Caso práctico: Almacén de datos para el análisis del parque de vehículos

PR3 – Sistemas de bases de datos analíticas

La tercera práctica **PR3 – Sistemas de base de datos analíticas** consta de dos partes, una primera parte destinada a la evaluación del contenido teórico y una segunda parte destinada a la evaluación práctica de los conocimientos adquiridos mediante un conjunto de ejercicios a realizar en el entorno de prácticas de la asignatura.

La parte teórica tiene una ponderación del 30% sobre la nota total de la práctica. Se evalúa mediante un **cuestionario,** disponible en el apartado *Contenidos del aula*. Su objetivo es comprobar la correcta comprensión del tercer módulo de la asignatura. Debe realizarse después de haber estudiado el módulo 3 y haber leído detenidamente las indicaciones de realización del mismo.

La parte práctica también pondera el 70% de la nota total de la práctica, se lleva a cabo en la máquina virtual (VDI) mediante la realización de todos los ejercicios que se detallan en el apartado “Enunciado parte práctica.” Tened en cuenta que deberéis utilizar las especificaciones técnicas, tales como el nombre de las dos bases de datos PostgreSQL y SQL Server ya facilitadas y utilizadas en anteriores prácticas.

Se recomienda consultar la documentación disponible en el apartado *Recursos de Aprendizaje PR3 del espacio Contenidos del aula*. Dichos recursos pretenden facilitar la realización de los ejercicios y resolver posibles dudas. También se incluye la **documentación técnica**, imprescindible para trabajar en el entorno virtual (VDI), que está preparado con todas las herramientas necesarias, instaladas y configuradas. Recordad que ante cualquier duda y/o problema técnico relativo al funcionamiento del entorno o del software instalado debéis consultar al profesor del aula *76.596.- Laboratorio de Soporte a las bases de datos analíticas*. **Únicamente se evaluarán los ejercicios cuya resolución esté acompañada de comentarios; las imágenes o gráficas sin ningún tipo de explicación no serán evaluadas.**

Parte práctica

En esta tercera práctica (PR3) y dentro del contexto del caso práctico “Almacén de datos para el análisis del parque de vehículos”, se utilizarán diferentes componentes de la FIC (Factoría de Información Corporativa) para construir los distintos elementos necesarios para almacenar los datos en una base de datos analítica y poder llevar a cabo diferentes actividades de análisis.

Tras haber completado la PR1 y la PR2, y conociendo ya las distintas herramientas disponibles en la VDI, en particular las bases de datos PostgreSQL y SQL Server, así como el lenguaje SQL, es momento de afrontar un nuevo reto, que se describe a continuación.

Contexto

Conocedores de la existencia del software destinado a la gestión del parque de vehículos perteneciente al sector de la automoción, y familiarizados con el modelo de base de datos que lo sustenta, ha surgido la necesidad estratégica de construir una Factoría de Información Corporativa (FIC). Esta FIC tiene como objetivo central satisfacer eficientemente las diversas demandas analíticas derivadas del volumen creciente de datos y la complejidad inherente al parque vehicular español.

Para alcanzar este propósito, se han identificado tres fuentes fundamentales provenientes de la Dirección General de Tráfico (DGT).

La integración de estas fuentes en una Factoría de Información Corporativa (FIC) permitirá optimizar la toma de decisiones estratégicas mediante el análisis profundo y multidimensional del parque vehicular. Además, posibilitará una visión integrada y coherente de la información, simplificando la identificación de oportunidades comerciales, planificación de recursos y desarrollo de políticas públicas adaptadas a la realidad actual y futura del sector automovilístico español.

