

**DATA WAREHOUSING & ETL PROJECT**

S23 Cohort, Applied MSc in Data Engineering

*SSIS & SQL SERVER PROJECT REPORT*

**DATA WAREHOUSING & ETL**

Performed by: Gali Aydaraliev

MSc in Data Engineering 2023-2024

Data ScienceTech Institute, SPOC

Date: 19.10.2023

**TABLE OF CONTENTS**

[**I. INTRODUCTION**](#_I._INTRODUCTION)

[1. Data](#_1._DATA)

[**II. PIPELINE DESIGN**](#_II._PIPELINE_DESIGN)

[1. Staging database](#_1._STAGING_DATABASE)

[2. Operational data store](#_2._OPERATIONAL_DATA)

[3. Datawarehouse](#_3._DATAWAREHOUSE)

[4. Project deployment](#_4._PROJECT_DEPLOYMENT)

[**III. CONCLUSION**](#_III._CONCLUSION)

# **I. INTRODUCTION**

The importance of data is significant nowadays as it helps to make informed and verified decisions to drive business growth. However, to effectively utilize the accessible data and derive advantages from it, it is necessary to define how to store large amounts of data from multiple sources. The optimal solution for such an issue is a Data Warehouse implementation. This project exactly aims to investigate all processes concerning ETL & Data Warehouse.

The main assignment of this project was a development an ETL project with SSIS that should help Service Spot, an IT company, to load data into their data warehouse and analyze their call center data.

The tools used in this work are following:

* Visual Studio 2022;
* SQL Server Management Studio 19

## **1. DATA**

The data is composed of a set of csv files: Call Charges, Call Types, Employees and US States. In addition to those files, there are Data 2018, 2019, 2020 csv files that are stored in a separate folder called “Calls Data”.

**Lookup data**

The “Call Charges” is arranged as follows:

* Each column indicates the amount of money that is charged to a customer for each minute spent on the phone from 2018 to 2021
* Each line represents a specific call types

The “Call Types” consists of the following data:

* Call type ID
* Call type label

The “Employees” contains the following data:

* Employee ID
* Employee name
* Site (site name where employee is working at)
* Manager name

The “US States” consists of the three columns:

* State CD (2-letter state code)
* State name
* Region (US region name)

**CSV files (Data YYYY)**

These csv files have the historical data of calls from 2018 to 2020:

* Call timestamp
* Call type
* Employee ID
* Call duration
* Wait time
* Call abandoned (1 = Yes, 0 = No)

# **II. PIPELINE DESIGN**

## **1. STAGING DATABASE**

The Staging Area is responsible for extracting and storing raw data from multiple sources. We will cover each of STA components, one by one, in this project.

**Call charges**

According to the assignment, the “Call charges” file is provided in an “easy to read” format for a human, but not very efficient to deal with for a database. That is why we need to unpivot the data coming from that file using transformation in SSIS

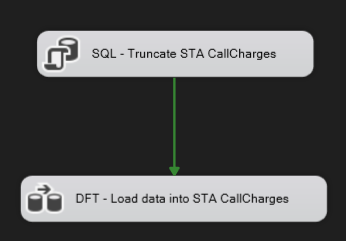
*The initial table of the “Call charges” file*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

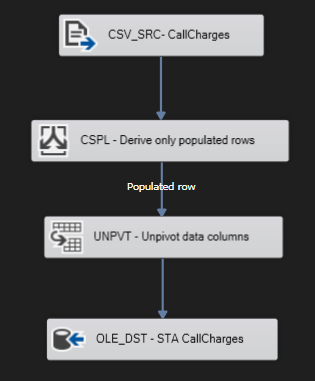
*The required unpivot table for the “Call charges” file*

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

To start with, at this step we define control flow level by connecting a data flow task with an execute SQL task. It is necessary to truncate table and fill it with new data. It allows excluding any update inconsistency. Then we determine a data flow level by uploading csv file and directing it to the staging database. In general, all csv files have similar procedure in this work. However, some of them require certain alterations therefore, it is worthwhile to consider in detail each of their transformations.

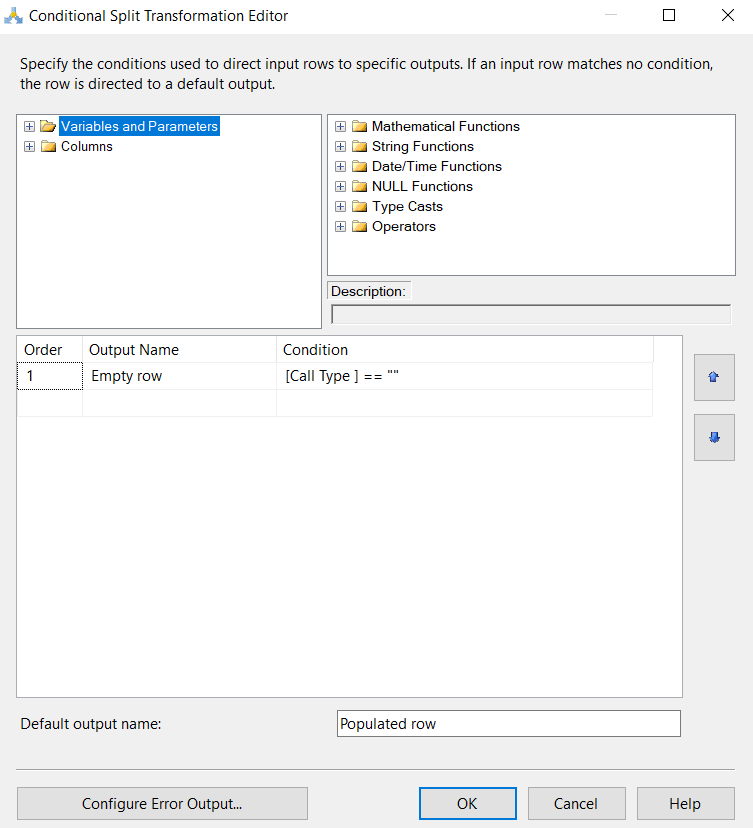


Control Flow (STA CallCharges)



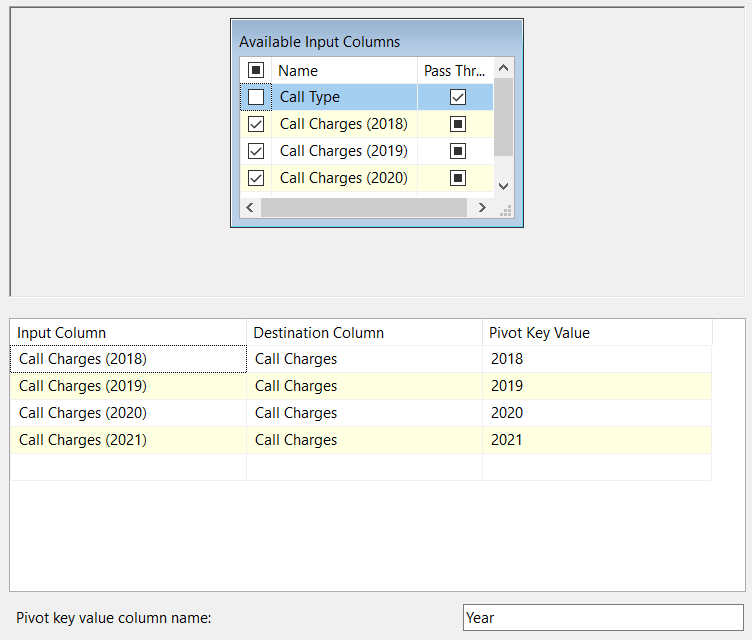
Data Flow (STA CallCharges)

As it was mentioned before, “CallCharges” table should be unpivoted. For this reason, we applied conditional split and unpivot transformations to the data columns.



