**Caso práctico:** Almacén de datos para el análisis de indicadores de crisis en la eurozona

**PRA2 – Carga de datos**

**Nombre estudiante: Lukaz Martin Doehne**

**Aula 3**

**Índice de contenidos**

[**1. IDENTIFICACIÓN DE LOS PROCESOS DE ETL 3**](#_heading=h.1t3h5sf)

[1.1. Identificación y descripción de los procesos de ETL de ORIGEN a STAGE 3](#_heading=h.olacf13vk40d)

[1.2. Identificación y descripción de los procesos de ETL de STAGE a DW 4](#_heading=h.6l2rsuywtiqt)

[**2. DISEÑO Y DESARROLLO DE LOS PROCESOS DE ETL 5**](#_heading=h.x7kogj8t5der)

[2.1. Transformaciones Bloque IN 11](#_heading=h.4d34og8)

[2.2. Transformaciones Bloque TR\_DIM 26](#_heading=h.wzfuy96690oo)

[2.3. Transformaciones Bloque TR\_FACT 34](#_heading=h.z3hmfw80aix3)

[**3. IMPLEMENTACIÓN TRABAJOS (JOBS) DE LOS PROCESOS DE ETL 38**](#_heading=h.22dnb67q6d4c)

[3.1. Jobs Bloque IN 38](#_heading=h.8c85sbsng2zc)

[3.2. Jobs Bloque TR\_DIM 39](#_heading=h.f68r40ak8h84)

[3.3. Jobs Bloque TR\_FACT 40](#_heading=h.qkop192919xl)

[3.4. Proceso completo (utiliza los jobs IN y TR) 41](#_heading=h.vzkj5scryk7d)

# IDENTIFICACIÓN DE LOS PROCESOS DE ETL

A la hora de diseñar los procesos de carga de una base de datos analítica no hay una única estrategia. Es habitual estructurar los procesos ETL sobre la base de las entidades de datos que se deben actualizar, ya que existen diferencias conceptuales en la actualización de una dimensión con respecto a la de una tabla de hechos. La división del proceso de carga inicial en diferentes bloques de actualización facilitará el diseño de un orden de ejecución y la gestión de las dependencias. Cada uno de estos bloques de actualización se dividirá en las correspondientes etapas de extracción, transformación y carga.

Se identifican los dos bloques siguientes:

● Bloque IN: procesos de carga de los datos desde las fuentes a las tablas intermedias en el área de maniobras (staging area). Estos procesos se distinguen por el prefijo «IN\_» en el nombre.

● Bloque TR: procesos de transformación para cargar los datos desde las tablas intermedias hasta nuestro almacén, según el modelo multidimensional diseñado. Así pues, son diferentes los procesos ETL de transformación para cargar las dimensiones de aquellos que se realizan para cargar las tablas de hechos. Estos procesos se distinguen con el prefijo «TR\_» en el nombre.

## Identificación y descripción de los procesos de ETL de ORIGEN a STAGE

**Bloque IN (de las fuentes a las tablas intermedias)**

| **Nombre del ETL** | **Descripción** | **Orígenes de los datos** | **Tabla destino (stage)** |
| --- | --- | --- | --- |
| IN\_COICOP | Carga de la información relativa al código y descripción de la finalidad de consumo | COICOP.xml | STG\_COICOP |
| IN\_COUNTRY | Contiene los datos de los países  (Nombre y Código ISO) | countries.json | STG\_COUNTRY |
| IN\_CONSUMPTION | Carga de la información  relativa a las distintas franjas de consumo | consumption\_band.csv | STG\_CONSUMPTION |
| IN\_HICP | Contiene la información relativa al indicador HICP | prc\_hicp\_mv12r\_tabular.tsv | STG\_HICP |
| IN\_PGAS | Carga de la información relativa a la evolución del mercado energético europeo de gas. | nrg\_pc\_202\_tabular.tsv | STG\_PGAS |
| IN\_PELEC | Carga de la información relativa a la evolución del mercado energético europeo de la electricidad. | nrg\_pc\_204\_tabular.tsv | STG\_PELEC |

## Identificación y descripción de los procesos de ETL de STAGE a DW

**Bloque TR (de las tablas intermedias a nuestro almacén)**

El bloque TR de procesos ETL para poblar el modelo multidimensional del almacén tiene dos partes diferenciadas. Por un lado, los procesos de carga y transformación de las dimensiones y, por otro, los de las tablas de hechos. El orden de ejecución es importante para que la carga de datos sea la correcta. Las dimensiones se cargarán primero y, después, las tablas de hechos para que no haya errores durante la carga.

| **Nombre del ETL** | **Descripción** | **Tabla de origen** | **Tabla de destino (dimensión)** |
| --- | --- | --- | --- |
| TR\_DIM\_DATE | Carga y transformación de la dimensión temporal | SQL | DIM\_DATE |
| TR\_DIM\_COUNTRY | Carga de la dimensión con información de los países | STG\_COUNTRY | DIM\_COUNTRY |
| TR\_DIM\_COICOP | Carga y transformación de la dimensión con información del código y descripción de la finalidad de consumo | STG\_COICOP | DIM\_COICOP |
| TR\_DIM\_DATE\_SEMESTER | Carga y transformación de la dimensión con información temporal de los precios de la energía | STG\_Gas  STG\_Electricity | DIM\_DATE\_SEMESTER |
| TR\_DIM\_PRODUCT | Carga de la dimensión con información de diferentes tipos de energías | STG\_Gas  STG\_Electricity | DIM\_PRODUCT |
| TR\_DIM\_TAX | Carga de la dimensión con información de los diferentes tipos de impuestos | STG\_Gas  STG\_Electricity | DIM\_TAX |
| TR\_DIM\_CURRENCY | Carga de la dimensión con información de las distintas monedas | STG\_Gas  STG\_Electricity | DIM\_CURRENCY |
| TR\_DIM\_UNIT | Carga de la dimensión con información de las distintas unidades de medida | STG\_Gas  STG\_Electricity | DIM\_UNIT |
| TR\_DIM\_CONSUMPTION | Carga de la dimensión con información de las distintas franjas de consumo | STG\_Gas  STG\_Electricity | DIM\_CONSUMPTION |

Los procesos del bloque de carga y transformación de las tablas de hechos son:

| **Nombre ETL** | **Descripción** | **Tabla origen** |
| --- | --- | --- |
| TR\_FACT\_EUROZONE\_INDICATORS | Carga y transformación de la tabla de hechos «FACT\_EUROZONE\_INDICATORS» | STG\_HICP  STG\_COICOP  STG\_Country |
| TR\_FACT\_ENERY\_PRICE | Carga y transformación de la tabla de hechos «FACT\_ENERY\_PRICE» | STG\_Consumption  STG\_Gas  STG\_Electricity  STG\_Country |

Existen otras estrategias válidas que nos permitirán cargar los datos, ya sea organizando los procesos de otra forma o fusionándolos en un único proceso que lleve a cabo todas las tareas.

# DISEÑO Y DESARROLLO DE LOS PROCESOS DE ETL

En este apartado, se deben diseñar los procesos de carga identificados en el punto anterior con la herramienta de diseño proporcionada. En este caso es Pentaho Data Integration (PDI).

**Creación de tablas**

El primer paso para la implementación de los procesos ETL consiste en la creación de las tablas. Esto se llevará a cabo una única vez, mediante scripts, sobre la base de datos proporcionada (en nuestro caso: SQL Server). Se deberán crear las tablas intermedias y las tablas del modelo dimensional de la solución oficial, es decir, las dimensiones y las tablas de hechos. Para hacerlo, se han utilizado los scripts facilitados junto con el enunciado de la PRA2.

A continuación, se muestran los scripts para la generación de las tablas *Staging*:

**IN\_COUNTRY**

--IN\_COUNTRY: Tabla de staging con la información de los países

IF OBJECT\_ID(N'dbo.STG\_COUNTRY', N'U') IS NOT NULL DROP TABLE dbo.STG\_COUNTRY;

CREATE TABLE dbo.STG\_COUNTRY(

name nvarchar(100),

code nvarchar(10) not null

)

**IN\_HICP**

--IN\_HICP: Tabla de staging con los datos relativos al indicador HICP

IF OBJECT\_ID(N'dbo.STG\_HICP', N'U') IS NOT NULL DROP TABLE dbo.STG\_HICP;

CREATE TABLE dbo.STG\_HICP(

freq nvarchar(50) not null,

unit nvarchar(100) not null,

coicop varchar(50) not null,

geo nvarchar(50) not null,

period int not null,

HICP numeric(12,2) null

)

**IN\_PGAS**

--IN\_PGAS: Tabla de staging con los datos relativos al precio del gas

IF OBJECT\_ID(N'dbo.STG\_PGAS', N'U') IS NOT NULL DROP TABLE dbo.STG\_PGAS;

CREATE TABLE dbo.STG\_PGAS(

freq nvarchar(10) not null,

product int not null,

consom int not null,

unit nvarchar(10) not null,

tax nvarchar(10) not null,

currency nvarchar(10) not null,

geo nvarchar(10) not null,

period nvarchar(10) not null,

pgas numeric(12,4) null

)

**IN\_PELEC**

--IN\_PELEC: Tabla de staging con los datos relativos al precio de la electricidad

IF OBJECT\_ID(N'dbo.STG\_PELEC', N'U') IS NOT NULL DROP TABLE dbo.STG\_PELEC;

CREATE TABLE dbo.STG\_PELEC(

freq nvarchar(10) not null,

product int not null,

consom int not null,

unit nvarchar(10) not null,

tax nvarchar(10) not null,

currency nvarchar(10) not null,

geo nvarchar(10) not null,

period nvarchar(10) not null,

pelec numeric(12,4) null

)

**IN\_COICOP**

--IN\_COICOP: Tabla de staging con información de finalidad de consumo

IF OBJECT\_ID(N'dbo.STG\_COICOP', N'U') IS NOT NULL DROP TABLE dbo.STG\_COICOP;

CREATE TABLE dbo.STG\_COICOP(

code nvarchar(25) not null,

description nvarchar(300) null

)

**IN\_CONSUMPTION**

--IN\_CONSUMPTION: Tabla de staging con información de la franja de consumo de la energía

IF OBJECT\_ID(N'dbo.STG\_CONSUMPTION', N'U') IS NOT NULL DROP TABLE dbo.STG\_CONSUMPTION;

CREATE TABLE dbo.STG\_CONSUMPTION(

code nvarchar(25) not null,

description nvarchar(250) null

)

A continuación, se muestran los scripts para la generación de las tablas dimensionales:

**DIM\_DATE**

--DIM\_DATE

CREATE TABLE [dbo].[DIM\_DATE](

[pk\_date] Int NOT NULL,

[Year] Int,

[Mo] Int,

[Quarter] Int,

CONSTRAINT [PK\_DIM\_DATE] PRIMARY KEY CLUSTERED

(

[pk\_date] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**DIM\_DATE\_SEMESTER**

--DIM\_DATE\_SEMESTER

CREATE TABLE [dbo].[DIM\_DATE\_SEMESTER](

[pk\_date\_semester] nvarchar(10) NOT NULL,

[Year] Int,

[Semester] nvarchar(5),

CONSTRAINT [PK\_DIM\_DATE\_SEMESTER] PRIMARY KEY CLUSTERED

(

[pk\_date\_semester] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

**DIM\_COUNTRY**

--DIM\_COUNTRY

CREATE TABLE [dbo].[DIM\_COUNTRY] (

[pk\_country] int NOT NULL,

[code] nvarchar(10),

[country\_name] nvarchar(100),

CONSTRAINT [PK\_DIM\_COUNTRY] PRIMARY KEY CLUSTERED

(

[pk\_country] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_COICOP**

--DIM\_COICOP

CREATE TABLE [dbo].[DIM\_COICOP](

[pk\_coicop] int NOT NULL,

[code] nvarchar(25),

[coicop\_name] nvarchar(300),

CONSTRAINT [PK\_DIM\_COICOP] PRIMARY KEY CLUSTERED

(

[pk\_coicop] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_CURRENCY**

--DIM\_CURRENCY

CREATE TABLE [dbo].[DIM\_CURRENCY](

[pk\_currency] int NOT NULL,

[currency\_name] nvarchar(10),

CONSTRAINT [PK\_DIM\_CURRENCY] PRIMARY KEY CLUSTERED

(

[pk\_currency] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_TAX**

--DIM\_TAX

CREATE TABLE [dbo].[DIM\_TAX](

[pk\_tax] int NOT NULL,

[tax\_name] nvarchar(50),

CONSTRAINT [PK\_DIM\_TAX] PRIMARY KEY CLUSTERED

(

[pk\_tax] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_UNIT**

--DIM\_UNIT

CREATE TABLE [dbo].[DIM\_UNIT](

[pk\_unit] int NOT NULL,

[unit\_name] nvarchar(25),

CONSTRAINT [PK\_DIM\_UNIT] PRIMARY KEY CLUSTERED

(

[pk\_unit] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_PRODUCT**

--DIM\_PRODUCT

CREATE TABLE [dbo].[DIM\_PRODUCT](

[pk\_product] int NOT NULL,

[product\_code] int,

CONSTRAINT [PK\_DIM\_PRODUCT] PRIMARY KEY CLUSTERED

(

[pk\_product] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**DIM\_CONSUMPTION**

--DIM\_CONSUMPTION

CREATE TABLE [dbo].[DIM\_CONSUMPTION](

[pk\_consumption] int NOT NULL,

[code] nvarchar(25),

[consumption\_name] nvarchar(250),

CONSTRAINT [PK\_DIM\_CONSUMPTION] PRIMARY KEY CLUSTERED

(

[pk\_consumption] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

Finalmente, se muestran los scripts para la generación de las tablas de hechos:

**FACT\_EUROZONE\_INDICATORS**

--FACT\_EUROZONE\_INDICATORS

CREATE TABLE [dbo].[FACT\_EUROZONE\_INDICATORS] (

[pk\_id] int NOT NULL,

[fk\_date] int,

[fk\_country] int,

[fk\_coicop] int,

[HICP] numeric(12,2),

CONSTRAINT [PK\_FACT\_EUROZONE\_INDICATORS] PRIMARY KEY CLUSTERED

(

[pk\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**FACT\_ENERGY\_PRICE**

--FACT\_ENERGY\_PRICE

CREATE TABLE [dbo].[FACT\_ENERGY\_PRICE] (

[pk\_id] int NOT NULL,

[fk\_date\_semester] nvarchar(10),

[fk\_country] int,

[fk\_currency] int,

[fk\_tax] int,

[fk\_consumption] int,

[fk\_unit] int,

[fk\_product] int,

[price] numeric(12,4)

CONSTRAINT [PK\_FACT\_ENERGY\_PRICE] PRIMARY KEY CLUSTERED

(

[pk\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

**CLAVES FORÁNEAS**

--Claves foráneas

ALTER TABLE FACT\_EUROZONE\_INDICATORS ADD FOREIGN KEY (fk\_date) REFERENCES DIM\_DATE(pk\_date);

ALTER TABLE FACT\_EUROZONE\_INDICATORS ADD FOREIGN KEY (fk\_country) REFERENCES DIM\_COUNTRY(pk\_country);