La arquitectura de la FIC estará constituida por varios elementos alojados en diferentes sistemas e interconectados entre ellos:

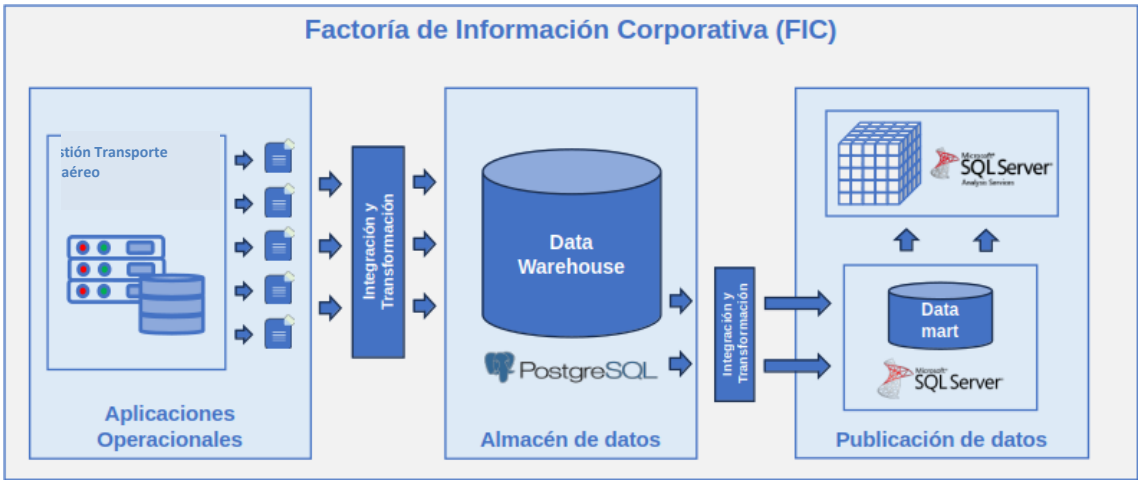
- Capa de aplicaciones operacionales, donde están alojados el servidor de aplicaciones y la base de datos destinados a dar soporte a la aplicación de Almacén de datos para el análisis del parque de vehículos. Con periodicidad diaria, existirá un proceso de extracción de datos que creará las siguientes fuentes:

    - ***dgt_type.csv*****,** que facilita una clasificación estructurada de los vehículos en dos niveles (tipo y subtipo), permitiendo categorizar con precisión los vehículos registrados según las normativas oficiales.
    - ***vehicles.csv***, un conjunto detallado con datos específicos sobre matriculaciones vehiculares proporcionando información crucial para estudios de mercado y tendencias de adquisición.
    - ***municipalities.csv***, que contiene información geográfica y demográfica oficial de los municipios españoles. Esta información contextual permite realizar análisis territoriales profundos, incluyendo distribución, densidad y patrones geográficos del parque automovilístico.

- Capa de almacén de datos, lugar donde estará ubicado el *Data Warehouse* soportado por la base de datos PostgreSQL. Los datos estarán almacenados en un mismo sistema, pero estructurados en dos capas diferenciadas:
    - Una capa de *staging* destinada a la consolidación de las diferentes fuentes de datos, y aunque esté implementada físicamente con el resto de los datos, todas sus tablas estarán identificadas mediante el prefijo *stg_nombre_de_tabla*. Por ejemplo, *stg_vehicles.*
    - Una capa que ejercerá propiamente la función de almacén de datos, donde estarán todas las tablas de hechos y dimensiones, procedentes de diferentes sistemas operacionales.
- Capa de publicación de datos, lugar donde se almacenará la información específica para ser consultada y analizada por los usuarios. Esta capa estará constituida por dos componentes:
    - Un *Datamart* destinado a almacenar los datos del parque de vehículos, siendo un área temática específica para esta tipología de gestión. Este almacén de datos departamental estará soportado por la base de datos *SQL Server.*

- Una instancia de *SQL Server Analysis Services* destinada a almacenar mediante tecnología MOLAP un cubo destinado al análisis de los datos del sector de la automoción.

Todos estos componentes están interconectados y orquestados mediante procesos de integración y transformación (ETL), destinados a trasladar y transformar los datos entre los distintos componentes de la FIC. La herramienta ETL es *Pentaho Spoon* y es la encargada de ejecutar los procesos de integración y transformación, interconectando a la vez los diferentes sistemas de la FIC.