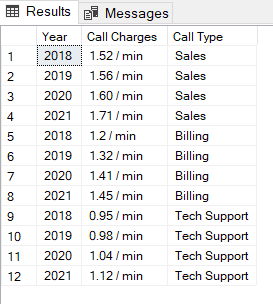
Conditional Split (STA CallCharges)

The conditional split was needed to exclude empty rows and derive only populated ones.



Unpivot (STA CallCharges)

As for unpivot operation, we expanded values of the call charges by year and redirected them to the new column “Call Charges”. The pivot key values were moved to the “Year” column.

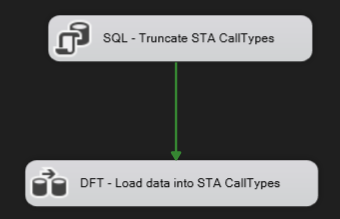


SQL Query Results (STA CallCharges)

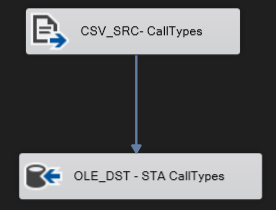
As the result, the data from “Call Charges” file was stored in the SQL Server for further exploitation.

The following csv files have similar design only without any transformations. Their data was transmitted to the STA database directly.

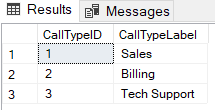
**Call types**



Control Flow (STA CallTypes)

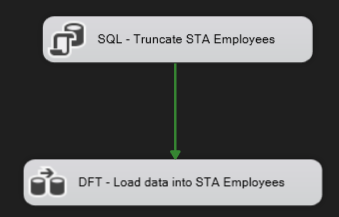


Data Flow (STA CallTypes)

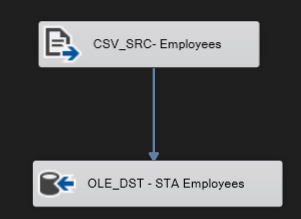


SQL Query Results (STA CallTypes)

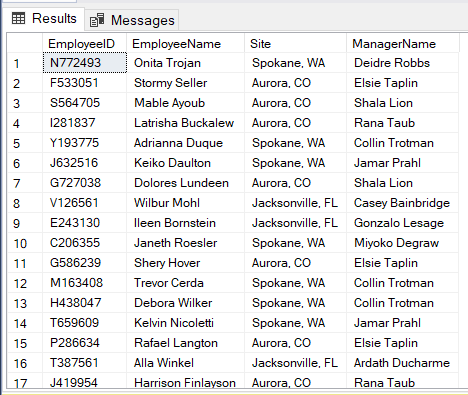
**Employees**



Control Flow (STA Employees)

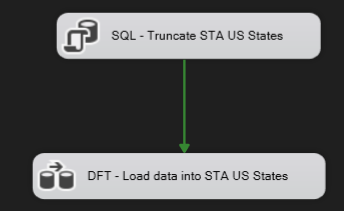


Data Flow (STA Employees)

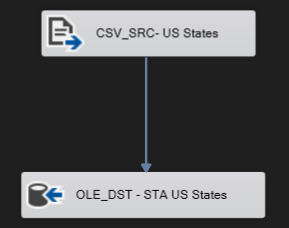


SQL Query Results (STA Employees)

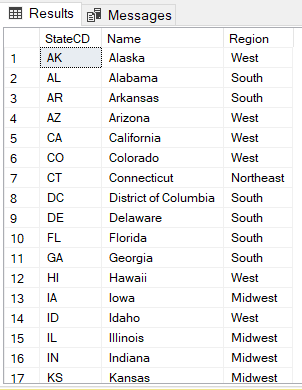
**US States**



Control Flow (STA US States)



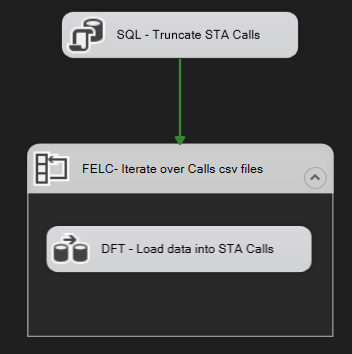
Data Flow (STA US States)



SQL Query Results (STA US States)

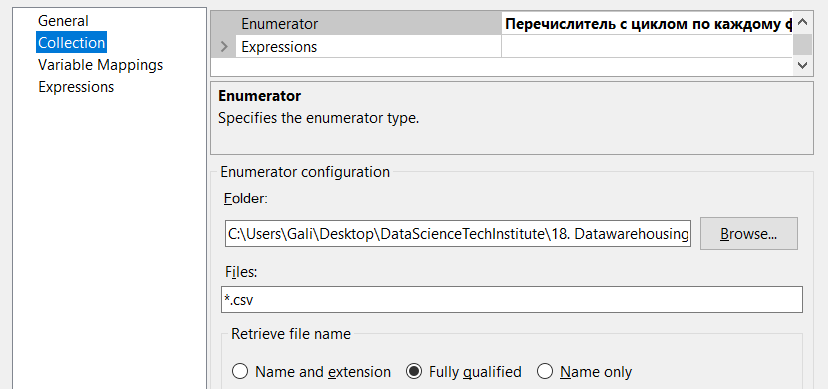
**Calls data**

For the “Calls Data” sources, which consist of three same-structured csv files, “Foreach Loop Container” was used to loop over all the files presented in the “Calls data” folder. It must be applied to consider the case of new files coming.

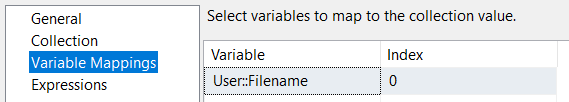


Control Flow (STA Calls)

As there are multiple files, we should specify the enumerator type. The most acceptable seemed to be “Foreach Item Enumerator” which have certain configuration connected with folder directory and file names. In our case, we determined fully qualified path in order to retrieve required files.

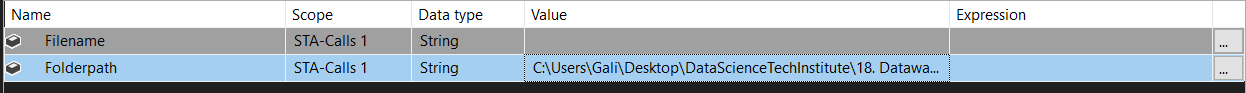


Foreach Loop Container (STA Calls)

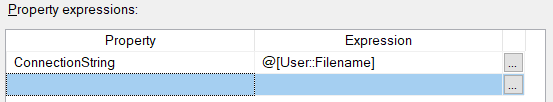


Foreach Loop Container (STA Calls)

However, before configuring Foreach Loop Container, a *“Filename”* variable should be created. It was essential especially for property definition which allows the flat file connection manager to be more dynamic and iterate through the list of files.

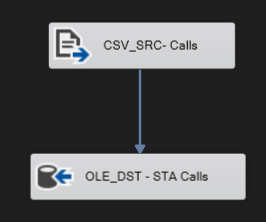


Variables (STA Calls)



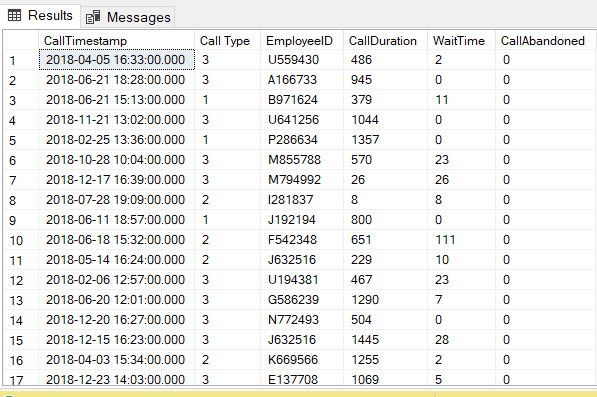
Property Expressions Editor (STA Calls)

The dataflow of “STA Calls” was defined as follows:



Data Flow (STA Calls)

The result of SQL query for “STA Calls” database is presented below. It consists of 98,975 rows ordered by year.



