.

Firstly, we converted “CallDate”, “CallTimestamp” and “CallDuration”. Then, we used “CallTimeDuration” to create 2 columns: “DurationMinutes” and “DurationSeconds”. (cf. screenshot below).



Derived Column Name	Derived Column	Expression	Data Type
DurationMinutes	<ajouter comme nou...	CallDuration_int / 60	entier signé (4 bits) [D...
DurationSeconds	<ajouter comme nou...	CallDuration_int % 60	entier signé (4 bits) [D...

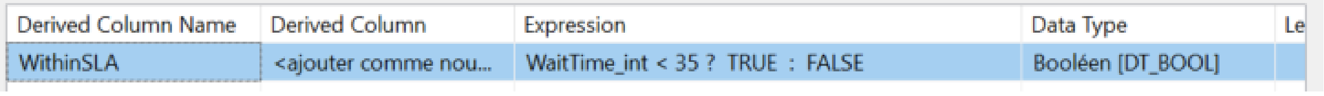


CallDuration
486
945
379
1044
1357
570

DurationMinutes	DurationSeconds
17	39
7	45
5	2
6	57
2	52
23	27

Secondly, we converted “WaitTime” to create a new column named “WithinSLA”. We did it because in this company, it is expected that a call must be answered within 35 seconds of waiting time to comply with the SLA. Thus, any call must be considered “Within SLA” if the waiting time is below 35 seconds, otherwise it should be “Outside SLA”.

This how we proceeded:



Derived Column Name	Derived Column	Expression	Data Type	Le
WithinSLA	<ajouter comme nou...	WaitTime_int < 35 ? TRUE : FALSE	Booléen [DT_BOOL]	

Thirdly, we converted “CallAbandonned” and created a column named “Year”. This latest column is the year of the call, we’ll use it after to make the link in lookup function to get “CallInfoSurrogateKey” in the **FactData table** from **DWH database**.

In the middle right segment, we redirect the values of certain columns when there is an error in their format and add them to the **Technical Rejects table** in the **ADM database**.

In the last segment, we did some resizing and created **Data table** in the **ODS database**.

4. Datawarehouse

The last part is to integrate the data into the Data Warehouse. To do this, we use a star schema for the database. This schema comprises a main table, the “fact table”, surrounded by dimension tables. The fact table and dimension tables are linked by surrogate key.

4.1 Database design

In our case, the fact table will contain calls data (call date, duration...).

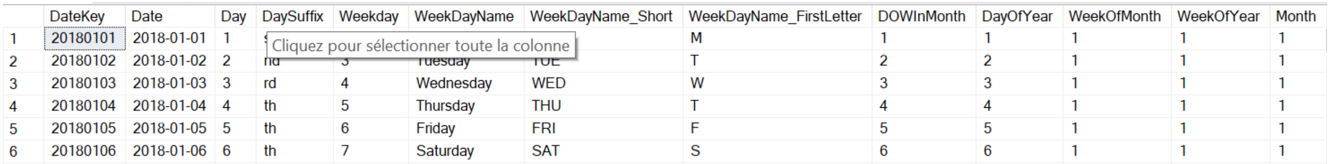
For the dimensions, we'll use the **Date table**. This contains various information about the date (name of the day, number of the month...).

Then we'll have **CallInfos table**. It contains information about calls (calltype, callcharges…).

Finally, the **Employee table**. As its name suggests, it contains employee data.

4.2 Integration of the Date dimension

As said before, this table contains data about dates in different formats. To create it, we used the script provided in the project description. This is what the tables looks like:

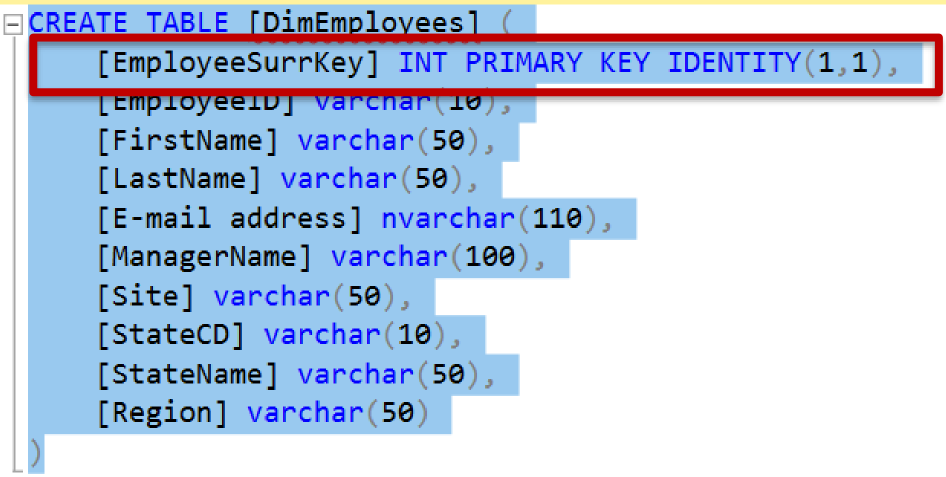


	DateKey	Date	Day	DaySuffix	Weekday	WeekDayName	WeekDayName_Short	WeekDayName_FirstLetter	DOWInMonth	DayOfYear	WeekOfMonth	WeekOfYear	Month
1	20180101	2018-01-01	1	s	[illegible]	[illegible]	[illegible]	M	1	1	1	1	1
2	20180102	2018-01-02	2	nd	3	Tuesday	TUE	T	2	2	1	1	1
3	20180103	2018-01-03	3	rd	4	Wednesday	WED	W	3	3	1	1	1
4	20180104	2018-01-04	4	th	5	Thursday	THU	T	4	4	1	1	1
5	20180105	2018-01-05	5	th	6	Friday	FRI	F	5	5	1	1	1
6	20180106	2018-01-06	6	th	7	Saturday	SAT	S	6	6	1	1	1

Cliquez pour sélectionner toute la colonne

4.3 Integration of the dimension

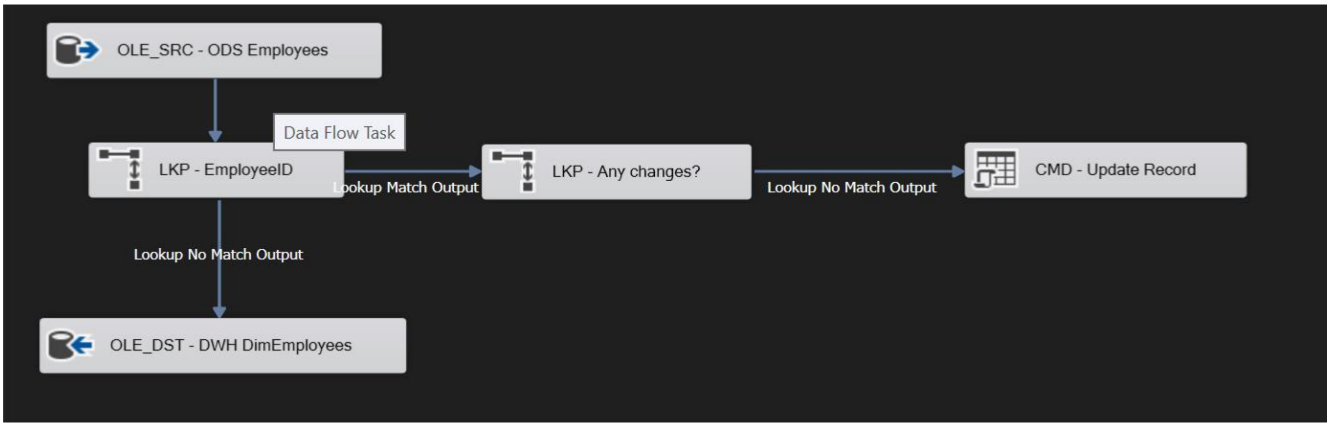
Firstly, we create the **DimEmployees table** in **DWH database**. We also add a column named "EmployeeSurrKey", because we'll need it to join the **Fact table.**