ALTER TABLE FACT\_EUROZONE\_INDICATORS ADD FOREIGN KEY (fk\_coicop) REFERENCES DIM\_COICOP(pk\_coicop);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_date\_semester) REFERENCES DIM\_DATE\_SEMESTER(pk\_date\_semester);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_country) REFERENCES DIM\_COUNTRY(pk\_country);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_currency) REFERENCES DIM\_CURRENCY(pk\_currency);

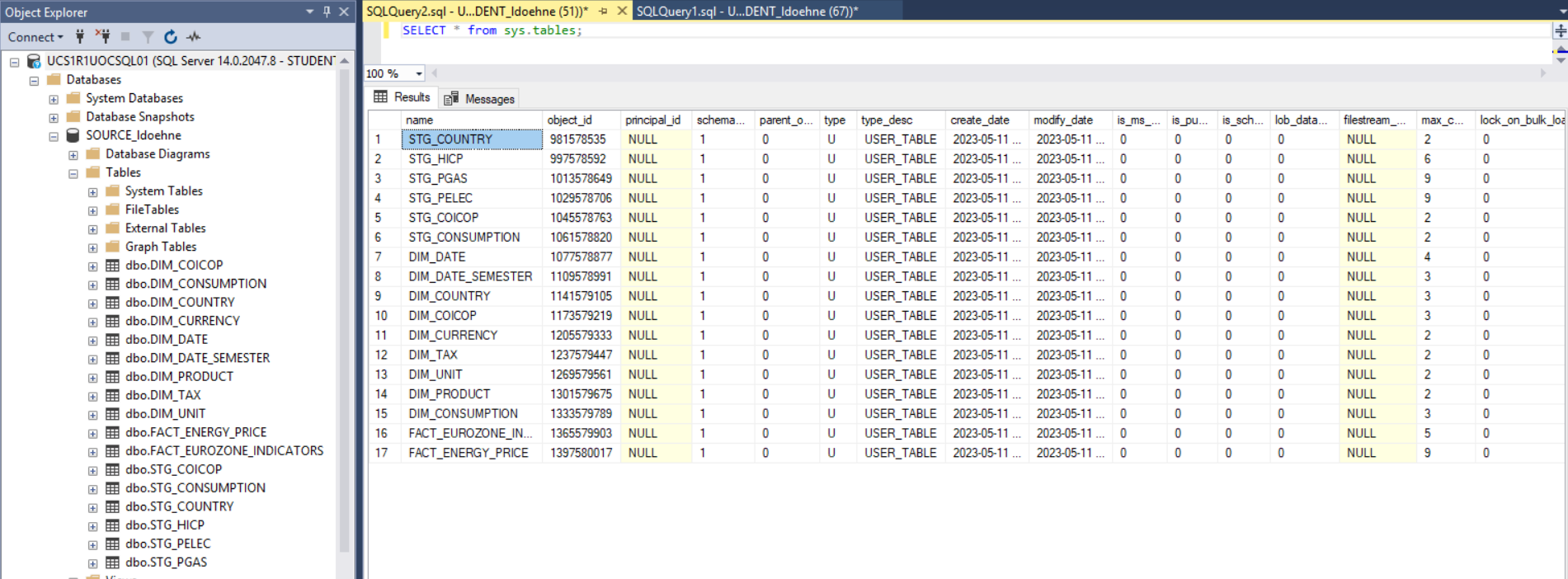
ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_tax) REFERENCES DIM\_TAX(pk\_tax);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_consumption) REFERENCES DIM\_CONSUMPTION(pk\_consumption);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_unit) REFERENCES DIM\_UNIT(pk\_unit);

ALTER TABLE FACT\_ENERGY\_PRICE ADD FOREIGN KEY (fk\_product) REFERENCES DIM\_PRODUCT(pk\_product);

A continuación se muestra una captura de pantalla después de ejecutar los scripts donde se ve la creación de las tablas:



Una vez se tiene implementado el modelo físico del almacén, se pasa a diseñar los procesos de ETL que permitirán poblar las tablas intermedias del área intermedia (staging area) y las tablas de dimensiones y de hechos del data mart que se han diseñado.

## Transformaciones Bloque IN

**Transformación de IN\_COICOP**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_COICOP»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **«COICOP.xml»**, en la tabla **«STG\_COICOP»** del área intermedia (staging area).

La transformación de «IN\_COICOP» contiene las siguientes etapas:

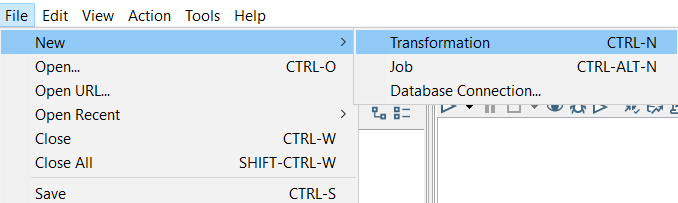
- lectura del fichero .xml

- filtración de valores nulos

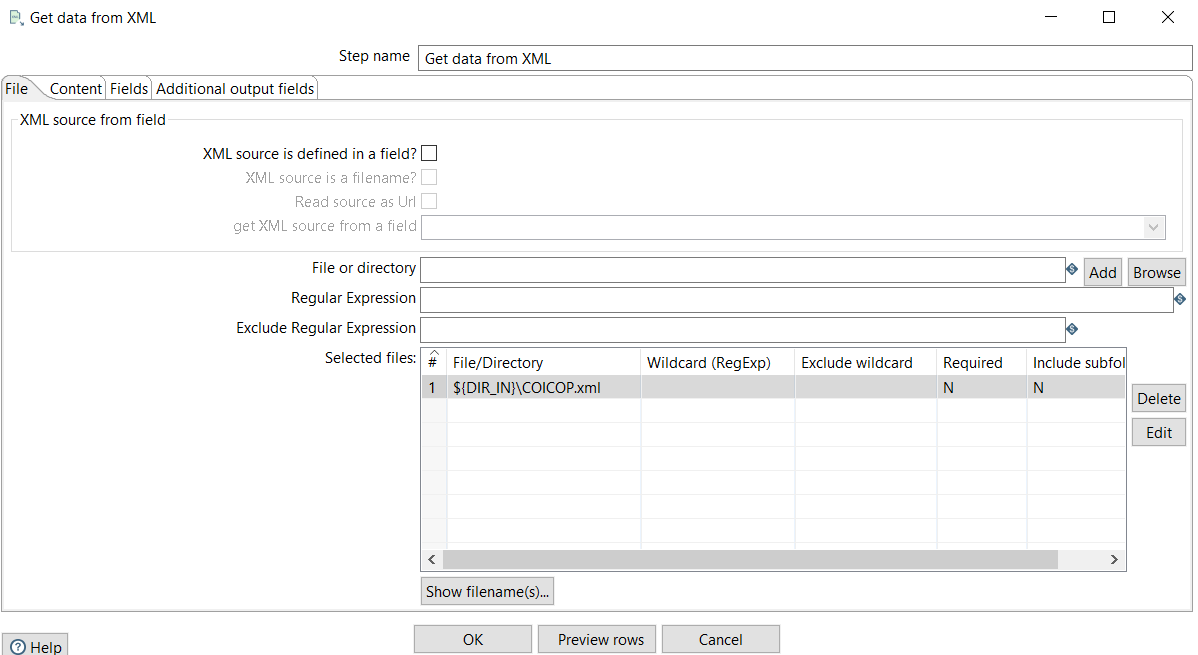
- carga a la tabla intermedia «STG\_COICOP»

A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos.

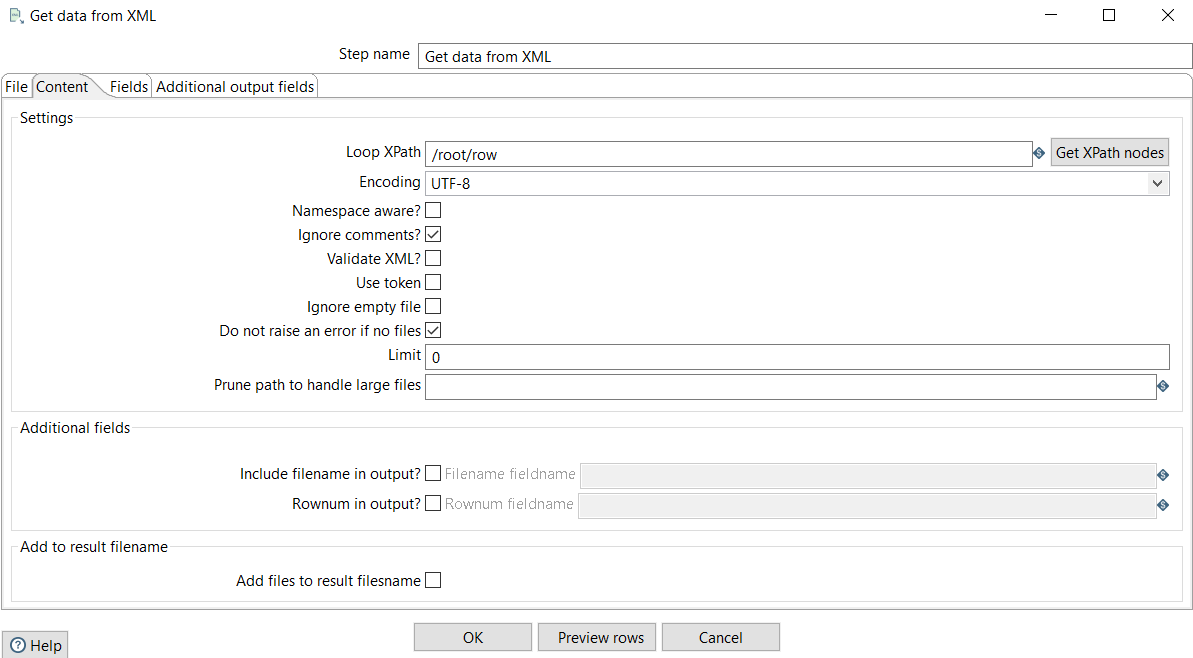
En primer lugar, se crea una nueva transformación.



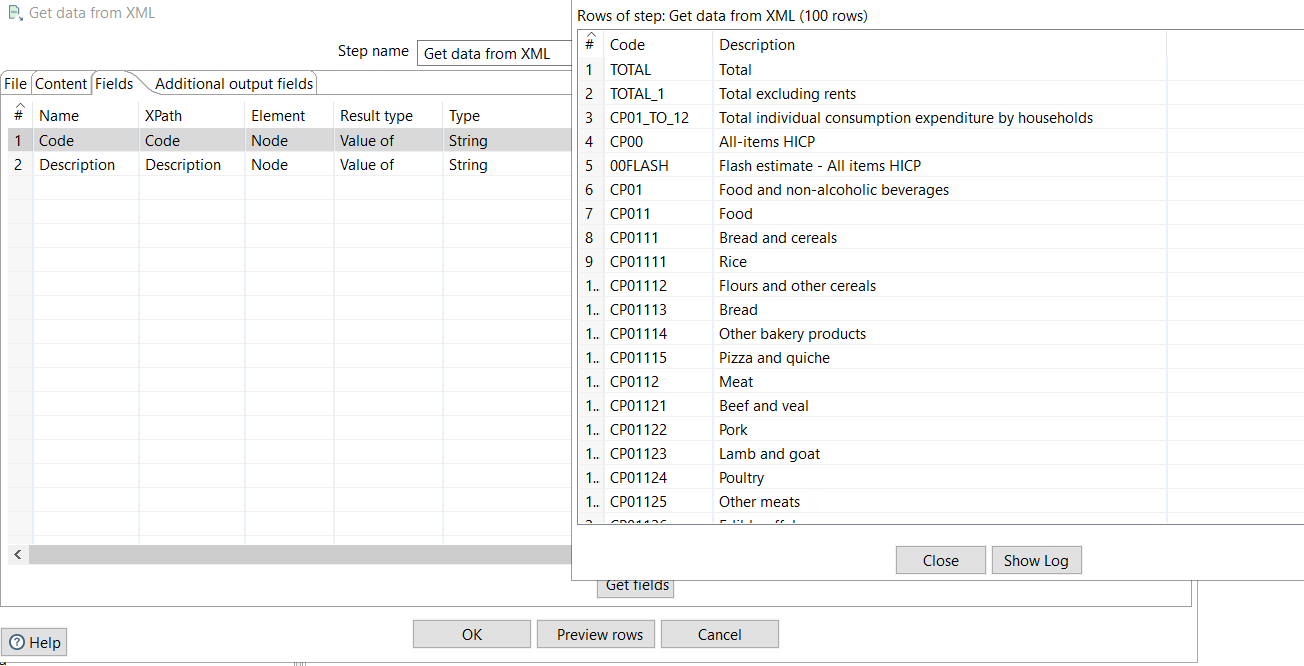
El primer paso será la entrada del fichero XML. Para ello se utiliza la transformación «Get data from XML». En este paso se indica el fichero desde donde se extraen los datos. Para este tipo de carga de datos se puede utilizar la variable de entorno «DIR\_IN» o introducir la ruta directamente.



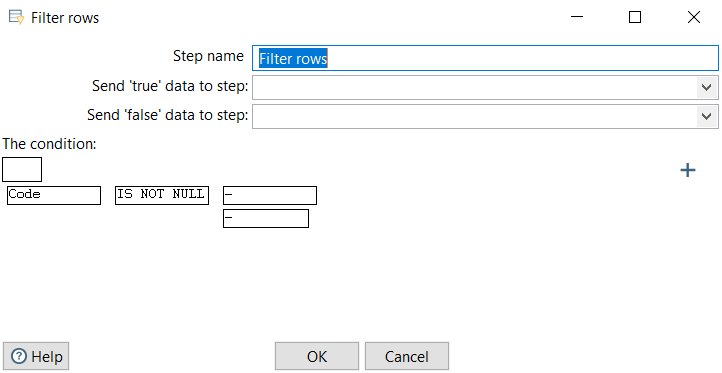
Seguidamente, indicamos el «Loop XPath» e ignoramos los posibles comentarios que haya dentro del fichero.