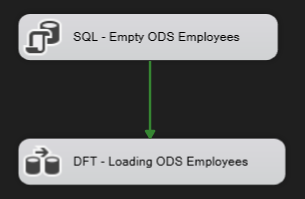
SQL Query Results (STA Calls)

## **2. OPERATIONAL DATA STORE**

An operational data store is a staging environment for storing and preparing data for operational and analytical uses. At this stage, the main data transformation processes are performed, which can be useful especially for ensuring data integrity and consistency. This may involve data cleansing, validation and standardization.

**ODS - Employees**

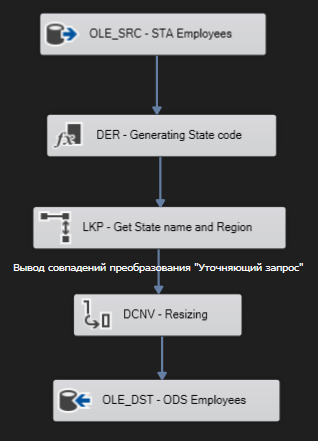
Like before, we truncate the data from the previous runs at control flow level.



Control Flow (ODS Employees)

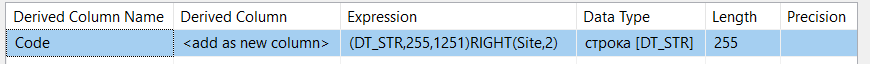
All of ODS packages have similar design pattern with some modifications that is why the procedure may be seemed identical.

To start with, we used our “STA Employees” as the source and started to transform its data. The dataflow of “STA Calls” was defined as follows:



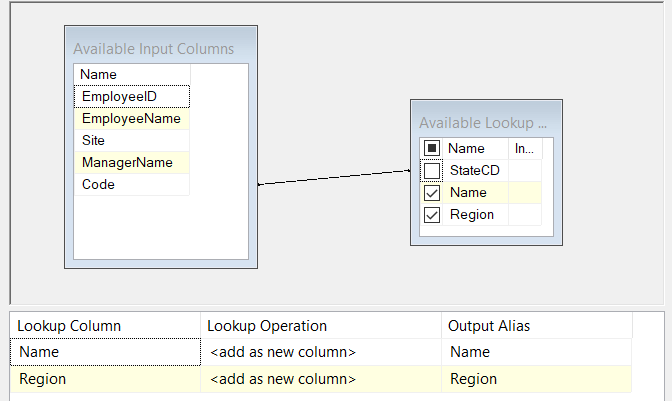
Data Flow (ODS Employees)

A new column called Code was generated using Expression, which allowed to derive separated country code in “Site” column of a primary “Employees” table stored in STA database.

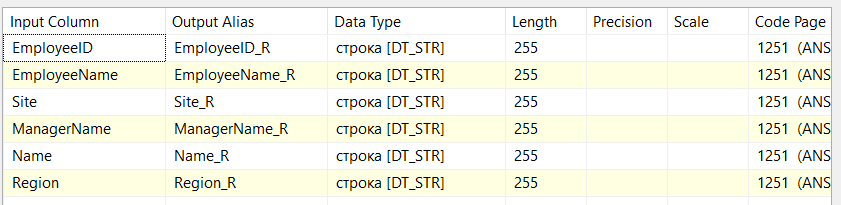


Derived Column (ODS Employees)

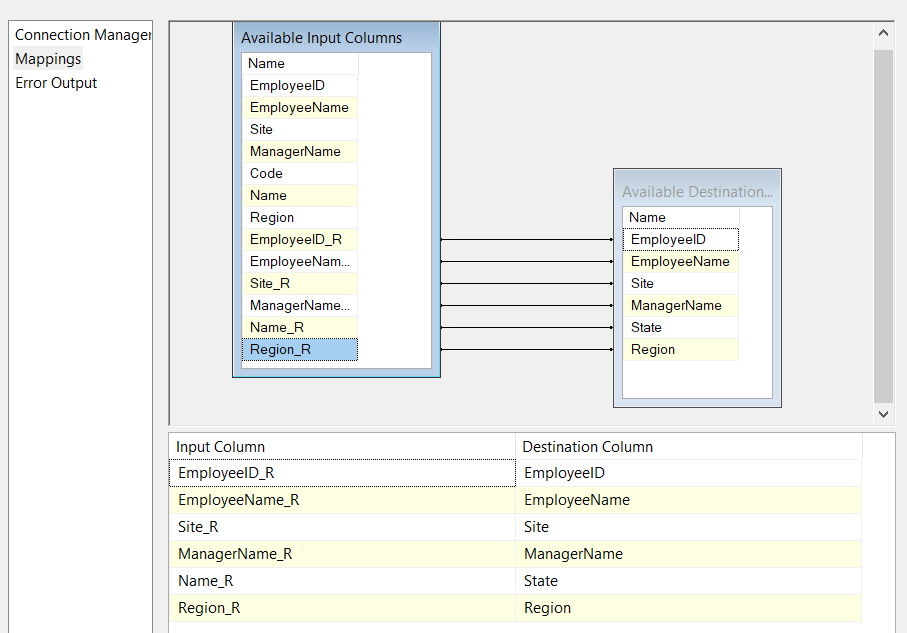
After that, the Lookup tool was used to append the data (Name and Region) from “STA US States” by matching country code, which was generated earlier. Then we renamed and resized table attributes with Data Conversion tool. The last step was to juxtapose these attributes with destination columns. The final outcome of “ODS Employees” is presented below as SQL query results.



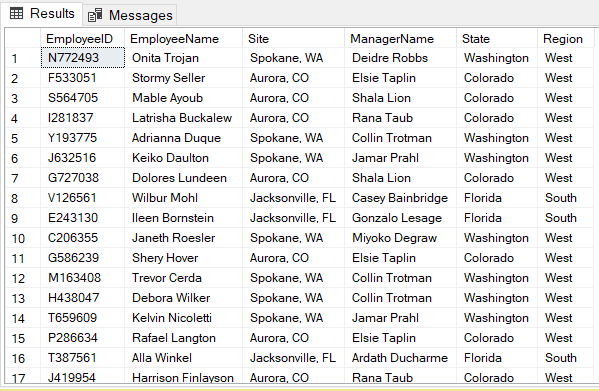
Lookup (ODS Employees)



Data Conversion (ODS Employees)



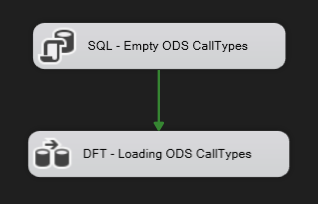
OLE DB Destination (ODS Employees)



SQL Query Results (ODS Employees)

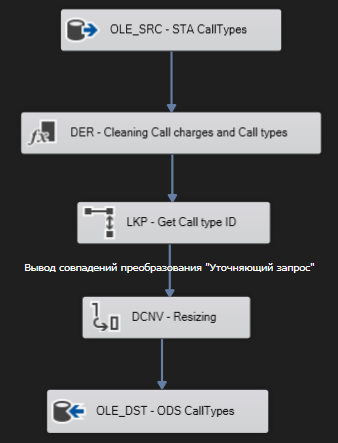
**ODS - Call types**

As it was mentioned before, “STA CallTypes” undergoes the same procedure as “STA Employees”.