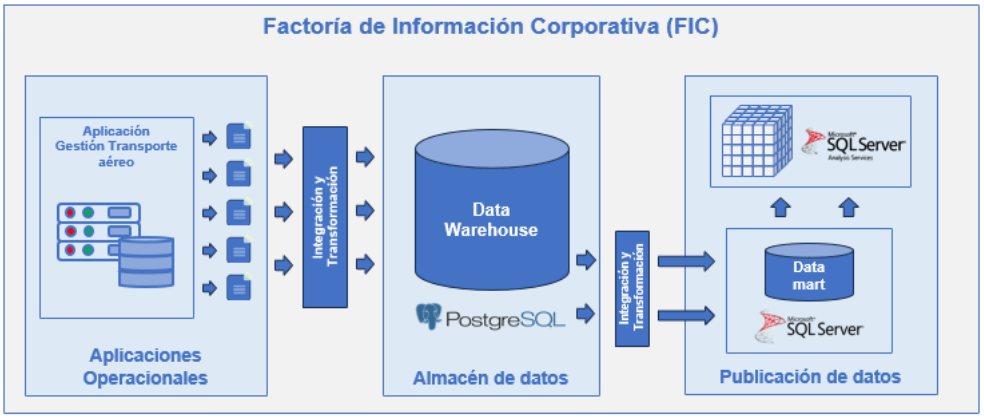
Ejercicio 1. Carga de un fichero en la base de datos (20%)

Objetivo:

Conocer la herramienta ETL **Pentaho Spoon (Pentaho Data Integration)** suministrada en el entorno de laboratorio VDI, y poder diseñar y ejecutar procesos de integración y transformación de datos.

Indicaciones para el desarrollo del ejercicio:

A continuación, se detallarán los diferentes pasos necesarios para familiarizarse con las herramientas de integración y transformación ETL disponibles en el laboratorio VDI. Para ello se llevarán a cabo un conjunto de tareas destinadas a cargar en la base de datos *PostgreSQL* los tres ficheros que diariamente genera el software de gestión del parque de vehículos y que pone a disposición de la FIC, con el objetivo de analizarlas por los usuarios.



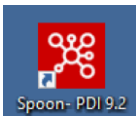
Los apartados 1.1 a 1.3 tienen carácter introductorio y sirven de guía para familiarizarse con el entorno, la herramienta Spoon y la estructura de los datos.

El apartado 1.4 marca el inicio de la actividad evaluable, por lo que el estudiantado deberá crear y ejecutar un proceso de transformación (ETL), que cargue los ficheros que diariamente genera el software de gestión del parque de vehículos y que pone a

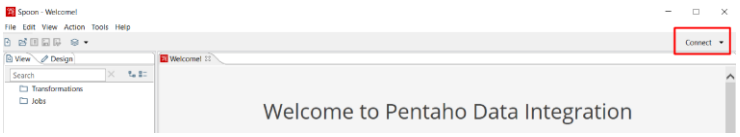
disposición de la FIC para facilitar el análisis a los usuarios. Se debe documentar con el máximo detalle las evidencias del proceso, incluyendo capturas de pantalla y explicaciones.