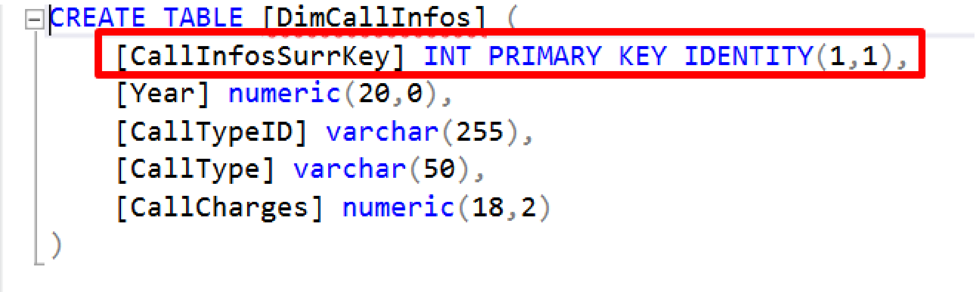
```
CREATE TABLE [DimEmployees] (
    [EmployeeSurrKey] INT PRIMARY KEY IDENTITY(1,1),
    [EmployeeID] varchar(10),
    [FirstName] varchar(50),
    [LastName] varchar(50),
    [E-mail address] nvarchar(110),
    [ManagerName] varchar(100),
    [Site] varchar(50),
    [StateCD] varchar(10),
    [StateName] varchar(50),
    [Region] varchar(50)
)
```

We decided to use SCD1 strategy. Basically, if there is any change we update the table. For that we do a lookup on "EmployeeID" between **ODS Employees table** and **DWH DimEmployees table**.

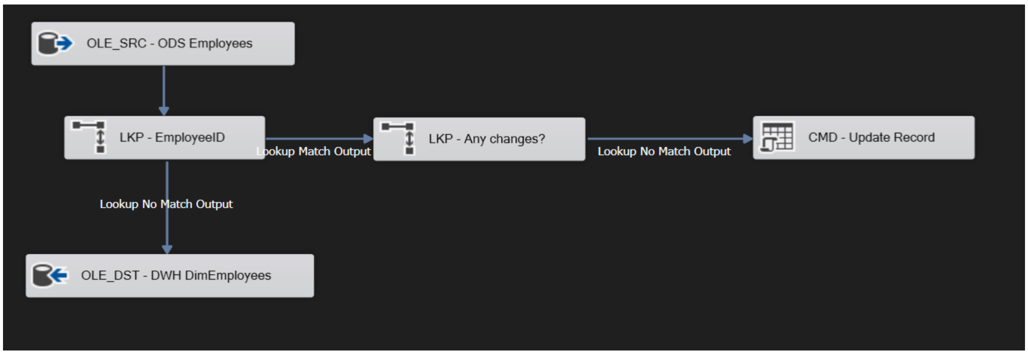
The data flow looks like that:



Secondly, we do the same process for the **DimCallInfos table** in **DWH database. (**cf.screenshot below).



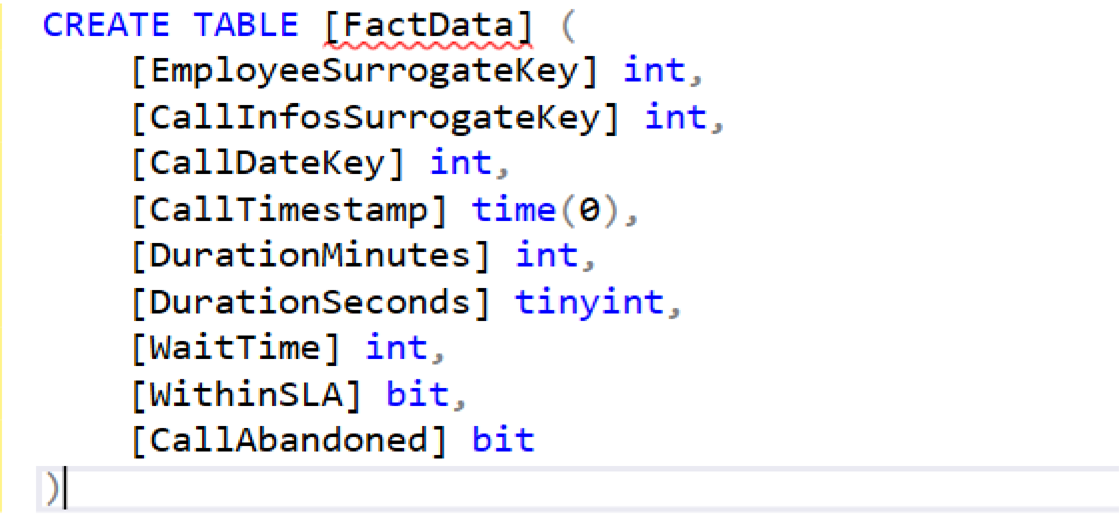
```
CREATE TABLE [DimCallInfos] (
    [CallInfosSurrKey] INT PRIMARY KEY IDENTITY(1,1),
    [Year] numeric(20,0),
    [CallTypeID] varchar(255),
    [CallType] varchar(50),
    [CallCharges] numeric(18,2)
)
```