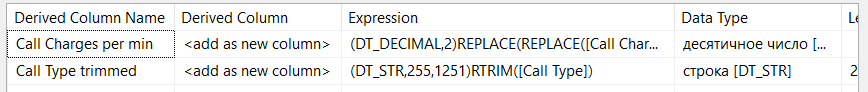


Control Flow (ODS CallTypes)

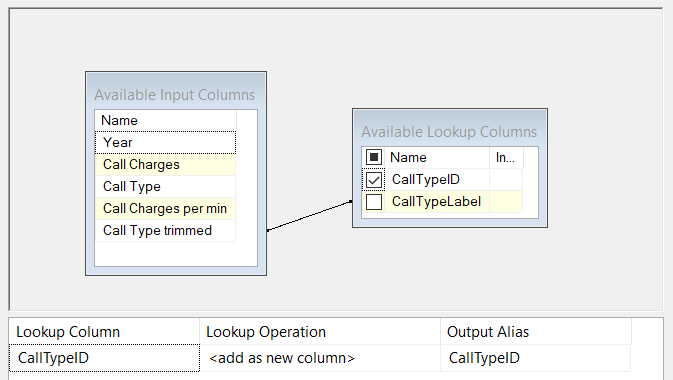
In this package we removed “/min” part in the “CallCharges” column and eliminated spaces in the “CallType” column. Then we added CallTypeID from “STA CallTypes” by matching CallTypeLabel. Next, the data conversion was performed by resizing and renaming table attributes. Finally, all transformations which were executed in “ODS CallTypes” package resulted in a new table which was named “CallTypes”. The final outcome of “ODS CallTypes” is presented below as SQL query results.



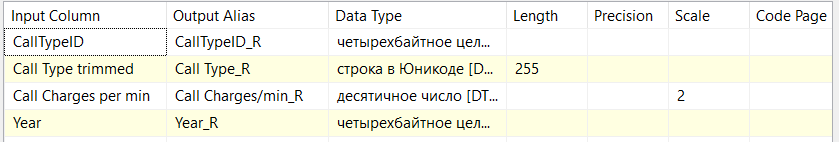
Data Flow (ODS CallTypes)



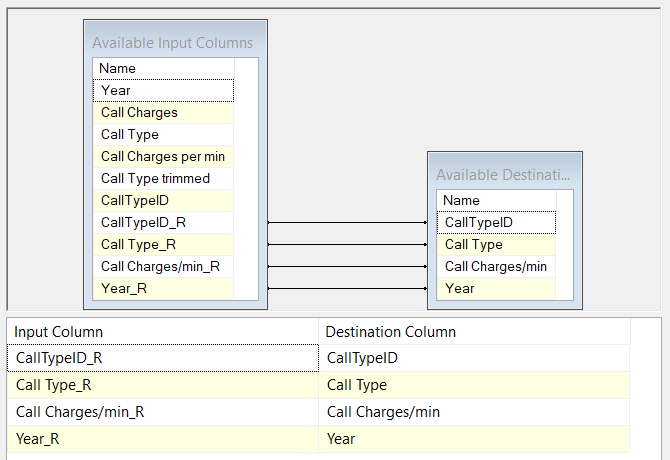
Derived Column (ODS CallTypes)



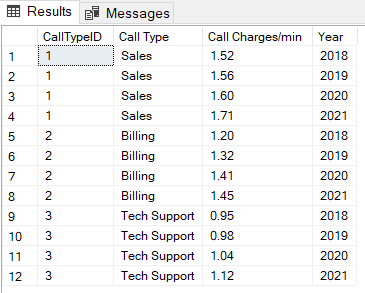
Lookup (ODS CallTypes)



Data Conversion (ODS CallTypes)



OLE DB Destination (ODS CallTypes)



SQL Query Results (ODS CallTypes)

**ODS - Calls data**

To avoid repetitions, here we will focus only on the main differences. Firstly, the “Call Timestamp” column was transformed; it became separated into “time” data and “date” data. Secondly, SLA (Service Level Agreement) was implemented by using the expression where data type was defined as Boolean. Probably it might be an incorrect approach as “ODS Calls” table will a basis for the “Fact table”, which should be mostly made of numerical columns. Although I suppose that, it would be more suitable to use in this way as Boolean type might provide data analysts with better understanding some patterns related to SLA compliance. That is why “SLA” column was defined as follows:

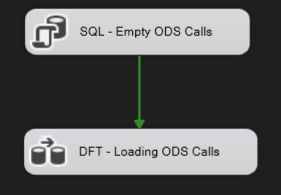
Within SLA (0 = True)

Outside SLA (-1 = False)

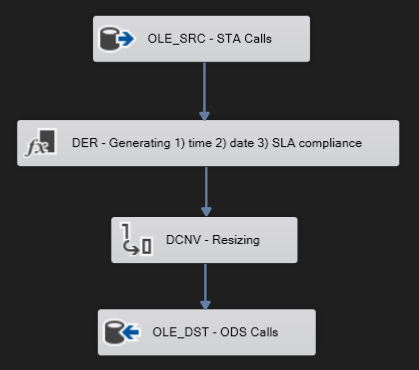
It is also worth to mention that CallAbandoned column was determined as Boolean as well.

Call abandoned: Yes (1 = True),

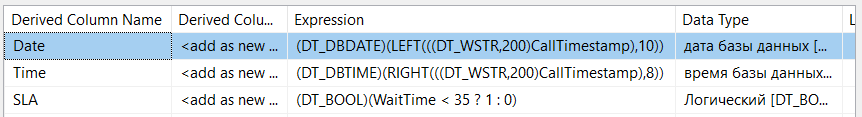
Call abandoned: No (0 = False)



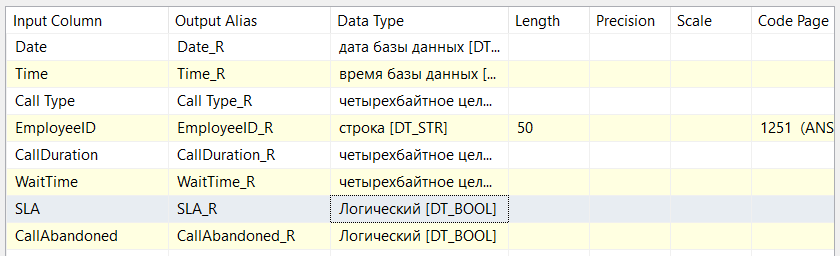
Control Flow (ODS Calls)



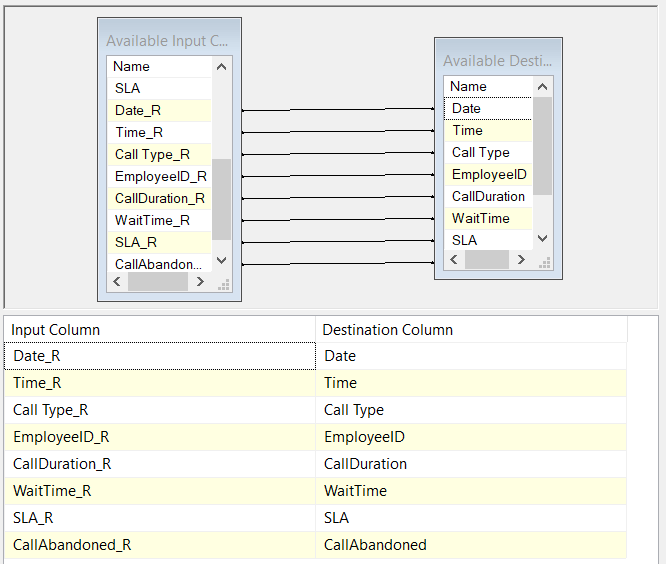
Data Flow (ODS Calls)