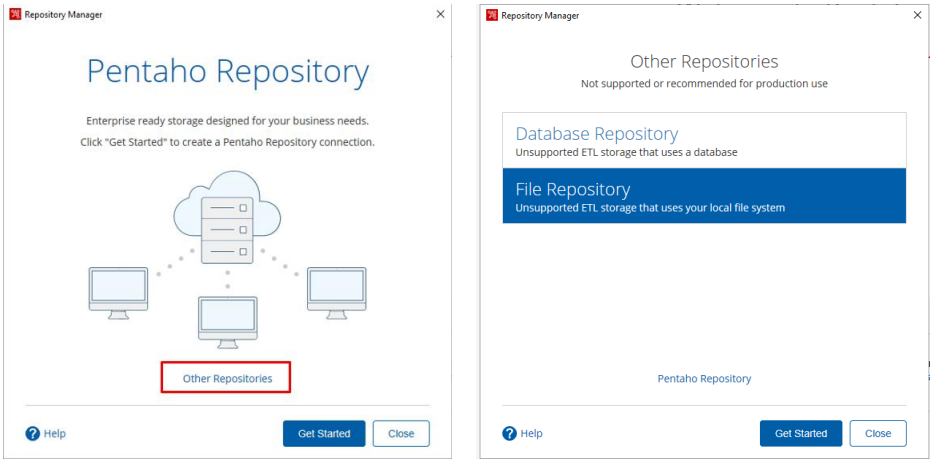
1.1. Configuración de Spoon

La primera tarea es acceder a la herramienta **Spoon** mediante el icono del escritorio de la VDI,

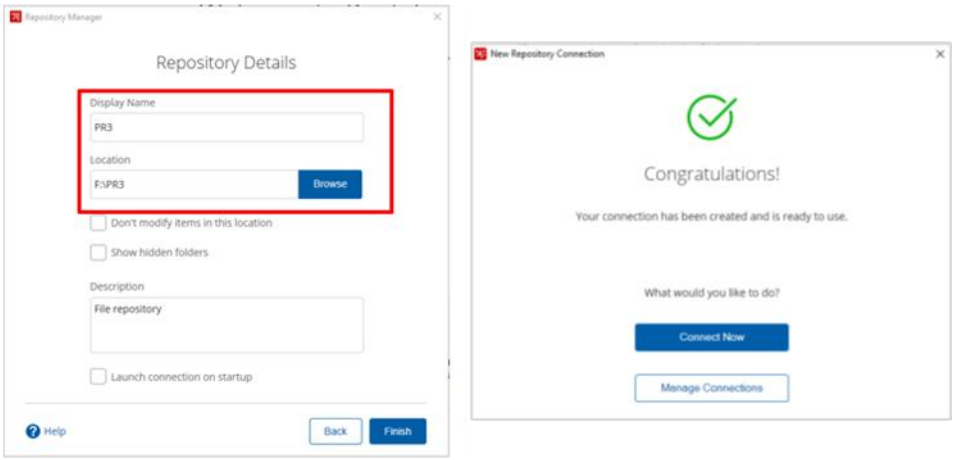


y crear un repositorio de tipo *file* con el nombre PR3, seleccionando la opción *File Repository, p*ara posteriormente indicar una ubicación física y asignar un nombre al repositorio. Es muy recomendable utilizar una ubicación persistente en la VDI.

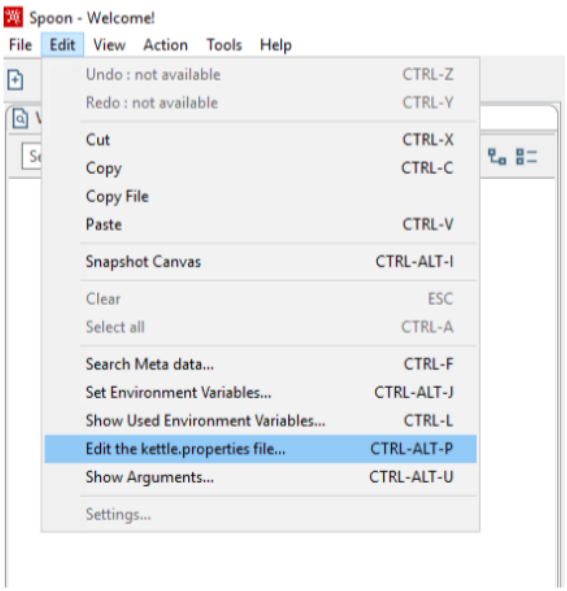




Importante: en caso de no visualizarse los botones de la primera pantalla, se debe ajustar el tamaño de las letras del escritorio al 100%.