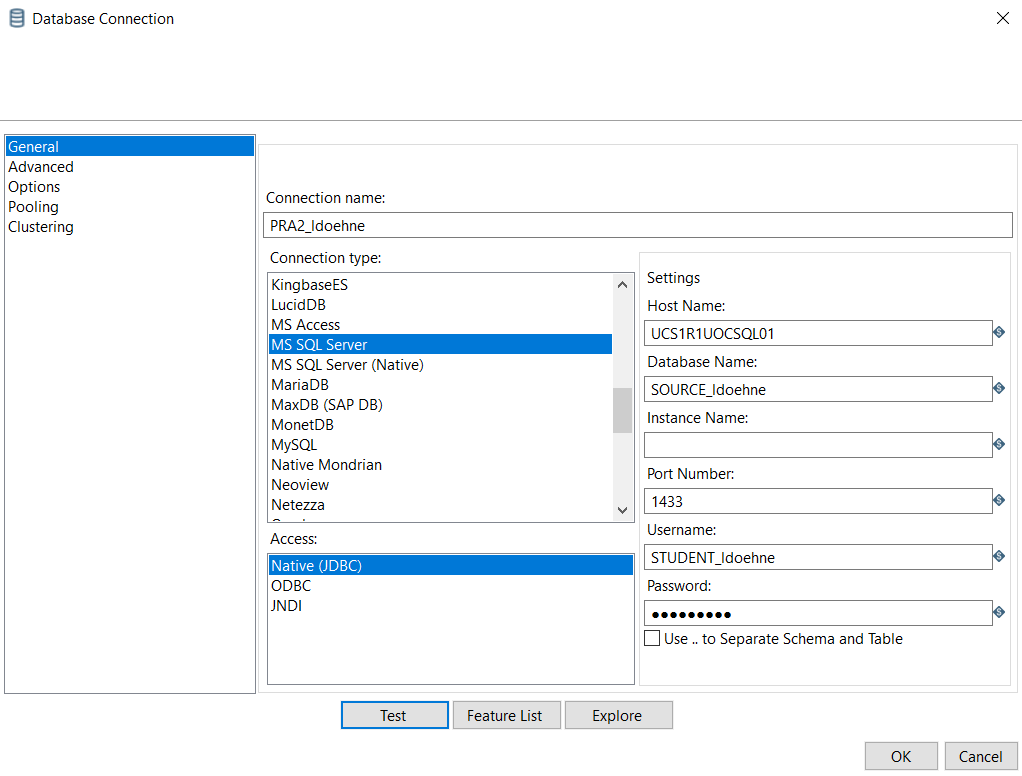
Por último, seleccionamos los campos mediante el botón de «Get fields», y podemos realizar una visualización previa de los datos que se cargarán con el botón «Preview rows».



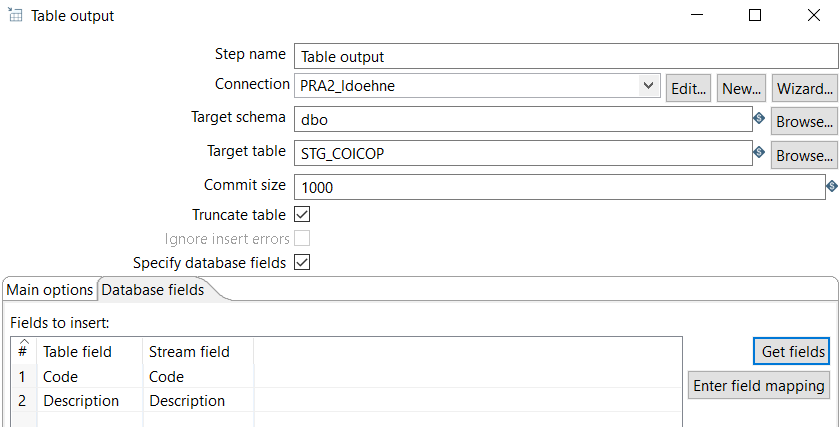
Dado que la tabla que hemos creado en nuestra base de datos SQL («STG\_COICOP») no admite valores nulos, deberemos filtrarlos mediante el paso «Filter rows»



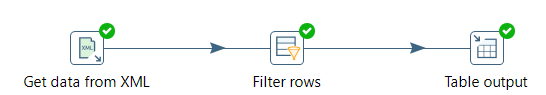
Para enviar los datos a nuestra tabla *Staging* hemos de crear una conexión entre Spoon y Microsoft SQL Server que lo podemos hacer como se observa a continuación:



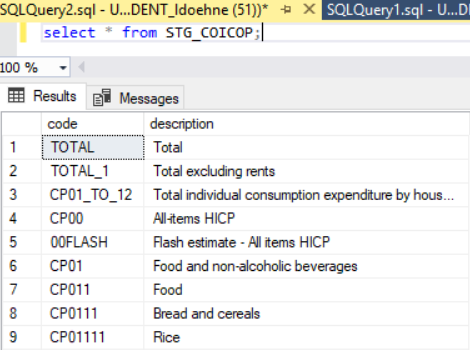
Seguidamente, podemos crear el elemento Table Output, que utilizará la salida de Filter rows para enviarlo a la base de datos STG\_COICOP.



El proceso de la transformación completa es el siguiente:



Y podemos comprobar en la base de datos el resultado del proceso realizado:



**Transformación de IN\_COUNTRY**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_COUNTRY»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **Countries.json**, en la tabla **«STG\_COUNTRY»** del área intermedia (staging area).

La transformación de «IN\_COUNTRY» contiene las siguientes etapas:

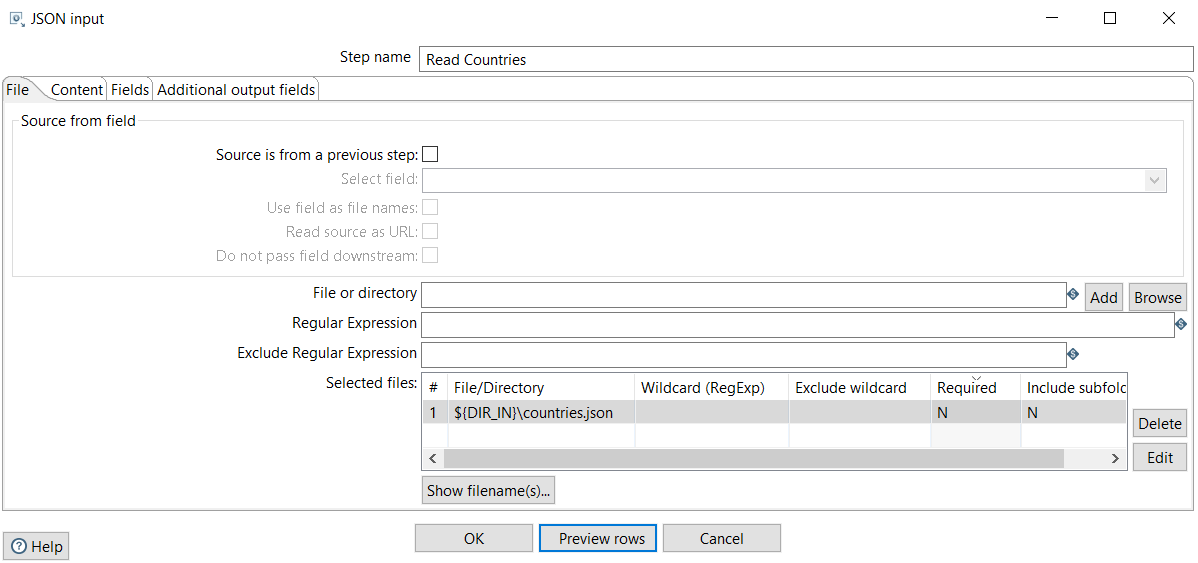
- lectura del fichero .json

- operaciones de String

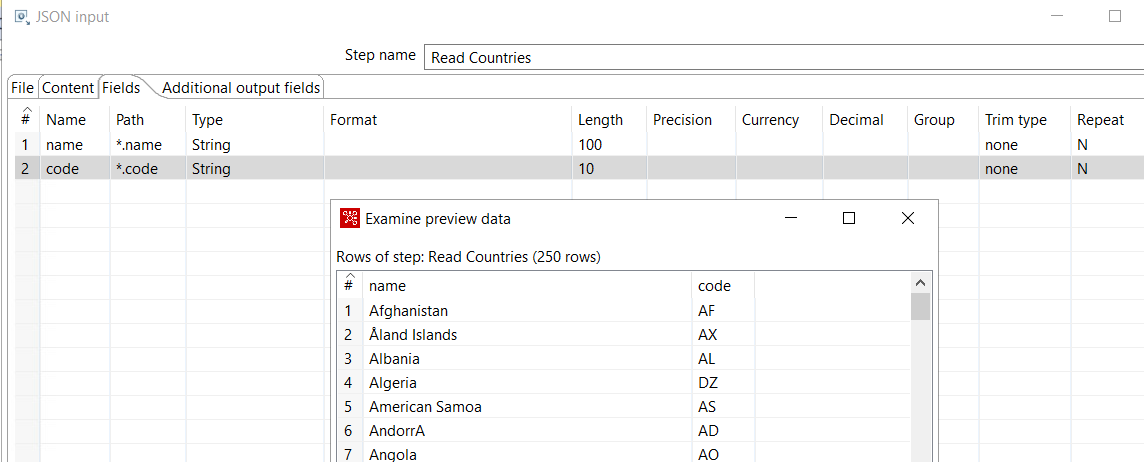
- carga a la tabla intermedia «STG\_COUNTRY»

A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos:

Con el elemento JSON input podemos especificar el archivo a leer.

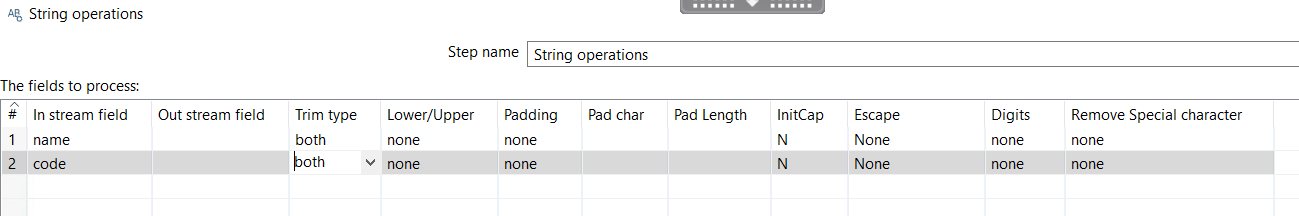


En la pestaña de Fields, seguidamente seleccionaremos los campos *name* y *code* y dándole al botón *Preview rows* podemos asegurarnos de que lee correctamente los atributos.



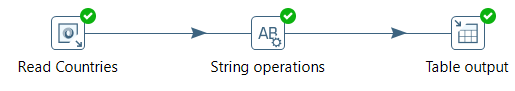
Nota: Es importante destacar que nuestra base de datos *Staging* tiene las columnas *name* y *code* delimitadas a 100 y 10 varchar respectivamente. Es por ello que delimitamos aquí también a los mismos valores.

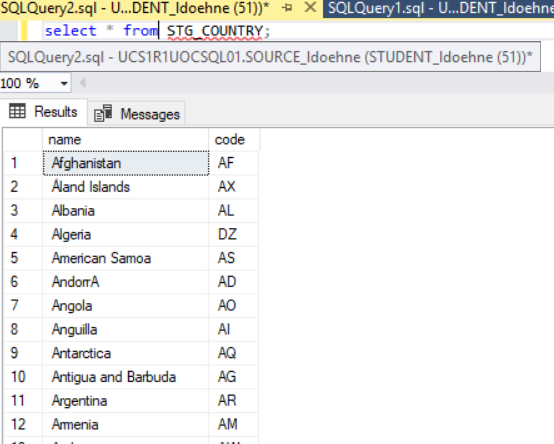
El siguiente paso de la transformación consiste en limpiar las columnas de espacios en blanco anteriores y posteriores, activando *trim* para ambos lados en caso de espacios laterales:



El resultado después de crear el Table Output y ejecutarlo se puede ver en estás capturas de pantalla:

Nota: no enseñaré el proceso de Table output nuevamente ya que es exactamente igual que con el elemento anterior.





**Transformación de IN\_CONSUMPTION**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_CONSUMPTION»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **consumption\_band.csv**, en la tabla **«STG\_CONSUMPTION»** del área intermedia (staging area).

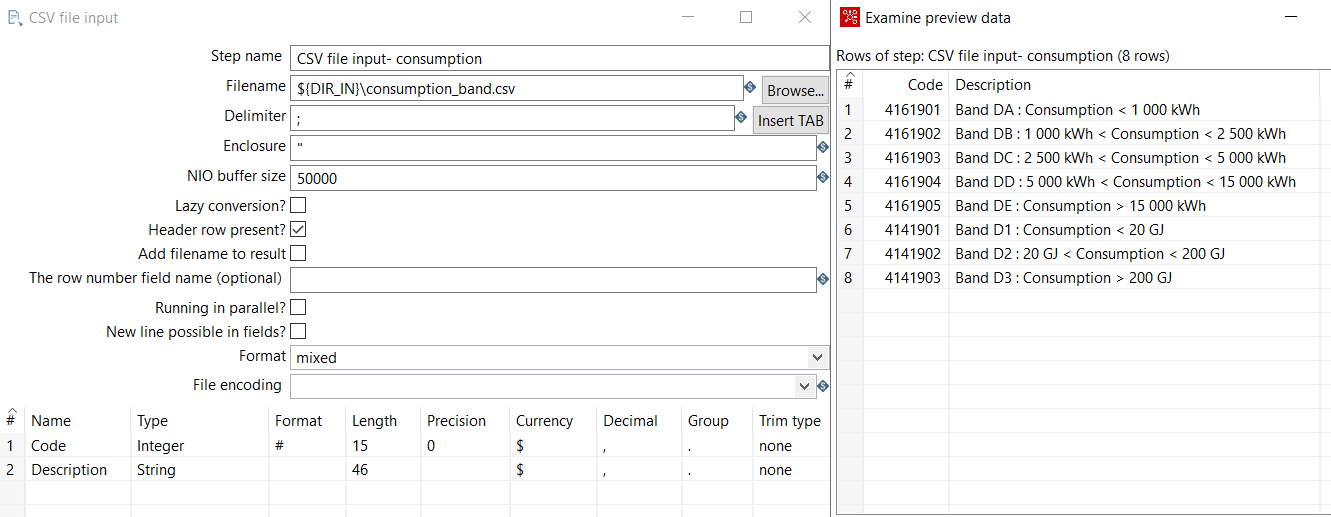
La transformación de «IN\_CONSUMPTION» contiene las siguientes etapas:

- lectura del fichero .csv (separado por punto y coma)

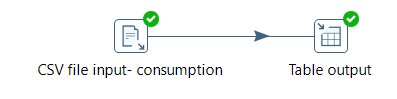
- carga a la tabla intermedia «STG\_CONSUMPTION»

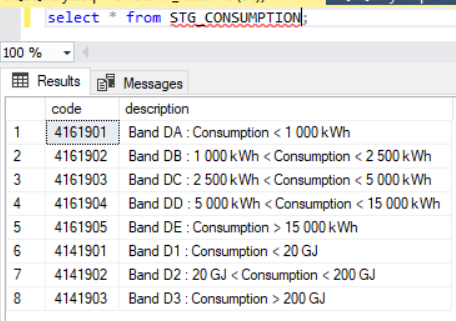
A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos:

Cargamos el fichero csv mediante el elemento CSV file input. Utilizamos la variable de entorno DIR\_IN para dirigirnos a nuestro directorio de datos. El delimitador para este fichero es de punto y coma (;). Podemos visualizar los datos bajo el botón *Preview Rows*.



El resultado de crear la conexión a la base de datos, crear el elemento Table output y ejecutar la transformación lo vemos en las siguientes capturas:

****

****

**Transformación de IN\_HICP**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_HICP»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **prc\_hicp\_mv12r\_tabular.tsv**, en la tabla **«STG\_HICP»** del área intermedia (staging area).