4.4 Integration of the Facts table

Firstly, we constructed the fact table incorporating the EmployeeSurrogateKey and CallInfosSurrogateKey to establish links with the dimension tables.

SQL Query:

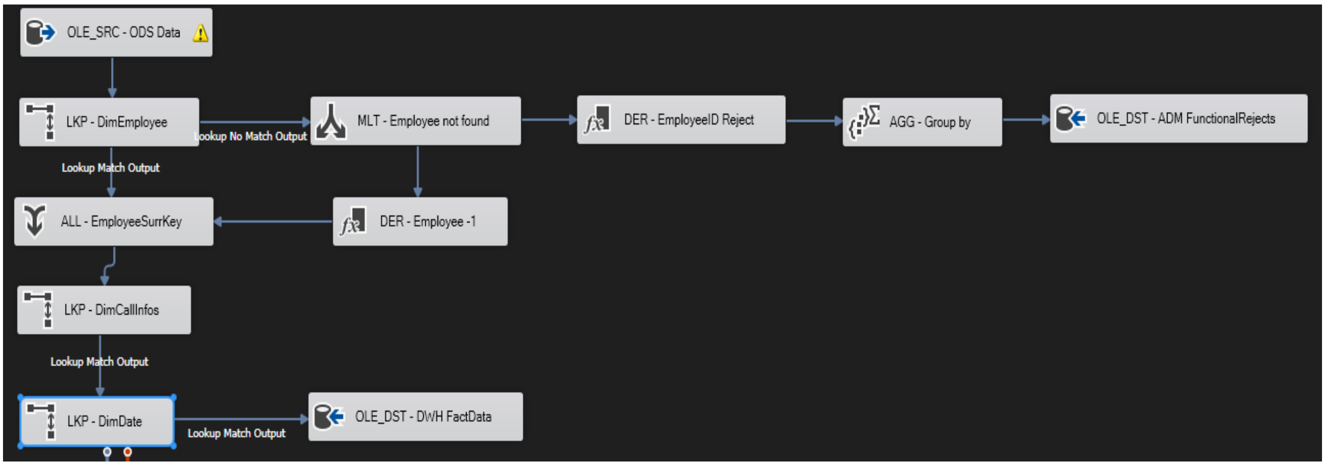


```
CREATE TABLE [FactData] (
    [EmployeeSurrogateKey] int,
    [CallInfosSurrogateKey] int,
    [CallDateKey] int,
    [CallTimestamp] time(0),
    [DurationMinutes] int,
    [DurationSeconds] tinyint,
    [WaitTime] int,
    [WithinSLA] bit,
    [CallAbandoned] bit
)
```

Subsequently, we employed the Lookup transformation to join DimEmployee, followed by uniting all the data with EmployeeSurrogateKey.

Thereafter, we once again utilised the Lookup transformation to join DimCallInfos and the DimDate table.

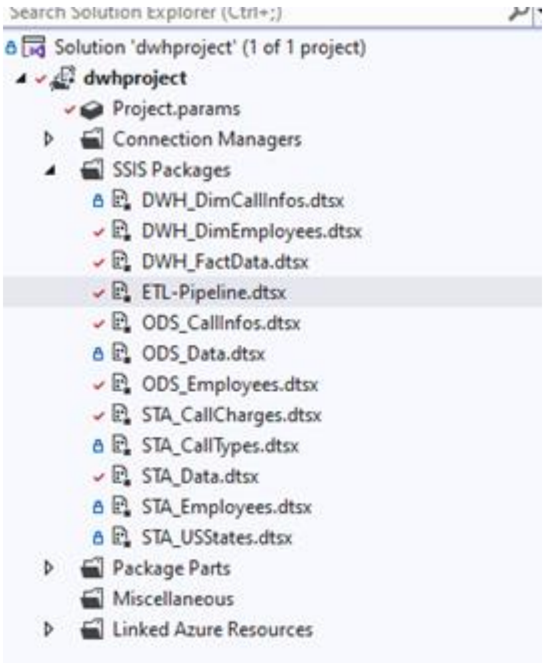
Data Flow:



At the stage of the Lookup with DimEmployee, the non-matching output was handled by the Multicast function to filter the unmatched cases.