Una buena práctica es utilizar variables de entorno para evitar introducir errores en definiciones repetitivas durante la implementación de los procesos de integración. En este sentido *Spoon* nos permite añadir variables propias en el archivo *kettle.properties*.



En este caso se utilizará la variable “DIR_IN” destinada a almacenar la ruta de las fuentes de datos. *DIR_IN = F:\Fuentes* La referencia a las variables de entorno durante la implementación de los procesos se indica mediante llaves, de esta manera: *${DIR_IN}*

Una vez copiado los ficheros *vehicles.csv, dgt_type.csv y municipalities.csv* en una de las carpetas persistentes de la VDI que indica la variable *DIR_IN*, se debe llevar a cabo el siguiente conjunto de actividades destinadas a la preparación de la base de datos y a la construcción del proceso de integración.

1.2. Análisis de las fuentes de datos

El objetivo del análisis previo es revisar las fuentes de datos proporcionadas, determinar qué tipo de información contienen, cuál es su formato y qué datos deben ser cargados y cuáles deben ser descartados.

El fichero *dgt_type.csv* contiene la clasificación oficial utilizada por la Dirección General de Tráfico (DGT) para categorizar los vehículos registrados en España. Presenta una estructura de dos niveles (tipo y subtipo de vehículo) con identificadores alfanuméricos y descripciones textuales claras, permitiendo asignar de forma precisa cada vehículo a una categoría específica según sus características**.** Contiene los datos de la clasificación de la DGT de los vehículos registrados en el parque en dos niveles: tipo y subtipo de vehículo. El fichero es de formato plano y contiene los campos separados por coma, y con la primera fila con los encabezados de cada columna, dando un total de 112 registros.

Campo	Descripción	Tipo	Ejemplo
cod_type_dgt	Tipo vehículo	Texto	“3”
des_type_dgt	Descripción del tipo vehículo	Texto	“AUTOBUSES”
cod_subtype_dgt	Subtipo vehículo	Texto	“31”
des_subtype_dgt	Descripción del subtipo vehículo	Texto	“AUTOBÚS ARTICULADO”

El fichero *municipalities.csv* es un conjunto de datos que reúne información geográfica y demográfica de los municipios de España. Incluye códigos oficiales del Instituto Nacional de Estadística (INE), nombre del municipio y provincia, población, superficie, coordenadas geográficas y altitud. Este archivo permite contextualizar geográficamente los datos vehiculares, facilitando análisis de distribución, densidad vehicular, y otros estudios territoriales. El fichero es de formato plano y contiene los campos separados por coma, y con la primera fila con los encabezados de cada columna, dando un total de 8.132 registros.

Campo	Descripción	Tipo	Ejemplo
cod_ine	Código INE del municipio (identificador único de 11 dígitos)	Texto	01001000000
cod_geo	Código geográfico del municipio (5 dígitos: provincia+municipio, según INE)	Texto	01010
cod_prov	Código de la provincia (2 dígitos)	Texto	01
provincie	Nombre de la provincia	Texto	Araba/Álava
name_muni	Nombre del municipio	Texto	Alegría-Dulantzi
population_muni	Población del municipio (número de habitantes)	Numérico	2975
surface	Superficie del término municipal (en hectáreas)	Numérico	1994,5872
perimeter	Perímetro municipal (longitud de frontera, en metros)	Numérico	35069
cod_ine_capital	Código INE de la localidad capital del municipio (11 dígitos)	Texto	01001000101
capital	Nombre de la localidad capital (ciudad/pueblo cabecera del municipio)	Texto	Alegría-Dulantzi
population_capital	Población de la localidad capital	Numérico	2860
longitude_etrs89	Longitud geográfica de la capital (coordenada ETRS89, grados decimales)	Numérico	-2,51243731
latitude_etrs89	Latitud geográfica de la capital (coordenada ETRS89, grados decimales)	Numérico	42,83981158
altitude	Altitud de la capital (metros sobre el nivel del mar)	Numérico	568

El fichero *vehicles.csv* proporciona información detallada sobre vehículos registrados, incluyendo la fecha de matriculación, municipio de matriculación (código geográfico INE), marca, modelo, condición (nuevo o usado) y subtipo según la clasificación DGT.