Derived Column (ODS Calls)

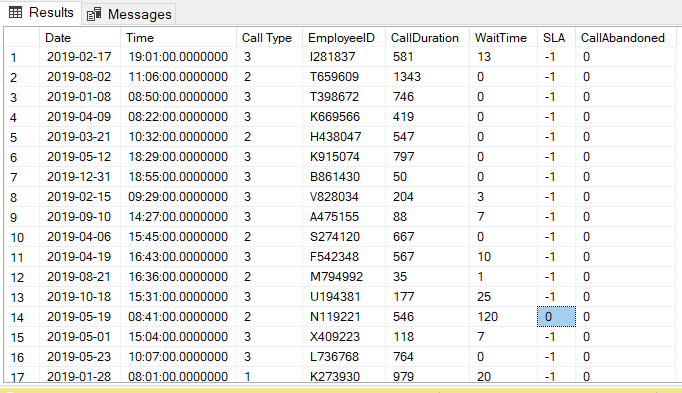


Data Conversion (ODS Calls)



OLE DB Destination (ODS Calls)

The final outcome of “ODS Calls” is presented below as SQL query results.



SQL Query Results (ODS Calls)

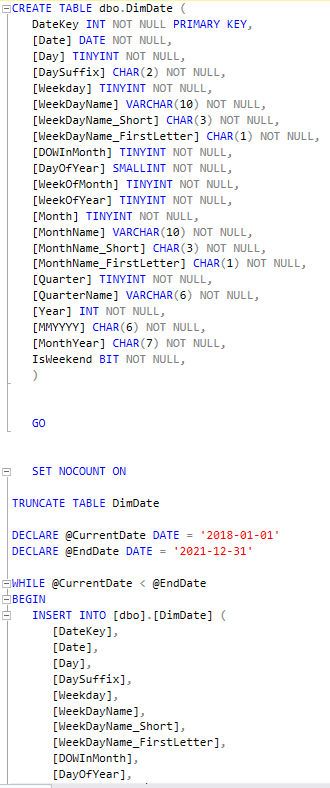
## **3. DATAWAREHOUSE**

The final part of ETL process is to implement the Data Warehouse, which consists of fact tables and dimension tables. These two types of tables should be fed with technical keys. At this stage, the schema starts to form and it usually depends on the number of fact tables. There are two types of DWH schemata: a star and a constellation.

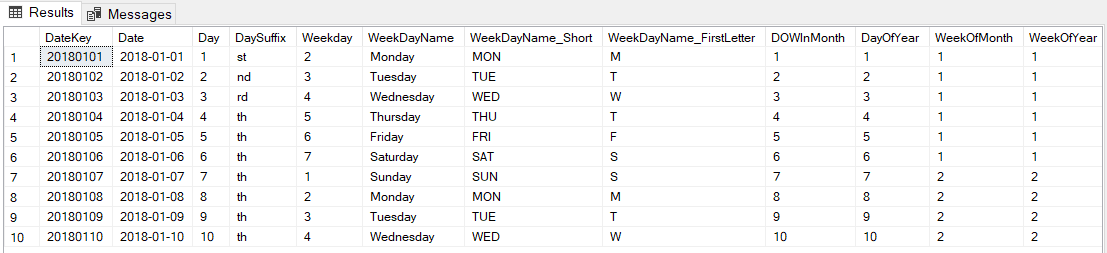
As for this project, we constructed DWH with three dimension tables (DimDate, DimEmployees and DimCallTypes) and one fact table (FactCalls) so we chose a star schema for the database design. All of them will be explained in detail further. In addition, the SCD type1 was set in order to store and manage historical data over time for all cases.

**DWH – DimDate**

This dimension table was generated using SQL script and it contains date formats capable of covering all required date ranges. The SQL code snippet and query results are presented below.



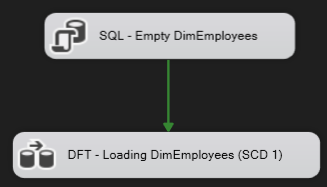
SQL code snippet for DimDate



SQL Query Results (DWH DimDate)

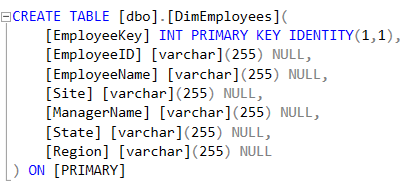
**DWH - DimEmployees**

Like many times before, we truncated the data from the previous runs at control flow level.



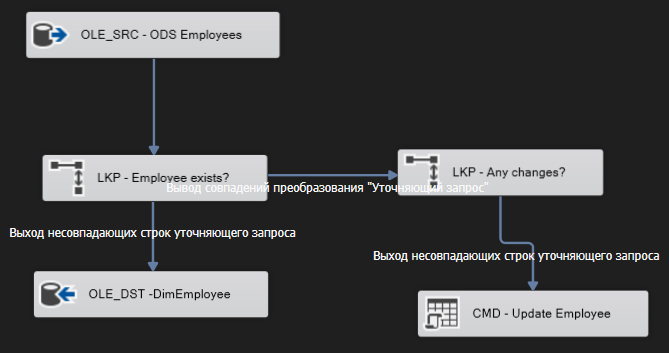
Control Flow (DWH DimEmployees)

Then we created “DimEmployees” table in order to load “ODS Employees” data there using the Lookup tool. It is worth to mention that we intentionally created a technical key. It will allow connecting the dimension table with fact table where the last one will be using it as a foreign key.

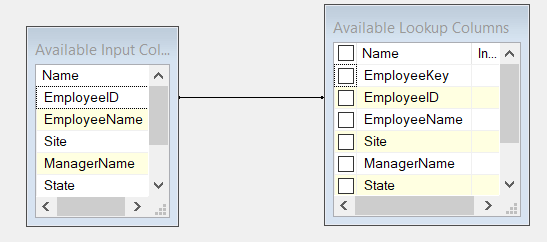


SQL script for DimEmployees

It is also necessary to consider that LKP tool used no match output. At this step, it is important to exclude any duplicates in further exploitation of data.

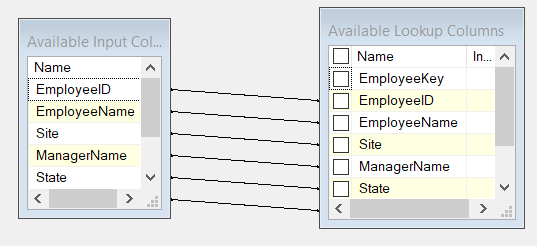


Data Flow (DWH DimEmployees)

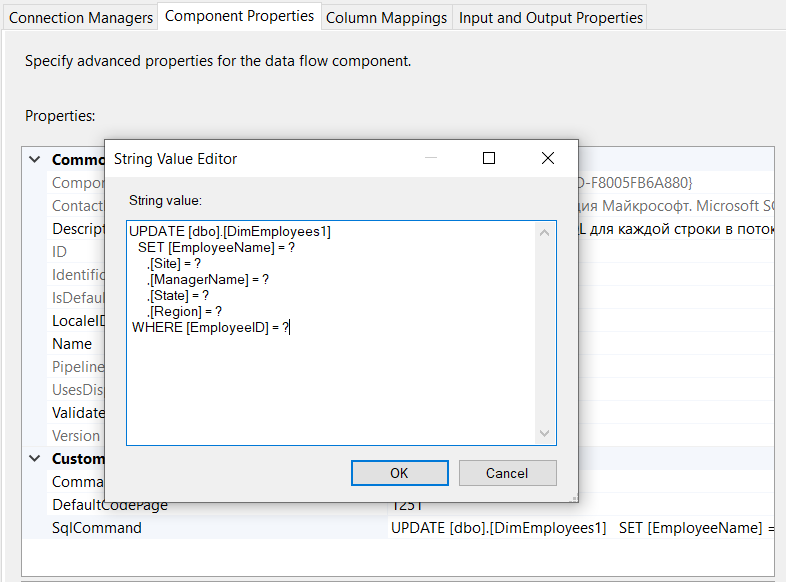


Lookup (Employee exists?)