La transformación de «IN\_HICP» contiene las siguientes etapas:

- lectura del fichero .tsv (separado por comas y tabuladores)

- separar columnas correctamente

- convertir columna de tiempo en filas

- reemplazar caracteres de los strings

- eliminar espacios en blanco

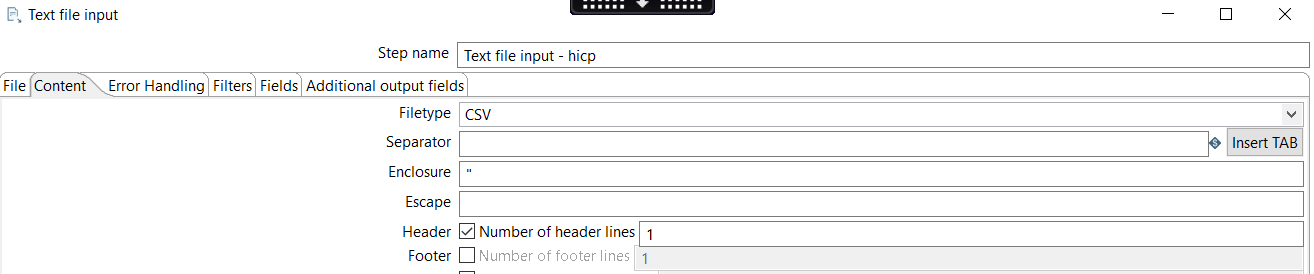
- convertir columnas en enteros y númericos

- eliminar filas con HICP igual a nulo

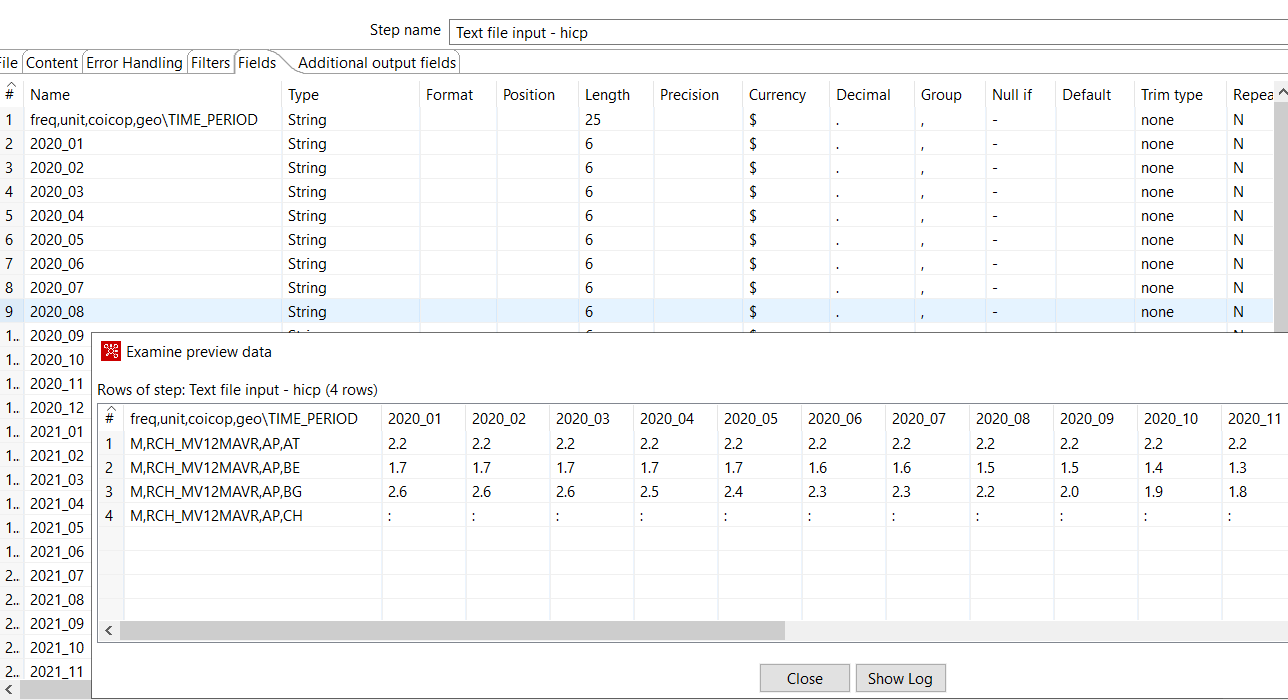
- carga a la tabla intermedia «STG\_HICP»

A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos:

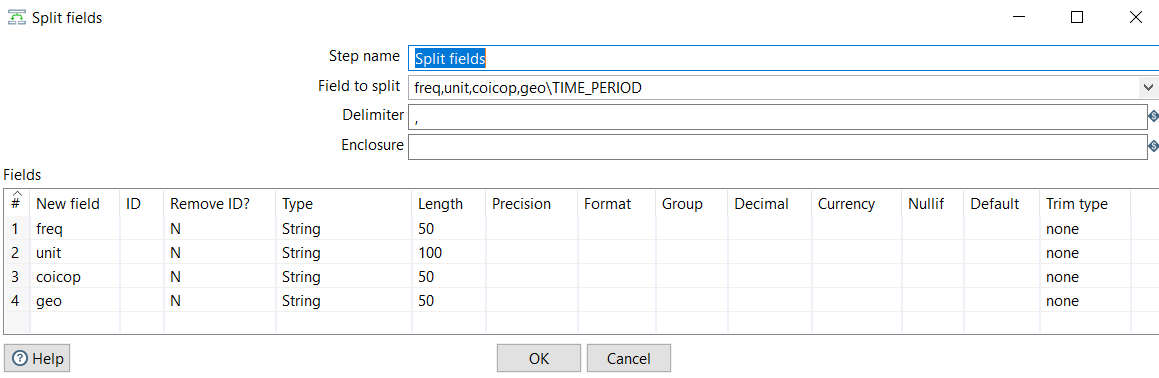
Empezamos leyendo el fichero con Text file input. Escogemos como Filetype CSV (aunque sea tsv) y seleccionamos como separador tab (podemos darle al botón insert TAB).

****

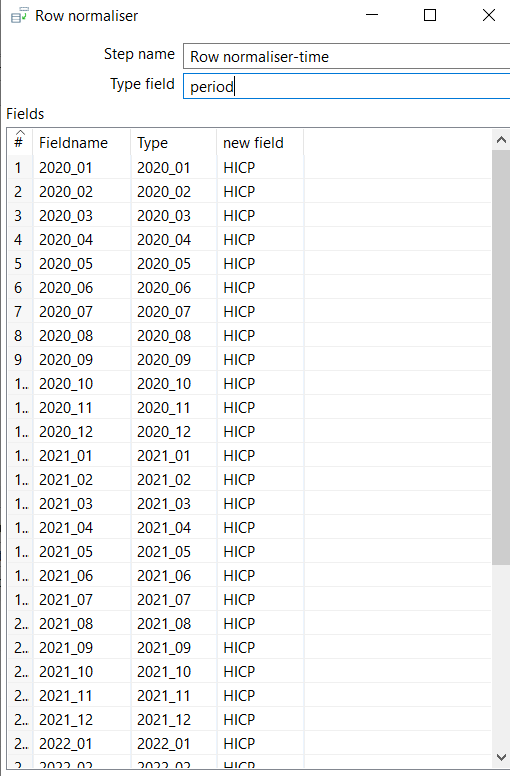
Con esto conseguimos separar las variables de tiempo (separadas por tabulador) de las variables freq,unit,coicop y geo.

****

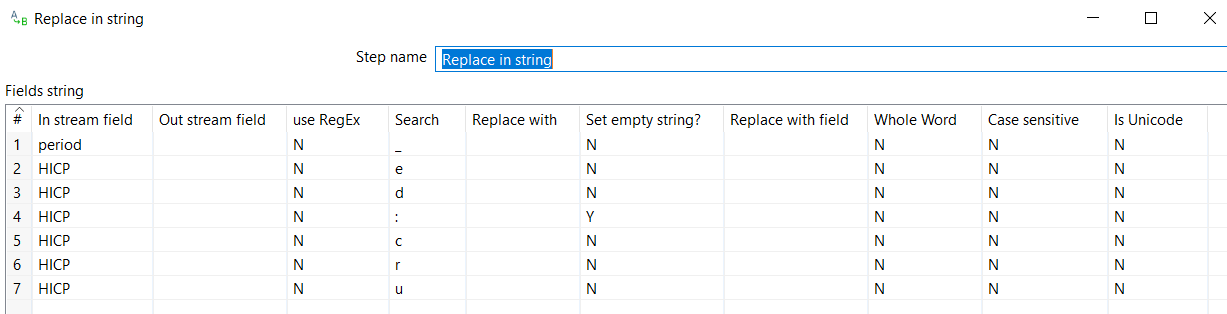
Para dividir la columna “combinada” en 4 diferentes utilizamos el elemento *Split fields*. Usando como delimitador la coma y definiendo las nuevas columnas.

****

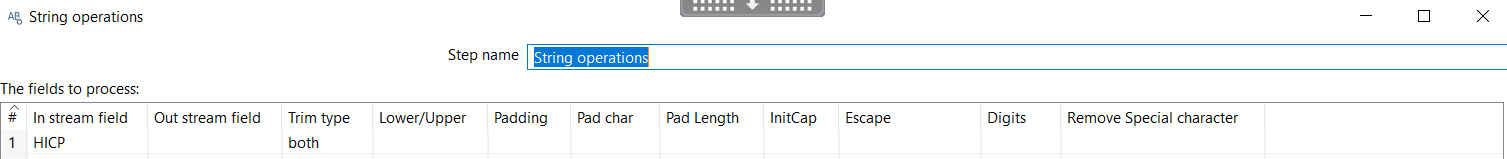
A continuación, queremos transformar las columnas de tiempo en nuevas filas. Para ello utilizaré el elemento *Row normaliser*. Aquí seleccionamos el nombre de la nueva columna en este caso period. El valor que contenía la variable de tiempo ahora lo hemos de renombrar a HICP.



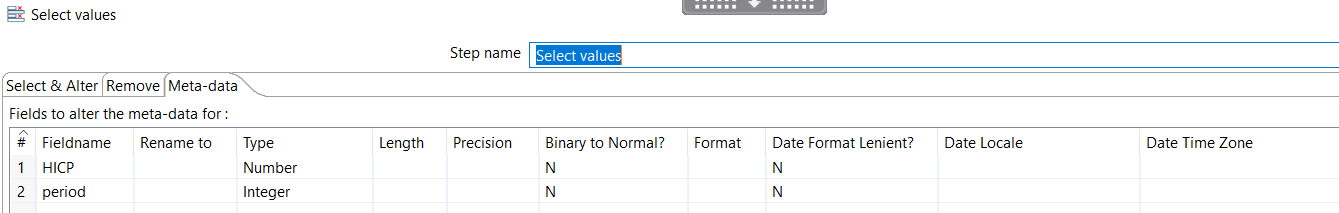
Como podemos observar, la columna period tiene los valores del tipo año\_mes (AAAA\_MM). Cómo queremos que sea un número integer y siguiendo la guía se espera un valor AAAAMM sin la barra baja usaremos el elemento *Replace in String*. Además, quitamos las letras d,e,c,r,u de HICP y los valores vacíos. Si se hubiera tratado de más valores se habría que recurrir a las funciones RegEx.



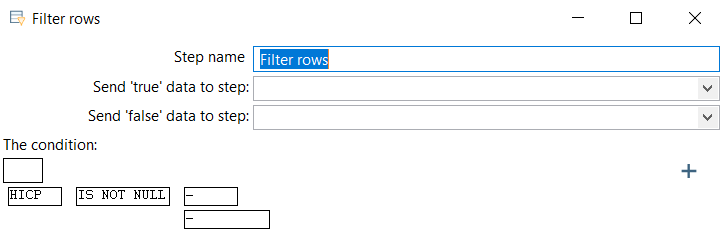
Este proceso nos deja espacios en blanco que queremos eliminar con el paso de *String Operations*.



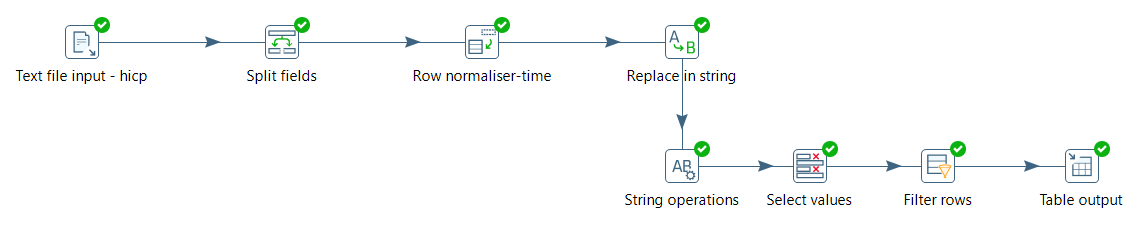
Seguidamente, convertimos las columnas period y HICP a integer y Number, respectivamente con el elemento *Select Values*.

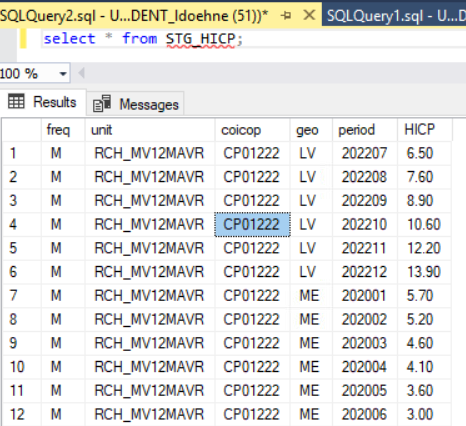


Descartamos las columnas donde HICP es nulo:



Finalmente, creando la conexión, la Tabla de Output y ejecutando la transformación obtenemos el paso de la fuente de datos .tsv a STG\_HICP:





**Transformación de IN\_PGAS**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_PGAS»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **nrg\_pc\_202\_tabular.tsv**, en la tabla **«STG\_PGAS»** del área intermedia (staging area).