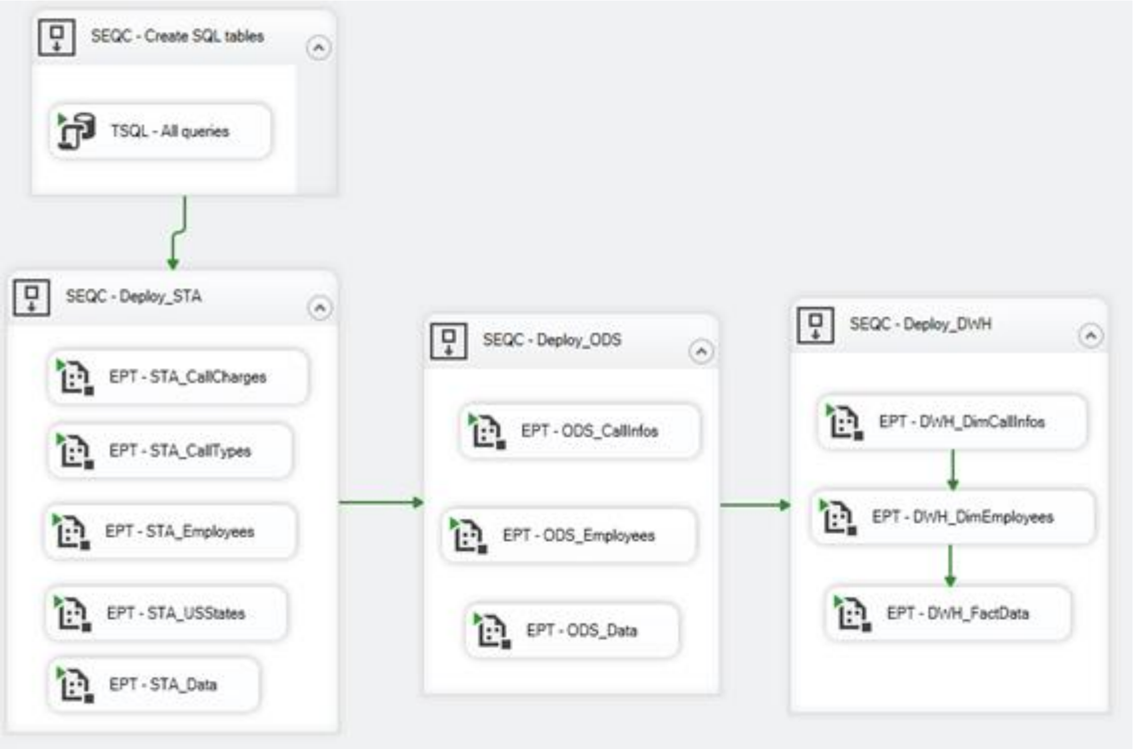
5. Project Deployment

We have outlined all necessary steps to deploy our data warehouse using the available data. To ensure the pipeline executes reproducibly, we have defined an additional package, "ETL-Pipeline.dtsx". Using this package allow to use one interface for the entire pipeline. The creation of the tables is done with TSQL task(s) and the execution package task (EPT) launches the data flows beginning with the transformation packages for STA followed by ODS, and finally DWH. The STA has 5 packages, ODS has 3 packages, and DWH has 3 packages adapted for transformation.



Search Solution Explorer (Ctrl+;)

Solution 'dwhproject' (1 of 1 project)
- dwhproject
  - Project.params
  - Connection Managers
  - SSIS Packages
    - DWH_DimCallInfos.dtsx
    - DWH_DimEmployees.dtsx
    - DWH_FactData.dtsx
    - ETL-Pipeline.dtsx
    - ODS_CallInfos.dtsx
    - ODS_Data.dtsx
    - ODS_Employees.dtsx
    - STA_CallCharges.dtsx
    - STA_CallTypes.dtsx
    - STA_Data.dtsx
    - STA_Employees.dtsx
    - STA_USStates.dtsx
  - Package Parts
  - Miscellaneous
  - Linked Azure Resources



This flowchart provides a visual representation of the data flow through the ETL process, ensuring that data is properly extracted, transformed, and loaded for analysis and reporting.