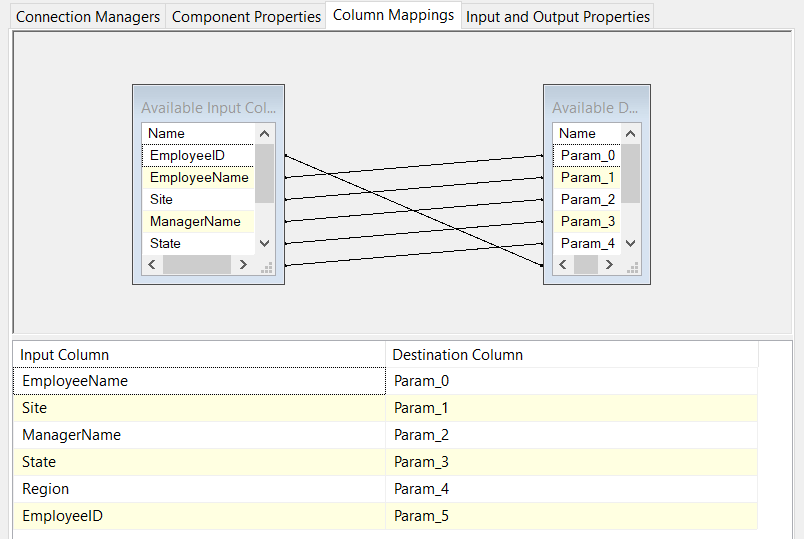
Here we defined an update policy for SCD 1 strategy. It has two operations: checking some changes in data and updating the table if some changes happened



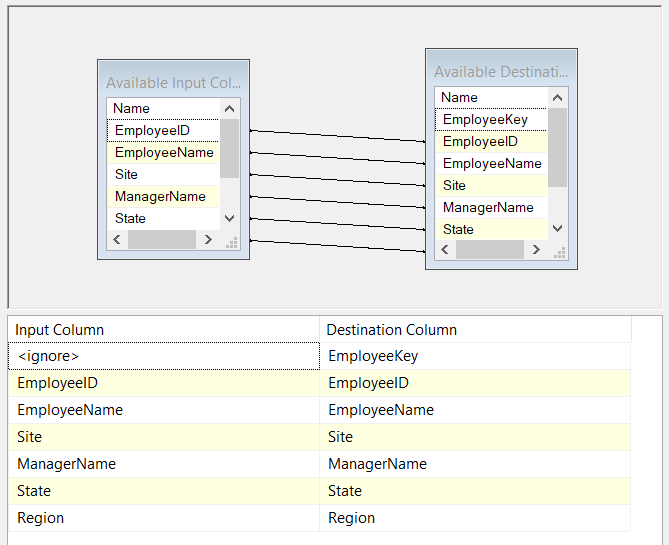
Lookup (Any changes?)



OLE DB Command (DWH DimEmployees)



OLE DB Command (DWH DimEmployees)



OLE DB Destination (DWH DimEmployees)

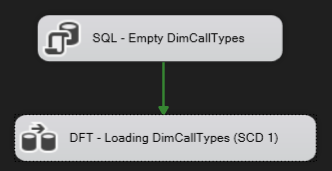
The final outcome of DWH DimEmployees with technical key is presented below as SQL query results.



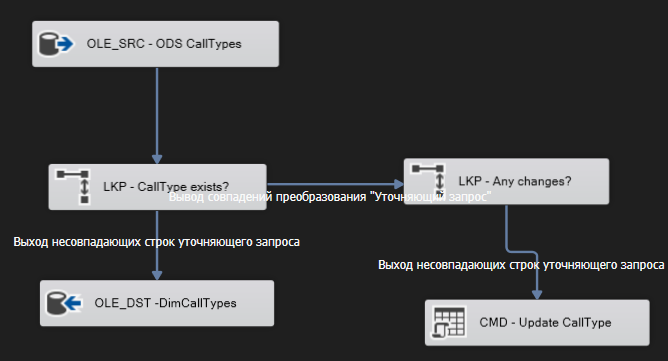
SQL Query Results (DWH DimEmployees)

**DWH – CallTypes**

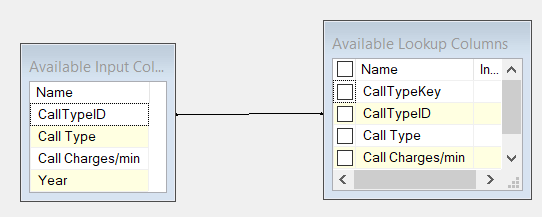
The same actions were conducted for the “CallTypes” dimension table. The control flow and data flow design completely repeat the scheme used on “DimEmployees” table but with generating a primary key for “CallTypes” data. The update policy was also performed with the SCD type1.



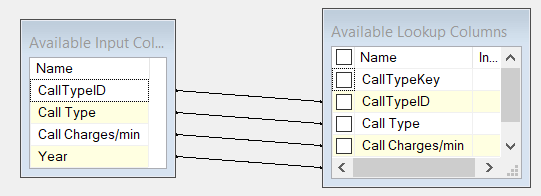
Control Flow (DWH DimCallTypes)



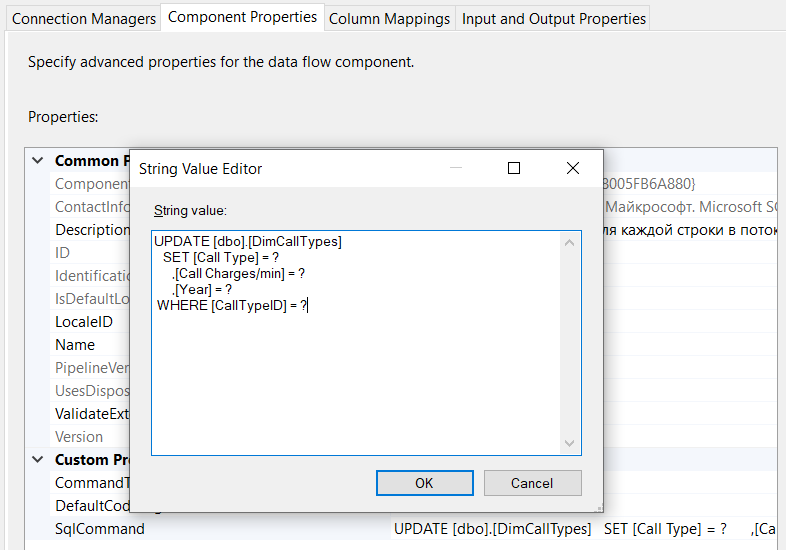
Data Flow (DWH DimCallTypes)



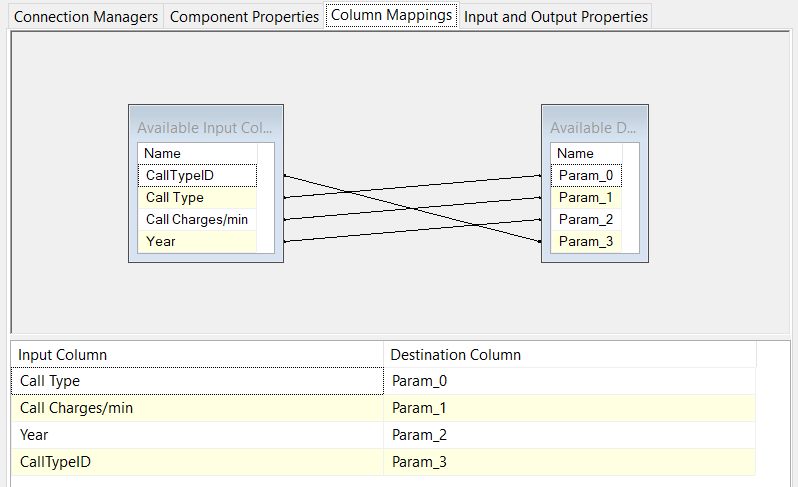
Lookup (CallType exists?)