La transformación de «IN\_PGAS» contiene las siguientes etapas:

- lectura del fichero .tsv (separado por comas y tabuladores)

- separar columnas correctamente

- convertir columna de tiempo en filas

- reemplazar caracteres de los strings

- eliminar espacios en blanco

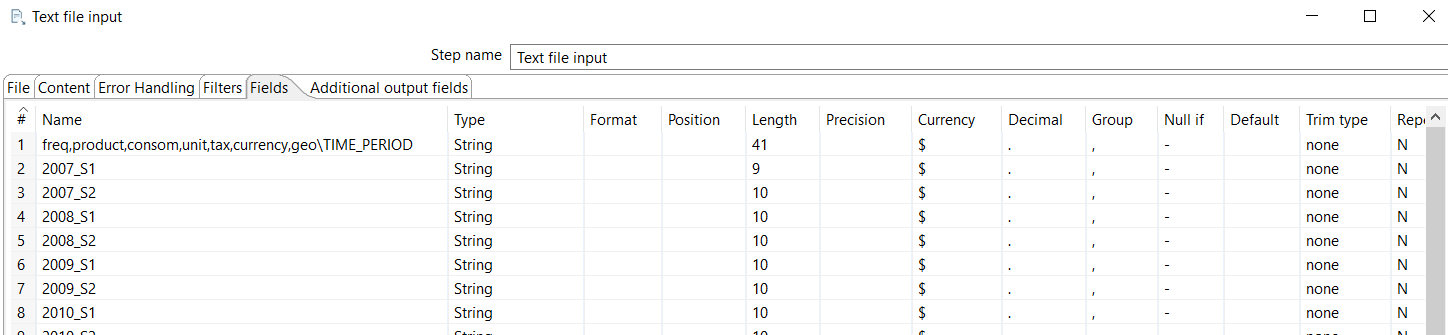
- convertir columnas en enteros

- eliminar filas con pgas igual a nulo

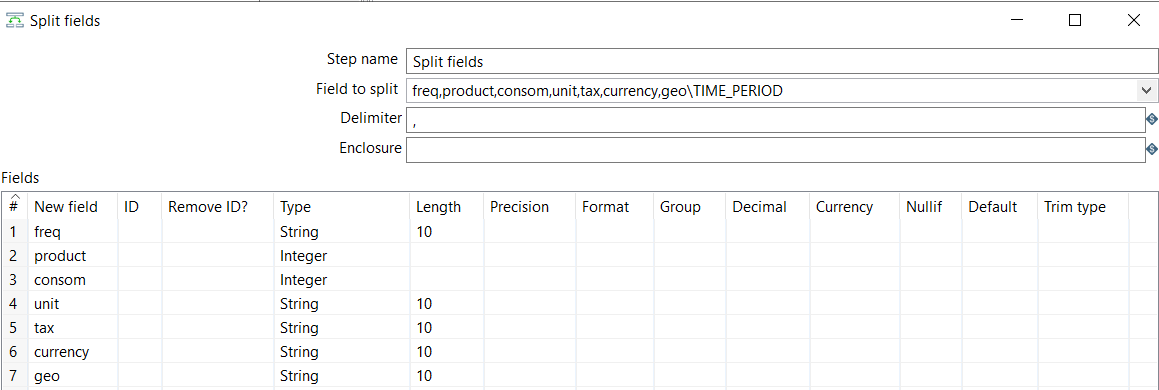
- carga a la tabla intermedia «STG\_PGAS»

A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos:

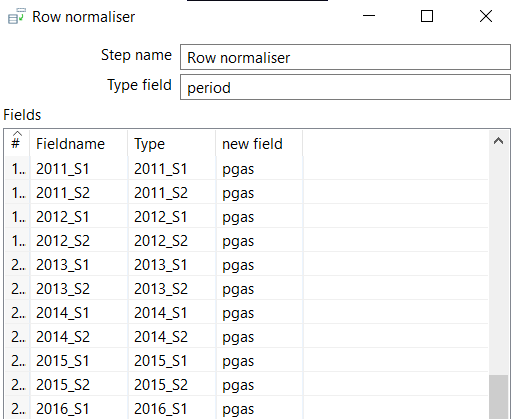
De manera similar al proceso IN\_HICP, empezamos leyendo el fichero .tsv mediante *Text file input*. Definimos como delimitador el tabulador igual que anteriormente y obtenemos en una columna las variables separadas por comas y una columna por cada periodo de tiempo.



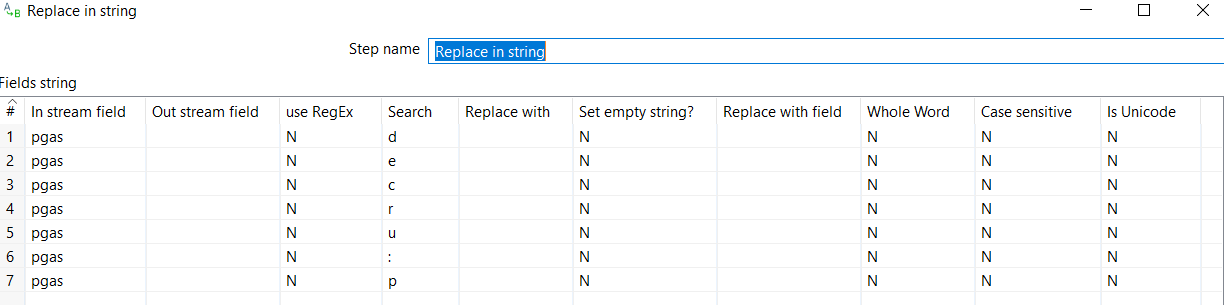
A continuación, pasamos a dividir la columna “combinada” en distintas columnas ajustandonos ya al formato esperado por la tabla STG\_PGAS. Dado que *product* y *consom* no requieren de más procesamiento las podemos declarar ya cómo Integers.

****

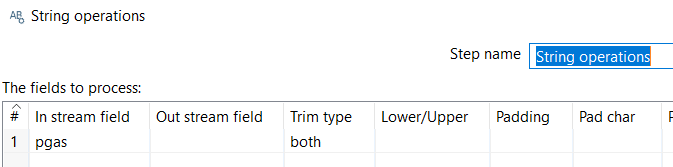
Seguidamente, convertimos las columnas de tiempo en atributos de filas asignándoles una nueva columna *period* para su valor. Y el valor que guardaba la columna de tiempo se convierte en una nueva columna pgas.

****

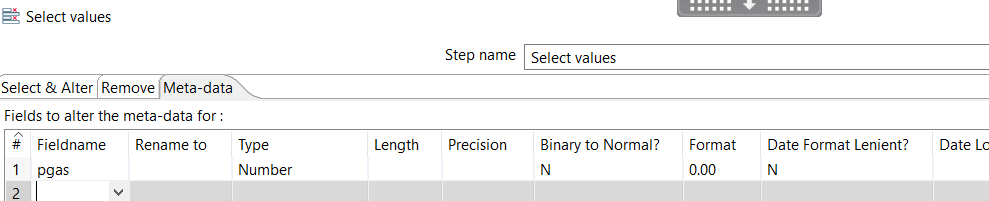
El siguiente paso es utilizar el elemento *Replace in string*, para quitar los carácteres indeseados de la variable pgas. En este caso son d,e,c,r,u,p y los dos puntos (:).

****

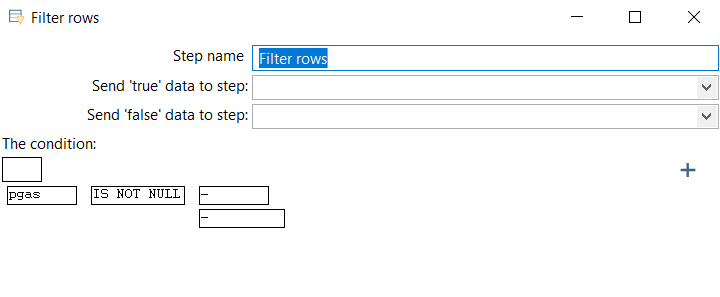
Utilizamos el elemento *String operations* para eliminar todos los espacios en blanco que aún pueda tener la variable pgas.

****

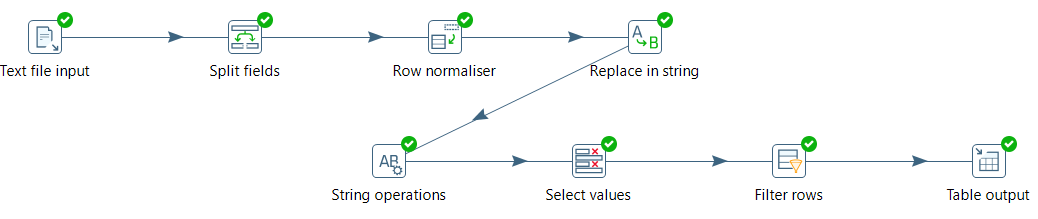
En relación a las transformaciones sólo nos queda hacer la de la variable *pgas* a numérica. En este caso, no hemos de transformar la variable de tiempo ya que contiene letras y la base de datos *Staging* espera un String.

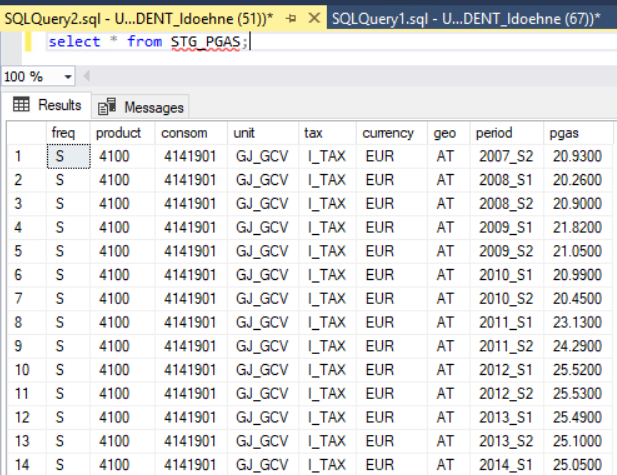


Similar al ejercicio anterior, basta con mirar que pgas sea diferente de nulo ya que las otras variables siempre están en el dataset.



Finalmente, creando la conexión con la base de datos *STG\_PGAS*, creando un *Table output* y ejecutando la transformación obtenemos la carga y transformación deseada.





**Transformación de IN\_PELEC**

A continuación, se describe parte del desarrollo de la transformación de **«IN\_PELEC»** mediante Spoon. El objetivo es cargar uno de los orígenes de los datos identificados, **nrg\_pc\_204\_tabular.tsv**, en la tabla **«STG\_PELEC»** del área intermedia (staging area).

La transformación de «IN\_PELEC» contiene las siguientes etapas:

- lectura del fichero .tsv (separado por comas y tabuladores)

- separar columnas correctamente

- convertir columna de tiempo en filas

- reemplazar caracteres de los strings

- eliminar espacios en blanco

- convertir columnas en enteros

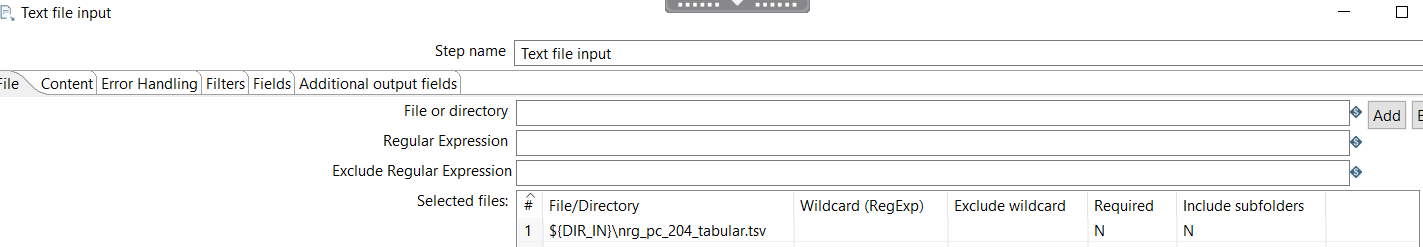
- eliminar filas con pelec igual a nulo

- carga a la tabla intermedia «STG\_PELEC»

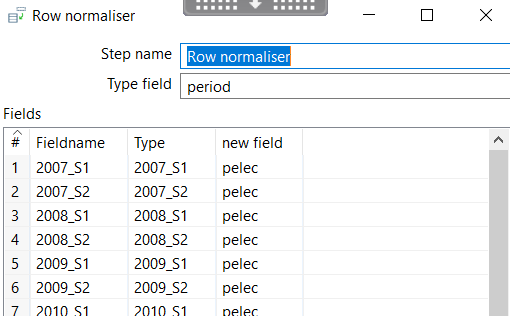
A continuación, se detallan las diferentes etapas que se implementarán para poder realizar la carga de datos:

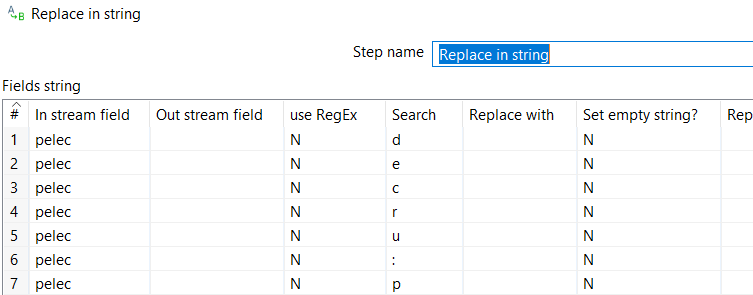
Dado que el proceso es exactamente igual que en PGAS, he copiado la transformación anterior y la he pegado en una nueva. Sólo mostraré los cambios realizados, todo lo demás es exactamente igual.

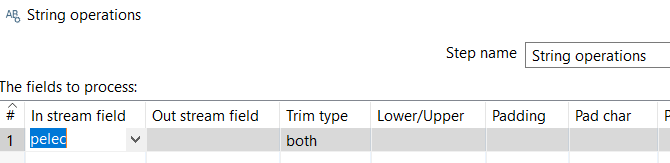
En el elemento *Text file input*, hemos de cambiar el fichero al que apuntamos. En este caso, a nrg\_pc\_204\_tabular.tsv.



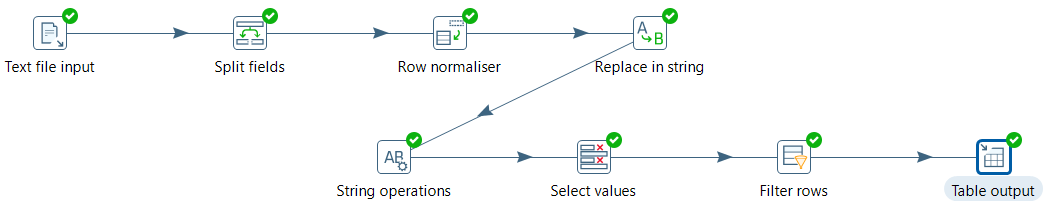
En el elemento *Row normaliser*,*replace in string, String operations*, *Select values* y *Filter rows* hemos de cambiar el nombre de la variable destino de *pgas* a *pelec*. Aquí unos ejemplos:

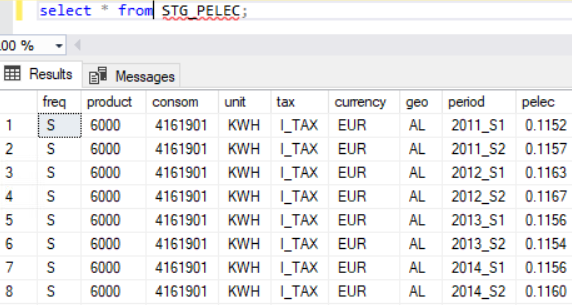






Finalmente, ajustando la conexión a la base de datos *STG\_PELEC*, creando la *Table output* y ejecutando la transformación obtenemos:



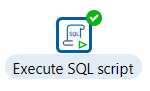


## Transformaciones Bloque TR\_DIM

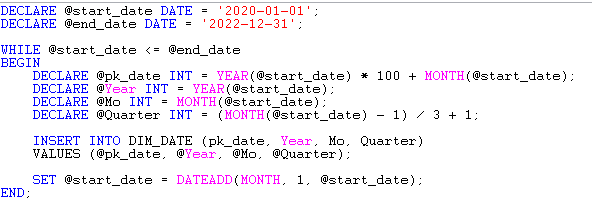
**Transformación de TR\_DIM\_DATE**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_DATE» mediante Spoon. El objetivo es generar los datos con una sentencia SQL, en la tabla “DIM\_DATE”.