- **Extract**: The process begins with the creation of SQL tables and the execution of all necessary queries to gather data. In our case, we have used TSQL to create these tables needed for ETL operations.
- **Transform**: Data is then deployed to different areas for transformation:
  - **STA (Staging Area)**: Handles various types of data such as CallCharges, CallTypes, Employees, USStates, and Data.
  - **ODS (Operational Data Store)**: Manages CallInfos, Employees, and Data.
  - **DWH (Data Warehouse)**: Involves more complex transformations for DimCallInfos, DimEmployees, and FactData.
- **Load**: After transformation, the data is loaded into the respective areas (STA, ODS, DWH) for further use.

6. Use Case

Scenario: Call Analysis for ServiceSpot Company

- **Objective**: To identify where employees are making the most calls that breach SLAs and incur the highest charges, providing valuable insights for resource allocation and cost management decisions.

- **Tables Involved**:
  - DimEmployees: Contains employee details.
  - DimCallInfos: Contains records of calls made by employees and the associated call charges.
  - DimDate: Contains time intelligence data
  - FactData: Contains details of all the records.

- **SQL Query**:

```
CREATE VIEW EmployeeCallAnalysis AS
SELECT
        e.EmployeeId, e.FirstName, e.LastName,
        COUNT(c.CallTypeID) AS TotalCalls,
        SUM(c.CallCharges) AS TotalCharges,
        COUNT(CASE WHEN d.CallAbandoned = 1 THEN 1 END) AS TotalCallAbandonned,
        COUNT(CASE WHEN d.WithinSLA = 1 THEN 1 END) AS Count_within_SLA,
        e.StateName,
        f.Year
FROM
        FactData d
JOIN
        DimEmployees e ON e.EmployeeSurrKey = d.EmployeeSurrogateKey
JOIN
        DimCallInfos c ON c.CallInfosSurrKey = d.CallInfosSurrogateKey
JOIN
        DimDate f ON f.DateKey=d.CallDateKey
GROUP BY
        e.EmployeeID, e.FirstName, e.LastName, e.StateName, f.Year
```

- **Explanation**:
  - The query joins the DimEmployee, DimCallInfos, DimDate, and FactData tables.
  - It calculates the total number of calls, total charges, total calls abandoned for each employee.
  - The COUNT(CASE WHEN d.CallAbandoned = 1 THEN 1 END) expression ensures that only the records where CallAbandoned equals 1 are counted in the TotalCallAbandoned column.
  - The COUNT(CASE WHEN d.WithinSLA = 1 THEN 1 END) expression ensures that only the records where WithinSLA equals 1 are counted in the Count_within_SLA column.
  - Finally, it groups the results by employee id, first and last name of the employee, and state to show the distribution of calls and charges by state.

- At the end, Order by sorts the data first by state, then by year in ascending order, and finally by the count of calls within SLA.

"

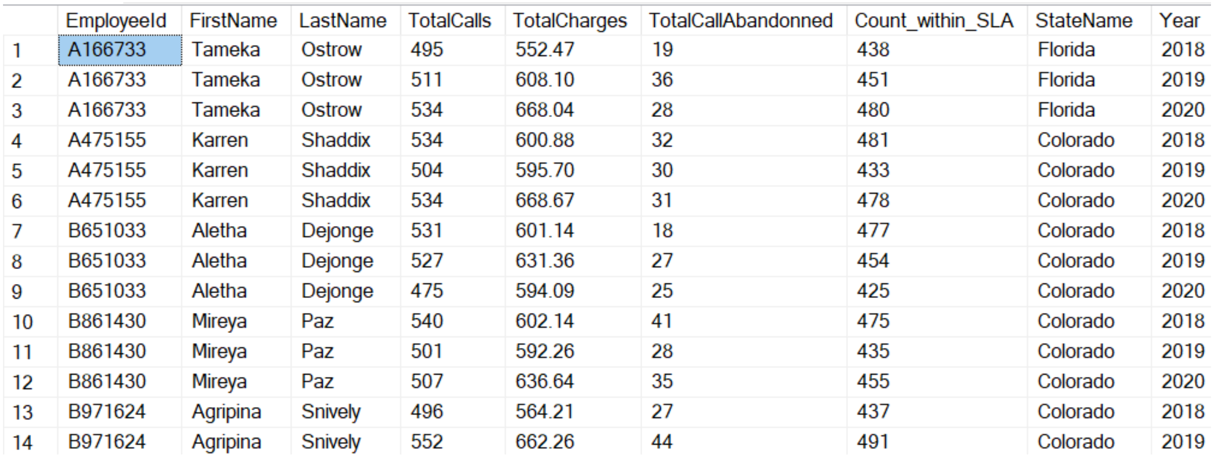
Note. If you want to use order by, then you can ignore creating View and simply run the SQL query and append ORDER BY e.StateName, f.Year ASC, Count_within_SLA to the last.