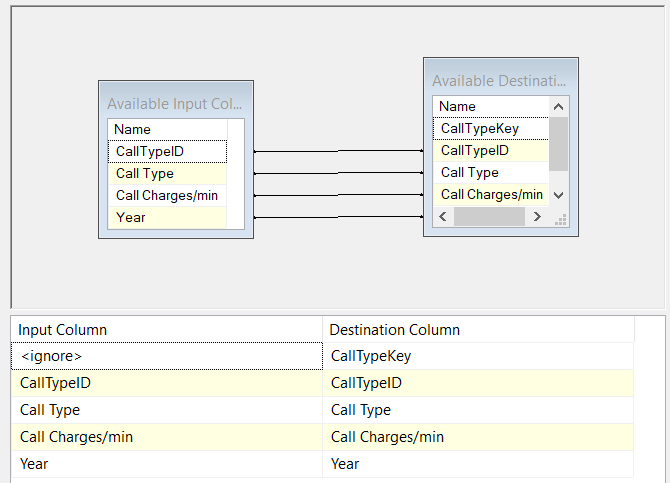
Lookup (Any changes?)



OLE DB Command (DWH DimCallTypes)

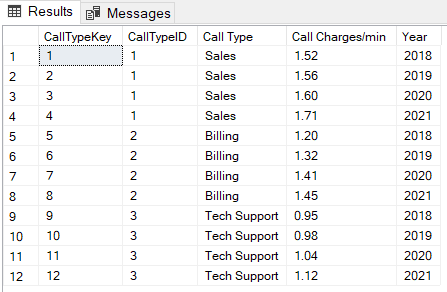


OLE DB Command (DWH DimCallTypes)



OLE DB Destination (DWH DimCallTypes)

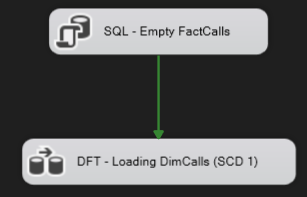
The final outcome of DWH “DimCallTypes” with technical key is presented below as SQL query results.



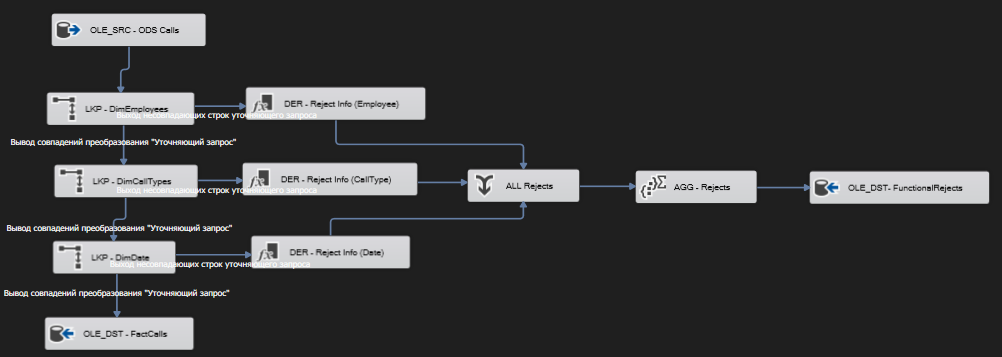
SQL Query Results (DWH DimCallTypes)

**DWH – Calls**

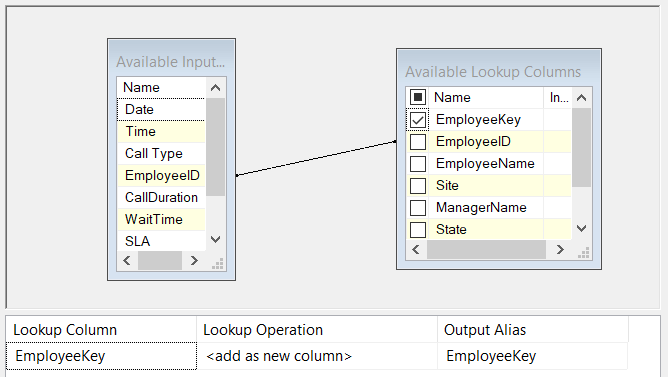
Finally, the last point of DWH implementation is a constructing a fact table, which should be composed of metrics and foreign keys. At the ODS level we already fed the table with required numerical values so all that’s left to do is to extract technical keys from dimension tables using the Lookup tool. This instrument helped to juxtapose the “natural” keys and derive the “surrogate” ones in order to make connection between tables.



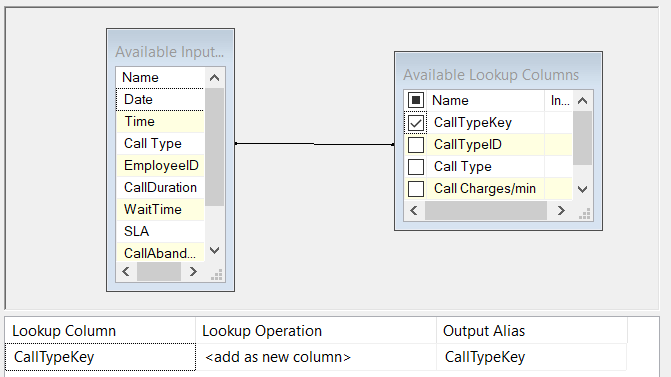
Control Flow (DWH FactCalls)



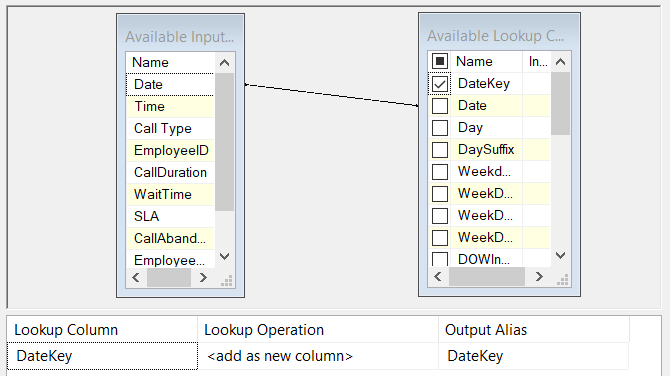
Data Flow (DWH FactCalls)