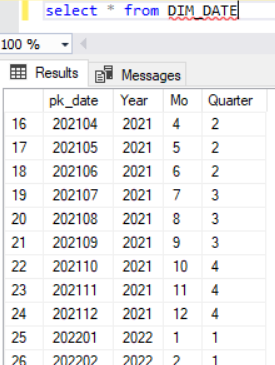
Cómo generamos los datos por una query SQL es lo mismo realizar esta acción directamente en el Microsoft SQL Server o en Spoon. En caso de coger Spoon el elemento a elegir sería *Execute SQL script.*



Con la siguiente query:



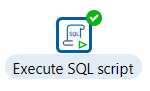
Esta query genera cada fila desde 2020-01 hasta 2022-12.



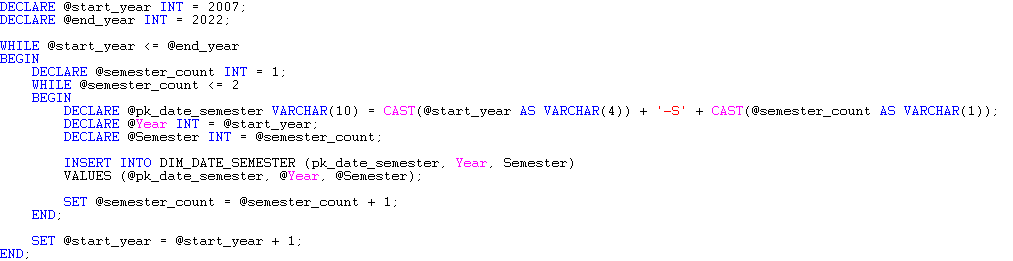
**Transformación de TR\_DIM\_DATE\_SEMESTER**

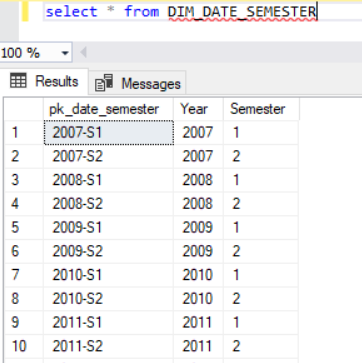
A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_DATE\_SEMESTER» mediante Spoon. El objetivo es generar los datos con una sentencia SQL, en la tabla “DIM\_DATE\_SEMESTER”.

Cómo generamos los datos por una query SQL es lo mismo realizar esta acción directamente en el Microsoft SQL Server o en Spoon. En caso de coger Spoon el elemento a elegir sería *Execute SQL script.*



Con la siguiente query:



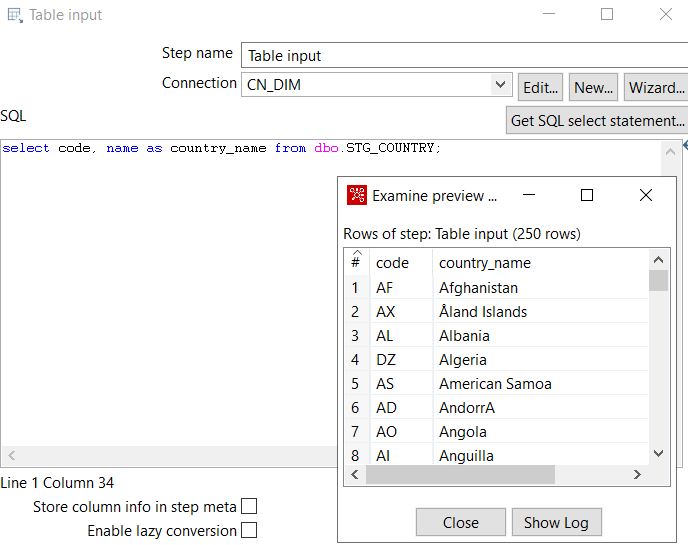


Nota: en la solución de la práctica 1 Semester está puesto cómo Integer. En el script proporcionado pero se genera la columna Semester como nvarchar(5).

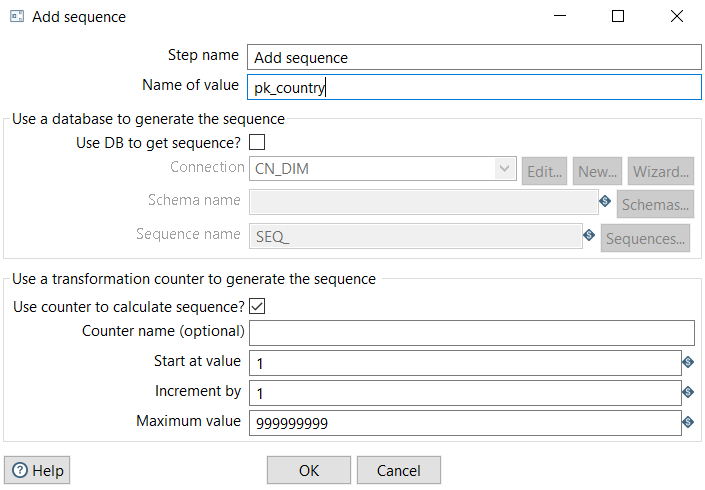
**Transformación de TR\_DIM\_COUNTRY**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_COUNTRY» mediante Spoon.

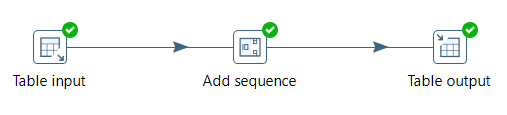
Para empezar seleccionamos los códigos y nombres de los países con el elemento *Table input*.

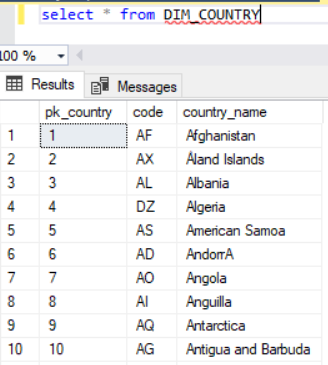


Generamos la clave privada empezando en 1 e incrementado en 1 por cada registro nuevo:



Finalmente, creamos la conexión nueva con la base de datos *DIM\_COUNTRY*, creamos un elemento *Table Output*  y ejecutamos la transformación:

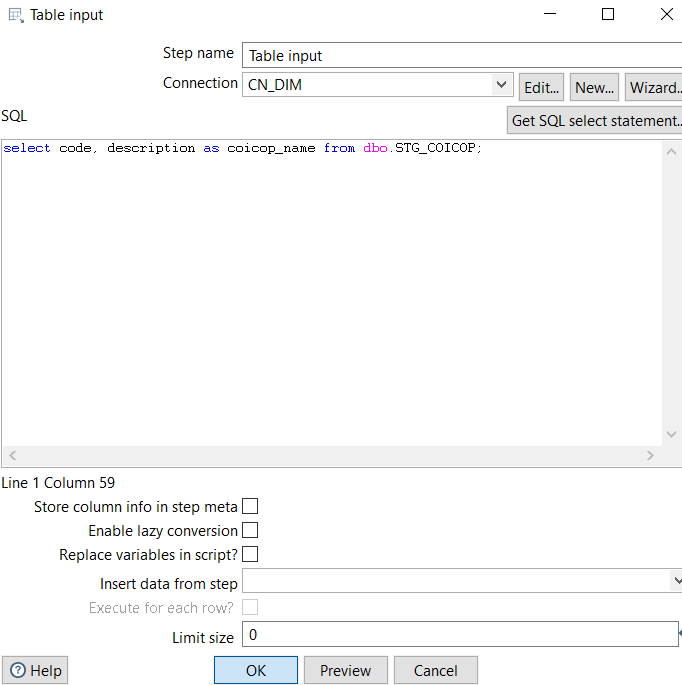




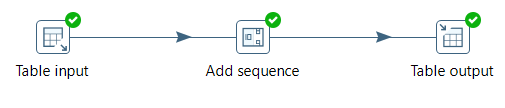
**Transformación de TR\_DIM\_COICOP**

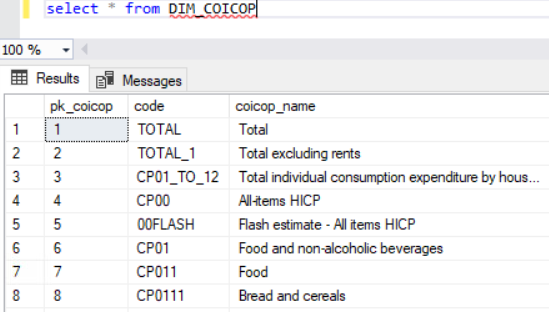
A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_COICOP» mediante Spoon.

De manera similar, empezamos cargando las columnas que nos interesan de la base de datos *STG\_COICOP*. Estos son el código y la descripción. Cambiamos el nombre de la descripción a *coicop\_name* que es como lo espera la tabla *DIM\_COICOP*.



Generamos el elemento *Add sequence* igual que hemos hecho antes (empezando en 1 e incrementado 1 cada vez). Y finalmente, terminamos creando la conexión, el elemento *Table output* y ejecutando la transformación:

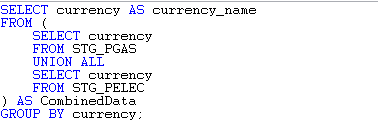




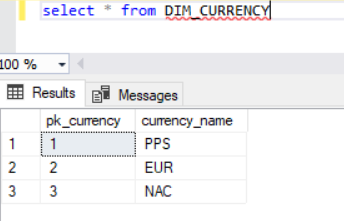
**Transformación de TR\_DIM\_CURRENCY**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_CURRENCY» mediante Spoon.

La única diferencia con la transformación anterior radica en la query sql que hacemos en el *Table Input* para seleccionar los atributos:



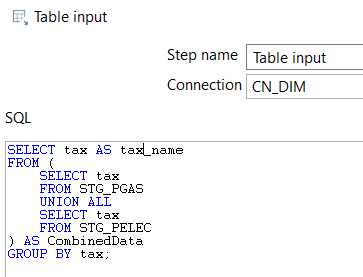
El resultado lo podemos observar a continuación:

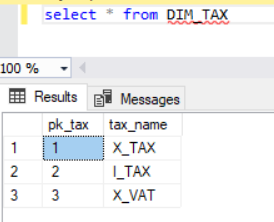


**Transformación de TR\_DIM\_TAX**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_TAX» mediante Spoon.

De forma similar, se presenta la nueva query requerida para buscar los datos de forma adecuado y el resultado de ejecutar la transformación:

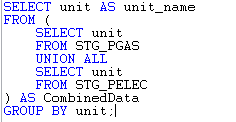


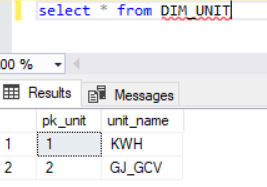


**Transformación de TR\_DIM\_UNIT**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_UNIT» mediante Spoon.

Nuevamente se genera una nueva query para seleccionar las unidad de *PGAS* y *PELEC* y se ejecuta la transformación:

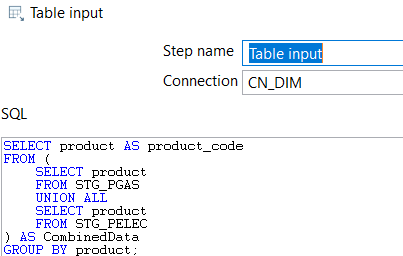


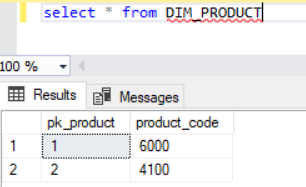


**Transformación de TR\_DIM\_PRODUCT**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_PRODUCT» mediante Spoon.

Nuevamente se genera una nueva query para seleccionar el producto de *PGAS* y *PELEC* y se ejecuta la transformación:

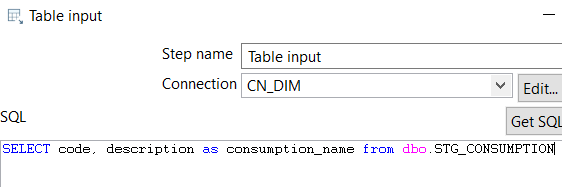


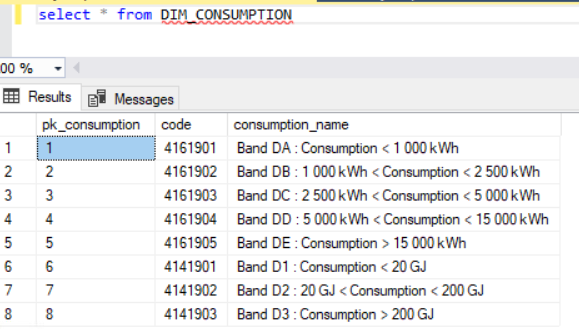


**Transformación de TR\_DIM\_CONSUMPTION**

A continuación, se describe parte del desarrollo de la transformación de «TR\_DIM\_CONSUMPTION» mediante Spoon.

Nuevamente se genera una nueva query para seleccionar el producto de *PGAS* y *PELEC* y se ejecuta la transformación:



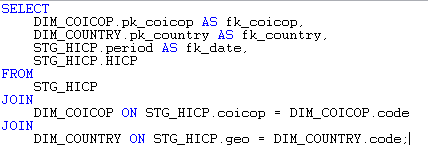


## Transformaciones Bloque TR\_FACT

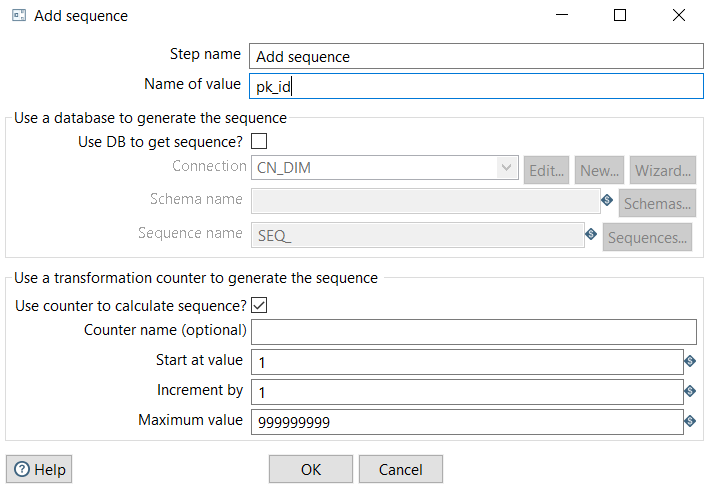
**Transformación de TR\_FACT\_EUROZONE\_INDICATORS**

A continuación, se describe parte del desarrollo de la transformación de «TR\_FACT\_EUROZONE\_INDICATORS» mediante Spoon.