"

This query helps the company analyze employee performance and call efficiency by providing insights into:

- The total number of calls and associated charges per employee.
- The number of calls abandoned and those handled within SLA.
- The geographical distribution of employees (by state).
- The performance of employees over different years.

These insights can inform decisions on improving call efficiency, managing costs, and optimizing resource allocation.



	EmployeeId	FirstName	LastName	TotalCalls	TotalCharges	TotalCallAbandonned	Count_within_SLA	StateName	Year
1	A166733	Tameka	Ostrow	495	552.47	19	438	Florida	2018
2	A166733	Tameka	Ostrow	511	608.10	36	451	Florida	2019
3	A166733	Tameka	Ostrow	534	668.04	28	480	Florida	2020
4	A475155	Karren	Shaddix	534	600.88	32	481	Colorado	2018
5	A475155	Karren	Shaddix	504	595.70	30	433	Colorado	2019
6	A475155	Karren	Shaddix	534	668.67	31	478	Colorado	2020
7	B651033	Aletha	Dejonge	531	601.14	18	477	Colorado	2018
8	B651033	Aletha	Dejonge	527	631.36	27	454	Colorado	2019
9	B651033	Aletha	Dejonge	475	594.09	25	425	Colorado	2020
10	B861430	Mireya	Paz	540	602.14	41	475	Colorado	2018
11	B861430	Mireya	Paz	501	592.26	28	435	Colorado	2019
12	B861430	Mireya	Paz	507	636.64	35	455	Colorado	2020
13	B971624	Agripina	Snively	496	564.21	27	437	Colorado	2018
14	B971624	Agripina	Snively	552	662.26	44	491	Colorado	2019

Conclusion

Our project implemented a comprehensive ETL pipeline for call center analysis, showcasing practical data warehousing and ETL techniques. Our diverse team collaboratively developed a robust data processing framework that effectively integrates data from various sources.

Key methods used include:

- **Unpivot Transformation**: Reorganized "Call Charges" CSV for better machine readability.
- **For Each Loop Container**: Dynamically processed multiple yearly CSV files.
- **Lookup Transformations**: Integrated dimension tables (DimEmployee, DimCallInfos, DimDate).
- **Multicast Function**: Filtered unmatched cases during Lookup operations.

This project was meticulously structured into three distinct phases: Staging (STA), Operational Data Store (ODS), and Data Warehousing (DWH).

The staging (STA) phase handled CSV data transformation, while the Operational Data Store (ODS) phase focused on data cleaning and standardization. The Data Warehouse (DWH) adopted a star schema, linking a fact table with dimension tables to facilitate efficient data retrieval and analysis.

Our deployment strategy included the "ETL-Pipeline.dtsx" package, ensuring reproducibility and streamlined execution of the ETL process.

Through our use case analysis, we provided valuable insights into call handling, SLA adherence, and cost management. This project demonstrates the importance of meticulous planning, teamwork, and technical proficiency in successful data warehousing. Our solution meets current analytical needs and provides a scalable framework for future data integration and analysis.

In conclusion, this project not only streamlined our data processing pipeline but also laid a robust foundation for future scalability and analytical capabilities. By systematically organising the data into well-defined phases and employing advanced ETL techniques, we ensured that the data warehouse is both efficient and scalable, ready to provide valuable insights for the call centre’s operations and strategic decision-making.