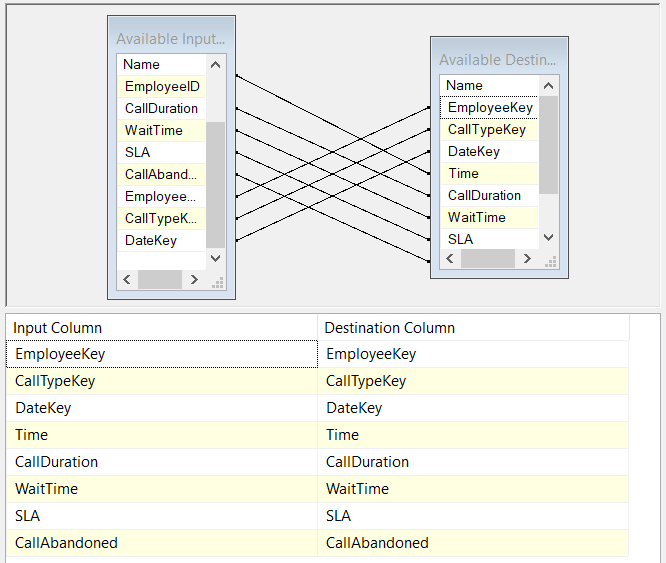
Lookup (DimEmployees)



Lookup (DimCallTypes)

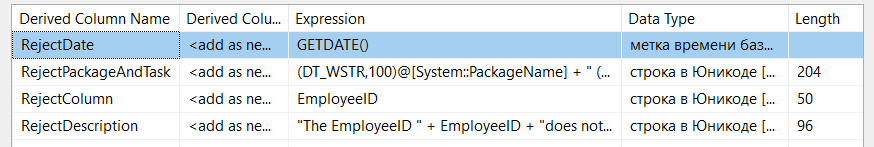


Lookup (DimDate)

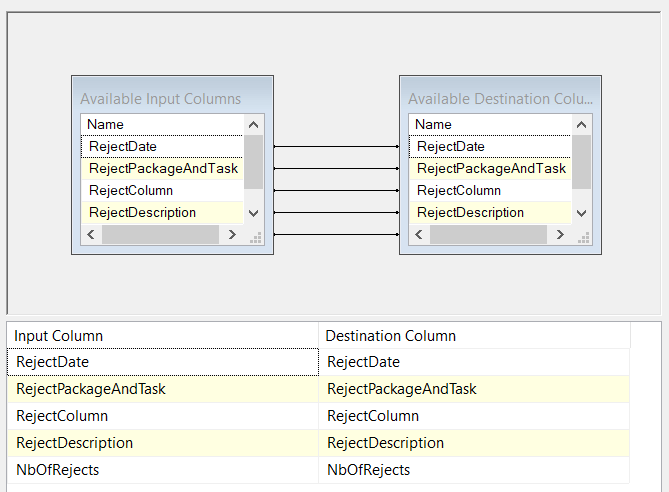
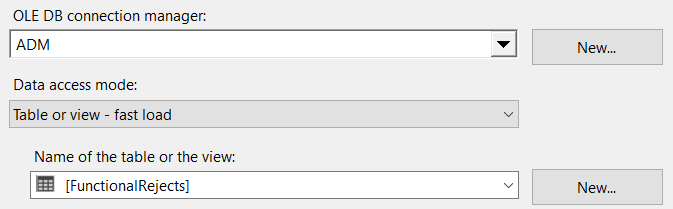


OLE DB Destination (DWH FactCalls)

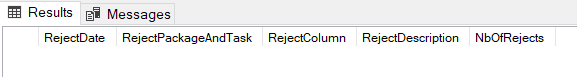
In case of errors in the processing, we generate a functional reject and insert it into a table “Functional Rejects”. It is important to track any troubles caused by data inconsistency. And, luckily for us, as it is seen below, there were no errors in the “Functional Rejects” table.



Derived Column (Reject Info)

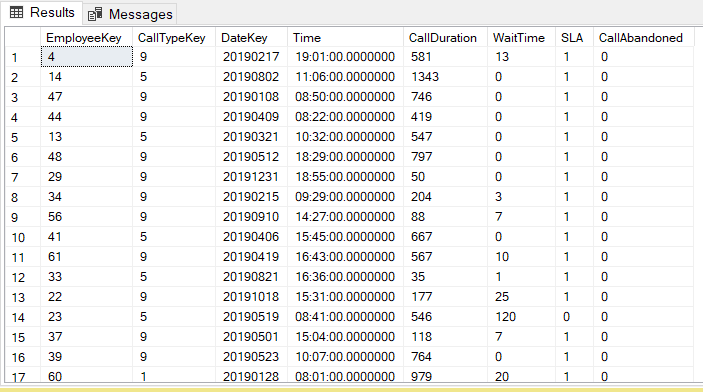


OLE DB Destination (ADM Functional Rejects)



SQL Query Results (ADM Functional Rejects)

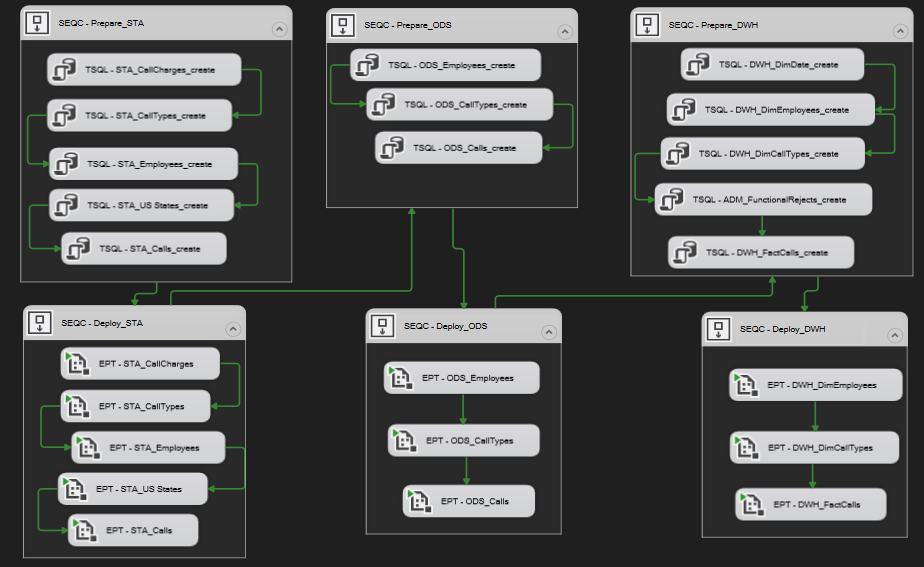
The final outcome of DWH FactCalls with foreign keys is presented below as SQL query results.



SQL Query Results (DWH FactCalls)

## **4. PROJECT DEPLOYMENT**

Once we executed all necessary steps of ETL process, we can finally combine them into a unified ETL Pipeline. For this purpose, all created packages must be inserted into the sequence containers in order to realize them one by one. And it might be obvious but all tables must be created before the package execution tasks launch the data flows.



ETL\_Pipeline

# **III. CONCLUSION**

In this project, we were able to implement and demonstrate the ETL Pipeline that could allow "Service Spot", an IT company, to store their disparate data in the single Data Warehouse. Considering the current maintenance of their database, it can be said that the created Data Warehouse is much more convenient to use. All “call” metrics can be easily analyzed using just one fact table (“FactCalls) which is connected with three very significant dimension tables (“DimEmployees”, “DimCallTypes”) containing only necessary descriptive attributes. That is why it can be said that the building of such a Data Warehouse will definitely help to gain valuable insights from the multiple data sources.

As for the learning experience, it was useful to perform this work for the following reasons:

1. the understanding principles of the ETL process
2. the execution of the ETL Pipeline deployment
3. the learning of database design and schema development
4. the obtainment of hard skills

Overall, this project provided with resources that are highly valuable in the field of data engineering. All these practical knowledge and expertise can be applied in real-world projects in future.