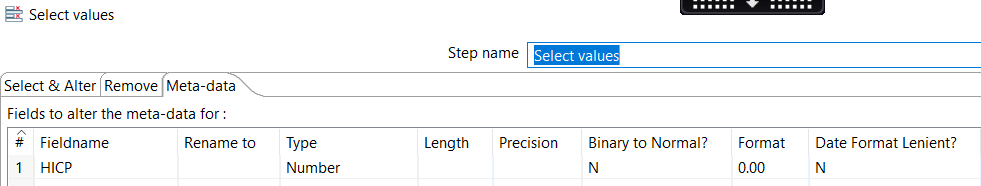
Comenzamos nuevamente con un elemento *Table Input* desde Spoon. Nuestro objetivo es capturar todos los datos de la tabla *STG\_HICP*, sustituyendo las variables coicop y geo por las claves privadas de las filas de las tablas *DIM\_COICOP* y *DIM\_COUNTRY* a las que pertenecen. La variable period y HICP las podemos capturar sin buscar en otras tablas pero hemos de cambiar el nombre de period a *fk\_date*. Todo esto lo podemos hacer con la siguiente consulta:

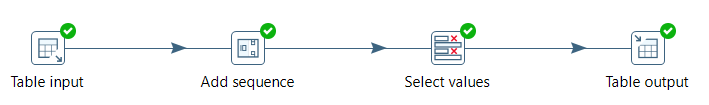


Seguidamente, utilizamos el elemento *Add sequence* para añadir la columna que nos falta *pk\_id*. Queremos que este comience en 1 y si incrementa por 1 para cada nuevo registro.

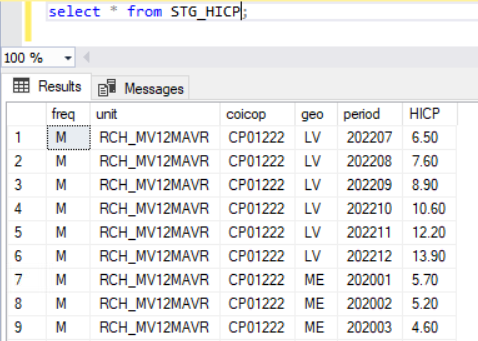
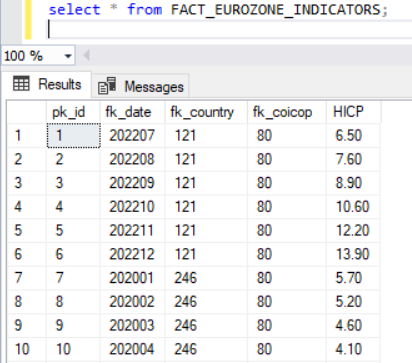


La variable *HICP* ha de ser numérica en la tabla destino, por lo que debemos asegurarnos de volver a transformarla a *Number* con el elemento *Select values*. Al hacer la consulta SQL, lo identifica como *String* por lo que hacemos la siguiente conversión:

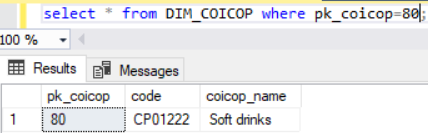
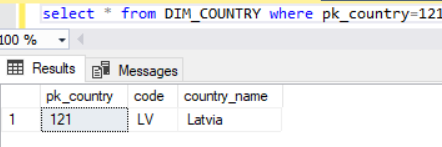


Finalmente, repetimos los pasos finales de usar la conexión a nuestra base de datos para el elemento *Table output* y ejecutamos la transformación:  


En las siguientes cartas podemos ver el resultado. A la izquierda tenemos la tabla *FACT\_EUROZONE\_INDICATORS* y a la derecha la tabla *STG\_HICP*. Podemos observar como correctamente hemos capturado los valores de coicop y geo (sustituyendolos por la clave privada de sus Dimensiones) y hemos mantenido los valores para period (ahora *fk\_date*) y HICP.



Comprobación de los valores *fk\_country* y *fk\_coicop*:

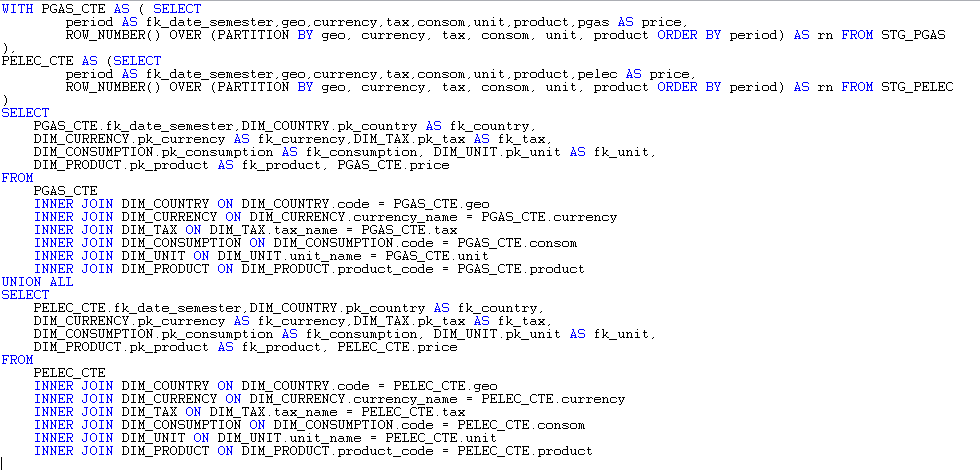


Tamaño total de *FACT\_EUROZONE\_INDICATORS* son 491014 registros. Que coincido con el tamaño de la *STG\_HICP*.

**Transformación de TR\_FACT\_ENERGY\_PRICE**

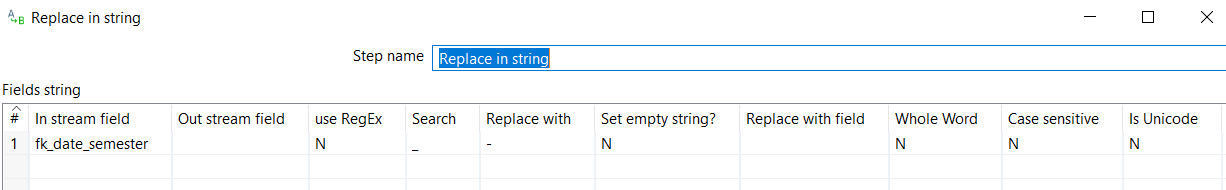
A continuación, se describe parte del desarrollo de la transformación de «TR\_FACT\_ENERGY\_PRICE» mediante Spoon.

Para la transformación comenzamos con el elemento *Table input*. La query en este caso es más compleja porque accedemos a dos tablas *STG\_PGAS* y *STG\_PELEC*. Mi manera de acceder a los datos ha sido para cada una de estás tablas hacer un inner join con cada Dimensión que necesitamos. Finalmente, unimos ambas tablas. Aquí la consulta en SQL:

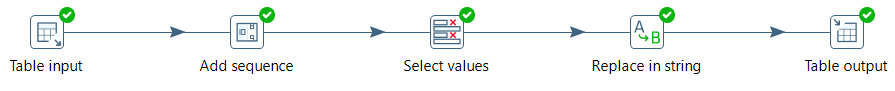


He mantenido los elementos *Add sequence* añadiendo un *pk\_id* inicializado a 1 y *Select values* donde estoy transformando el valor *price* a *Number*.

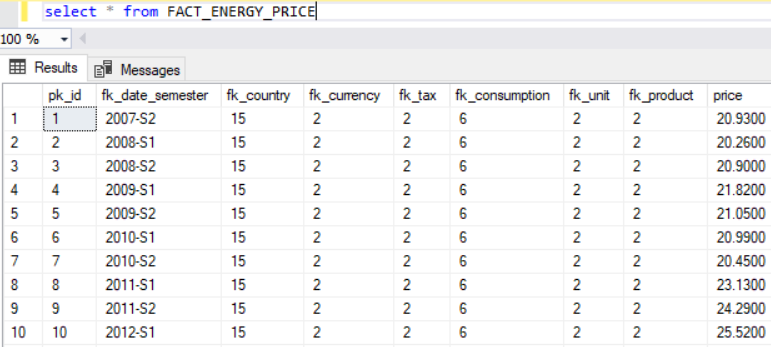
Para *FACT\_ENERGY\_PRICE* hemos de cambiar la barra baja (\_) por un guión (-). ya que la Dimension *DIM\_DATE\_SEMESTER* tiene las claves privadas con guión, sin embargo Spoon al hacer la lectura de la variable *period* en el *Staging* area ha interpretado el guión como una barra baja. Por ello, lo convertimos nuevamente en guión.



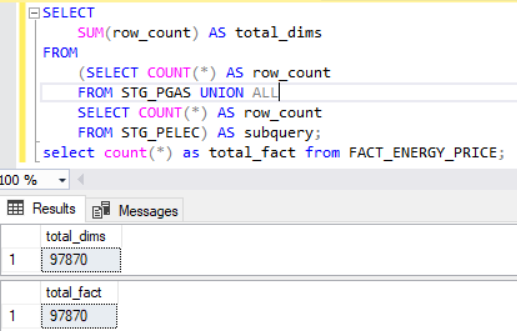
Finalizamos el proceso, conectando el elemento *Table output* a la base de datos y ejecutando la transformación.



A continuación se muestra la captura de pantalla del resultado:



Hay diversas maneras de asegurarnos de que se ha realizado correctamente la carga y la transformación. Por un lado, he comprobado que realmente las *foreign keys* coinciden con el elemento que queríamos en esa columna. También, he mirado las tablas *STG\_PGAS* y *STG\_PELEC* cogiendo registros al azar y comprobando que efectivamente son consistentes en la tabla *FACT\_ENERGY\_PRICE*. Finalmente, podemos mirar que el tamaño de *FACT\_ENERGY\_PRICE* es igual a la suma de los tamaños de *STG\_PGAS* y *STG\_PELEC*.



# IMPLEMENTACIÓN TRABAJOS (JOBS) DE LOS PROCESOS DE ETL

Los bloques de procesos ETL implementados que hay que tener en cuenta son los siguientes:

● Bloque IN\_: procesos ETL de transformación y carga al área intermedia.

● Bloque TR\_DIM: procesos ETL de transformación y carga de dimensiones.

● Bloque TR\_FACT: procesos ETL de transformación y carga de hechos.

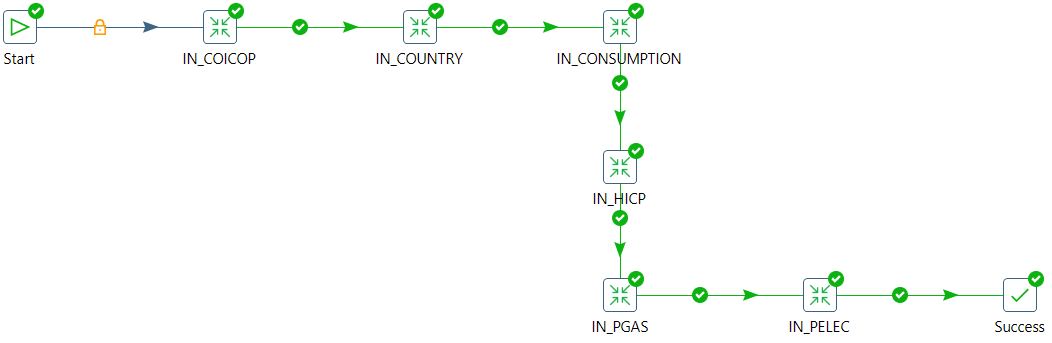
● Proceso completo: utiliza los jobs IN (bloque IN\_) y TR (bloque TR\_DIM y TR\_FACT).

En este punto, para realizar la carga efectiva de los datos, se debe diseñar mediante PDI los trabajos (jobs) que permitan la ejecución secuencial de todos los procesos ETL incluidos en cada bloque. En este apartado se deben incluir también las volumetrías obtenidas (número de registros cargados en cada tabla).

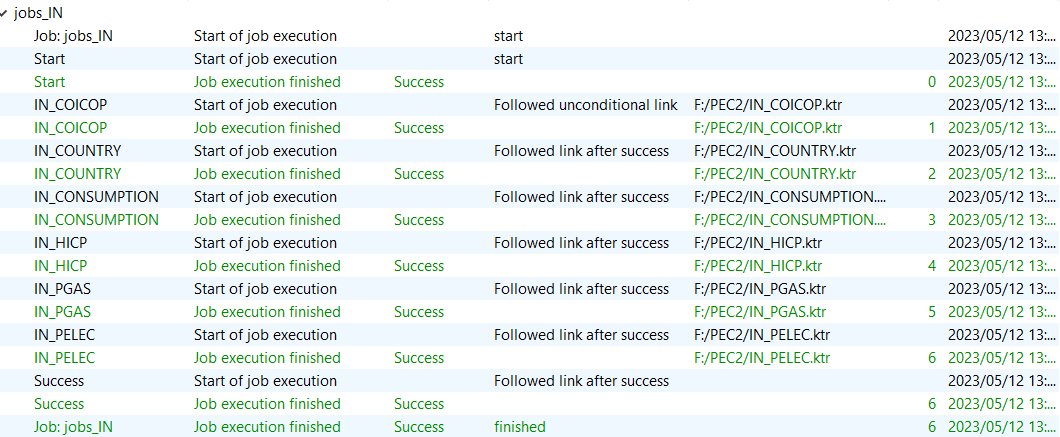
## Jobs Bloque IN

El trabajo (job) «JOB\_IN» procesa todas las transformaciones del bloque «IN\_» para la carga de datos desde las fuentes de datos proporcionadas al área intermedia (staging area).

El diseño completo del trabajo (job) «JOB\_IN» es el siguiente:



El resultado de la ejecución de la transformación completa es el siguiente:



Se observa el procesamiento con éxito de todos los pasos del «JOB\_IN» correspondientes a la ejecución de todas las transformaciones que están incluidas en el trabajo.

Volumetría:

2023/05/12 14:55:10 - Spoon - Starting job...

2023/05/12 14:55:10 - jobs\_IN - Start of job execution

2023/05/12 14:55:10 - jobs\_IN - Starting entry [IN\_COICOP]

2023/05/12 14:55:10 - Table output.0 - Finished processing (I=0, O=951, R=951, W=**951**, U=0, E=0)

2023/05/12 14:55:10 - jobs\_IN - Starting entry [IN\_COUNTRY]

2023/05/12 14:55:10 - Table output.0 - Finished processing (I=0, O=250, R=250, W=**250**, U=0, E=0)

2023/05/12 14:55:10 - jobs\_IN - Starting entry [IN\_CONSUMPTION]

2023/05/12 14:55:11 - Table output.0 - Finished processing (I=0, O=8, R=8, W=**8**, U=0, E=0)

2023/05/12 14:55:11 - jobs\_IN - Starting entry [IN\_HICP]

2023/05/12 14:55:20 - Table output.0 - Finished processing (I=0, O=491014, R=491014, W=**491014**, U=0, E=0)

2023/05/12 14:55:20 - jobs\_IN - Starting entry [IN\_PGAS]

2023/05/12 14:55:21 - Table output.0 - Finished processing (I=0, O=47980, R=47980, W=**47980**, U=0, E=0)

2023/05/12 14:55:21 - jobs\_IN - Starting entry [IN\_PELEC]

2023/05/12 14:55:22 - Table output.0 - Finished processing (I=0, O=49890, R=49890, W=**49890**, U=0, E=0)

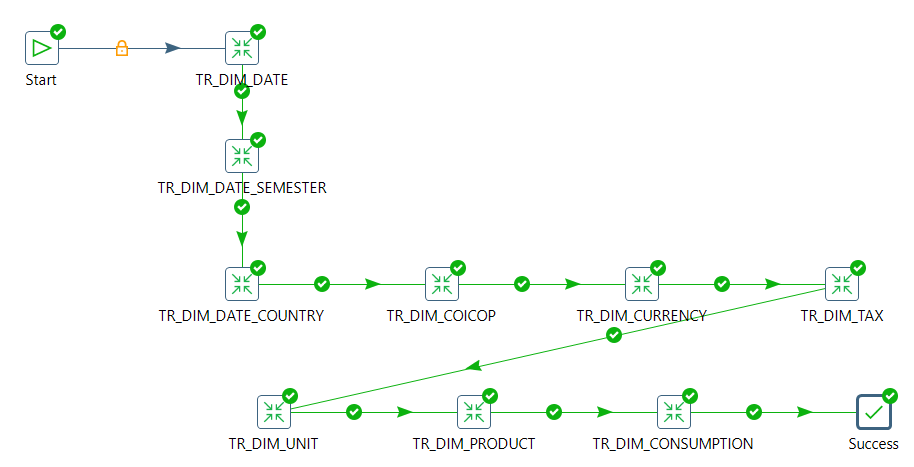
2023/05/12 14:55:22 - Spoon - Job has ended.

STG\_COICOP:951 STG\_COUNTRY: 250 STG\_CONSUMPTION: 8 SEG\_HICP:491014 STG\_PGAS:47980 STG\_PELEC:4980

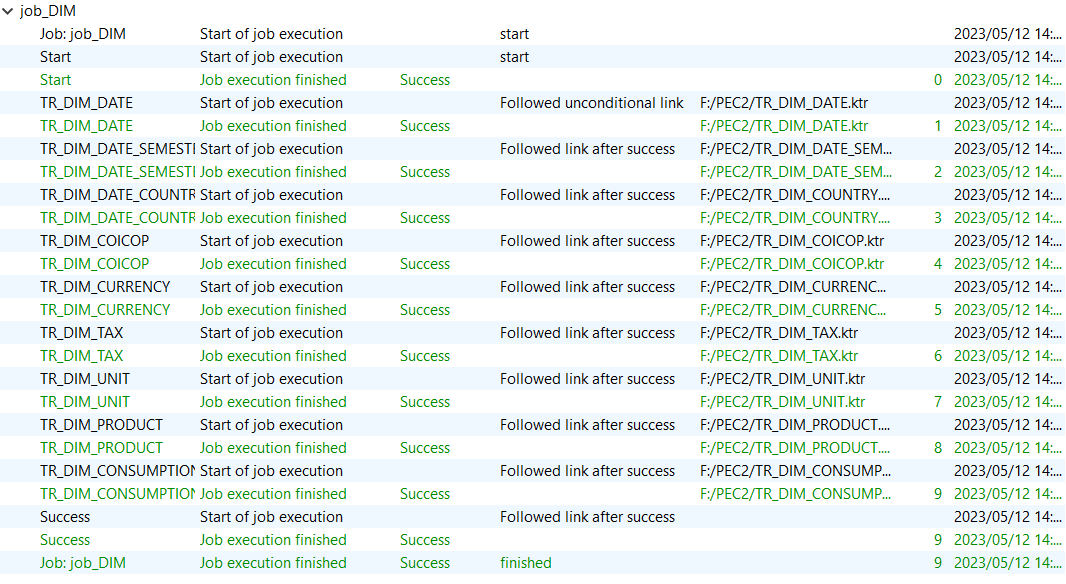
## Jobs Bloque TR\_DIM

El trabajo (job) «JOB\_TR\_DIMS» procesa todas las transformaciones del bloque «TR\_DIMS» para la carga de datos, desde las tablas intermedias hasta las tablas de dimensiones del almacén.

El diseño completo del trabajo (job) «JOB\_TR\_DIMS» es el siguiente:



El resultado de la ejecución de la transformación completa es el siguiente:



Se observa el procesamiento con éxito de todos los pasos del «JOB\_TR\_DIMS», correspondientes a la ejecución de todas las transformaciones que están incluidas en el trabajo.

Volumetría:

2023/05/12 14:59:03 - Spoon - Starting job...

2023/05/12 14:59:03 - job\_DIM - Start of job execution

2023/05/12 14:59:03 - job\_DIM - Starting entry [TR\_DIM\_DATE]

2023/05/12 14:59:09 - Execute SQL script.0 - Finished processing (I=0, O=0, R=0, W=**36**, U=0, E=0)

2023/05/12 14:59:09 - job\_DIM - Starting entry [TR\_DIM\_DATE\_SEMESTER]

2023/05/12 14:59:09 - Execute SQL script.0 - Finished processing (I=0, O=0, R=0, W=**31**, U=0, E=0)

2023/05/12 14:59:09 - job\_DIM - Starting entry [TR\_DIM\_COUNTRY]

2023/05/12 14:59:09 - Table output.0 - Finished processing (I=0, O=250, R=250, W=**250**, U=0, E=0)

2023/05/12 14:59:09 - job\_DIM - Starting entry [TR\_DIM\_COICOP]

2023/05/12 14:59:09 - Table output.0 - Finished processing (I=0, O=951, R=951, W=**951**, U=0, E=0)

2023/05/12 14:59:09 - job\_DIM - Starting entry [TR\_DIM\_CURRENCY]

2023/05/12 14:59:09 - Table output.0 - Finished processing (I=0, O=3, R=3, W=**3**, U=0, E=0)

2023/05/12 14:59:09 - job\_DIM - Starting entry [TR\_DIM\_TAX]

2023/05/12 14:59:10 - Table output.0 - Finished processing (I=0, O=3, R=3, W=**3**, U=0, E=0)

2023/05/12 14:59:10 - job\_DIM - Starting entry [TR\_DIM\_UNIT]

2023/05/12 14:59:10 - Table output.0 - Finished processing (I=0, O=2, R=2, W=**2**, U=0, E=0)

2023/05/12 14:59:10 - job\_DIM - Starting entry [TR\_DIM\_PRODUCT]

2023/05/12 14:59:10 - Table output.0 - Finished processing (I=0, O=2, R=2, W=**2**, U=0, E=0)

2023/05/12 14:59:10 - job\_DIM - Starting entry [TR\_DIM\_CONSUMPTION]

2023/05/12 14:59:10 - Table output.0 - Finished processing (I=0, O=8, R=8, W=**8**, U=0, E=0)

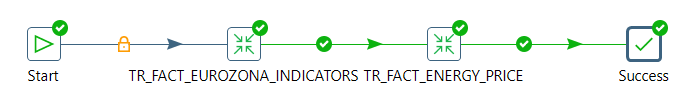
2023/05/12 14:59:10 - Spoon - Job has ended.

DIM\_DATE:36 DIM\_DATE\_SEMESTER:31 DIM\_COUNTRY:250 DIM\_COICOP:951 DIM\_CURRENCY:3 DIM\_TAX:3 DIM\_UNIT:2 DIM\_PRODUCT:2 DIM\_CONSUMPTION:8

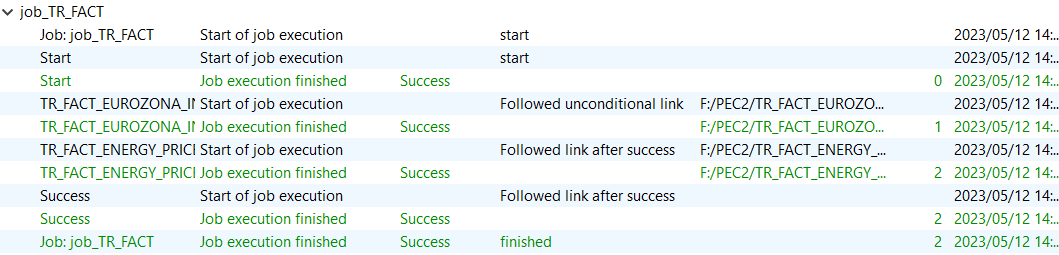
## Jobs Bloque TR\_FACT

El trabajo (job) «JOB\_TR\_FACTS» procesa todas las transformaciones del bloque «TR\_FACT» para la carga de datos desde las tablas intermedias a las tablas de hechos del almacén.

El diseño completo del trabajo (job) «JOB\_TR\_FACTS» es el siguiente:



El resultado de la ejecución de la transformación completa es el siguiente:



Se observa el procesamiento con éxito de todos los pasos del «JOB\_TR\_FACTS», correspondientes a la ejecución de todas las transformaciones que están incluidas en el trabajo.

Volumetría:

2023/05/12 15:07:36 - Spoon - Starting job...

2023/05/12 15:07:36 - job\_TR\_FACT - Start of job execution

2023/05/12 15:07:36 - job\_TR\_FACT - Starting entry [TR\_FACT\_EUROZONA\_INDICATORS]

2023/05/12 15:07:56 - Table output.0 - Finished processing (I=0, O=491014, R=**491014**, W=491014, U=0, E=0)

2023/05/12 15:07:56 - job\_TR\_FACT - Starting entry [TR\_FACT\_ENERGY\_PRICE]

2023/05/12 15:07:57 - Replace in string.0 - Finished processing (I=0, O=0, R=97870, W=**97870**, U=0, E=0)

2023/05/12 15:07:57 - Spoon - Job has ended.

FACT\_EUROZONA\_INDICATORS:491014

FACT\_ENERGY\_PRICE:97870

## Proceso completo (utiliza los jobs IN y TR)

El trabajo (job) «JOB\_CARGA\_DW» orquesta todos los trabajos anteriores en un único proceso.

El diseño completo del trabajo (job) «JOB\_CARGA\_DW» es el siguiente:



Los pasos incluidos en el trabajo «JOB\_CARGA\_DW» son:

* • Inicio del job.
* • Ejecución orquestada de los jobs de carga de todas las transformaciones («JOB\_IN», «JOB\_DIM», «JOB\_TR\_FACTS»).
* • Finalización del job.

El resultado de la ejecución de la transformación completa es el siguiente:



Se observa el procesamiento con éxito de todos los pasos del «JOB\_CARGA\_DW», correspondientes a la ejecución de todas las transformaciones que están incluidas en el trabajo.

Volumetría:

2023/05/12 15:10:04 - Spoon - Starting job...

2023/05/12 15:10:04 - Job\_CARGA\_DW - Start of job execution

2023/05/12 15:10:04 - jobs\_IN - Starting entry [IN\_COICOP]

2023/05/12 15:10:05 - Table output.0 - Finished processing (I=0, O=951, R=951, W=**951**, U=0, E=0)

2023/05/12 15:10:05 - jobs\_IN - Starting entry [IN\_COUNTRY]

2023/05/12 15:10:05 - Table output.0 - Finished processing (I=0, O=250, R=250, W=**250**, U=0, E=0)

2023/05/12 15:10:05 - jobs\_IN - Starting entry [IN\_CONSUMPTION]

2023/05/12 15:10:05 - Table output.0 - Finished processing (I=0, O=8, R=8, W=**8**, U=0, E=0)

2023/05/12 15:10:05 - jobs\_IN - Starting entry [IN\_HICP]

2023/05/12 15:10:14 - Table output.0 - Finished processing (I=0, O=491014, R=491014, W=**491014**, U=0, E=0)

2023/05/12 15:10:14 - jobs\_IN - Starting entry [IN\_PGAS]

2023/05/12 15:10:16 - Table output.0 - Finished processing (I=0, O=47980, R=47980, W=**47980**, U=0, E=0)

2023/05/12 15:10:16 - jobs\_IN - Starting entry [IN\_PELEC]

2023/05/12 15:10:17 - Table output.0 - Finished processing (I=0, O=49890, R=49890, W=**49890**, U=0, E=0)

2023/05/12 15:10:17 - job\_DIM - Starting entry [TR\_DIM\_DATE]

2023/05/12 15:10:23 - Execute SQL script.0 - Finished processing (I=0, O=0, R=0, W=**36**, U=0, E=0)

2023/05/12 15:10:23 - job\_DIM - Starting entry [TR\_DIM\_DATE\_SEMESTER]

2023/05/12 15:10:23 - Execute SQL script.0 - Finished processing (I=0, O=0, R=0, W=**31**, U=0, E=0)

2023/05/12 15:10:23 - job\_DIM - Starting entry [TR\_DIM\_COUNTRY]

2023/05/12 15:10:23 - Table output.0 - Finished processing (I=0, O=250, R=250, W=**250**, U=0, E=0)

2023/05/12 15:10:23 - job\_DIM - Starting entry [TR\_DIM\_COICOP]

2023/05/12 15:10:23 - Table output.0 - Finished processing (I=0, O=951, R=951, W=**951**, U=0, E=0)

2023/05/12 15:10:23 - job\_DIM - Starting entry [TR\_DIM\_CURRENCY]

2023/05/12 15:10:24 - Table output.0 - Finished processing (I=0, O=3, R=3, W=**3**, U=0, E=0)

2023/05/12 15:10:24 - job\_DIM - Starting entry [TR\_DIM\_TAX]

2023/05/12 15:10:24 - Table output.0 - Finished processing (I=0, O=3, R=3, W=**3**, U=0, E=0)

2023/05/12 15:10:24 - job\_DIM - Starting entry [TR\_DIM\_UNIT]

2023/05/12 15:10:24 - Table output.0 - Finished processing (I=0, O=2, R=2, W=**2**, U=0, E=0)

2023/05/12 15:10:24 - job\_DIM - Starting entry [TR\_DIM\_PRODUCT]

2023/05/12 15:10:24 - Table output.0 - Finished processing (I=0, O=2, R=2, W=**2**, U=0, E=0)

2023/05/12 15:10:24 - job\_DIM - Starting entry [TR\_DIM\_CONSUMPTION]

2023/05/12 15:10:25 - Table output.0 - Finished processing (I=0, O=8, R=8, W=**8**, U=0, E=0)

2023/05/12 15:10:25 - job\_TR\_FACT - Starting entry [TR\_FACT\_EUROZONA\_INDICATORS]

2023/05/12 15:10:46 - Table output.0 - Finished processing (I=0, O=491014, R=491014, W=**491014**, U=0, E=0)

2023/05/12 15:10:46 - job\_TR\_FACT - Starting entry [TR\_FACT\_ENERGY\_PRICE]

2023/05/12 15:10:47 - Replace in string.0 - Finished processing (I=0, O=0, R=97870, W=**97870**, U=0, E=0)

2023/05/12 15:10:47 - Spoon - Job has ended.

Staging:

STG\_COICOP:951 STG\_COUNTRY: 250 STG\_CONSUMPTION: 8 SEG\_HICP:491014 STG\_PGAS:47980 STG\_PELEC:4980

Dimensions:

DIM\_DATE:36 DIM\_DATE\_SEMESTER:31 DIM\_COUNTRY:250 DIM\_COICOP:951 DIM\_CURRENCY:3 DIM\_TAX:3 DIM\_UNIT:2 DIM\_PRODUCT:2 DIM\_CONSUMPTION:8

Facts:

FACT\_EUROZONA\_INDICATORS:491014

FACT\_ENERGY\_PRICE:97870

El tiempo total de la carga inicial del data warehouse es de aproximadamente un minuto.

2023/05/12 14:35:45 - Spoon - Starting job...

2023/05/12 14:36:41 - Spoon - Job has ended.