Campo	Descripción	Tipo	Ejemplo
id	Identificador único del registro del vehículo	Numérico	303304
municipality	Código del municipio de matriculación (5 dígitos, INE)	Texto	08156
brand	Marca del vehículo	Texto	FORD
model	Modelo del vehículo	Texto	TRANSIT CUSTOM
registration_date	Fecha de matriculación (formato AAAA-MM-DD)	Fecha	2020-03-01
new_used	Indica si el vehículo es Nuevo (N) o Usado (U)	Texto	N
subtype	Código de subtipo de vehículo según clasificación DGT	Texto	20

Nota: El campo *new_used* es un campo *boolean* simulado

Estos datos facilitan análisis relacionado con patrones de compra, distribución geográfica y tipología del parque vehicular español. El fichero es de formato plano y contiene los campos separados por coma, y con la primera fila con los encabezados de cada columna, dando un total de 1.721.704 registros.

Las tres fuentes se complementan perfectamente, permitiendo análisis integrados entre características del parque vehicular y datos geográficos y demográficos del territorio español.

1.3. Creación de las tablas en la base de datos en la capa de Staging

El primer paso para la implementación del proceso de ETL consiste en la creación de las tablas intermedias en la *staging area*.

La creación de tablas se debe llevar a cabo una única vez, mediante scripts sobre la base de datos (en nuestro caso *PostgreSQL*) y se utilizarán tablas intermedias en los procesos IN, que nos servirán para cargar en la tabla los datos procedentes de las fuentes de datos.
En base a esta información, los scripts de creación de las tablas en la base de datos *PostgreSQL* serán:

```
DROP TABLE IF EXISTS dbo.stg_dgt_type;
CREATE TABLE dbo.stg_dgt_type (
        cod_type_dgt varchar(1) NOT NULL,
        des_type_dgt varchar(64) ,
        cod_subtype_dgt varchar(2) NOT NULL,
        des_subtype_dgt varchar(64)
);
DROP TABLE IF EXISTS dbo.stg_municipalities;
CREATE TABLE dbo.stg_municipalities (
        cod_ine varchar(15) NOT NULL,
        cod_geo varchar(5),
        cod_prov varchar(2),
        provincie varchar(50),
        name_muni varchar(50),
        population_muni integer ,
        surface varchar(50),
        perimeter integer,
        cod_ine_capital varchar,
        capital varchar(50),
        population_capital integer,
        longitude_etrs89 varchar(50),
        latitude_etrs89 varchar(50),
        altitude integer
);
DROP TABLE IF EXISTS dbo.stg_vehicles;
CREATE TABLE dbo.stg_vehicles (
```

```
        id integer NOT null,
        municipality varchar(5),
        brand varchar(25),
        model varchar(25),
        registration_date date,
        new_used boolean,
        subtype varchar(2)
);
```

1.4. Carga de los ficheros

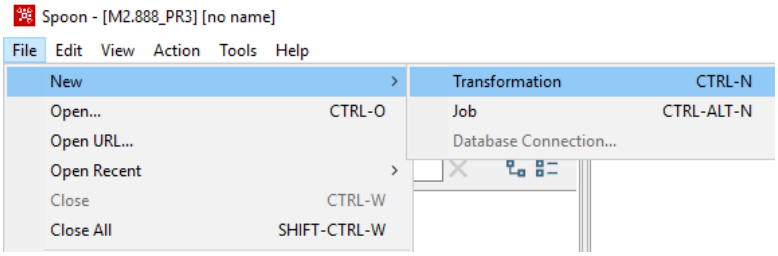
Una vez creadas las tablas en la base de datos PostgreSQL, a continuación, se describe parte del desarrollo de las transformaciones cuyo objetivo es cargar uno de los orígenes de los datos identificados, en las tablas el área intermedia (*staging area*).

IN_DGT_TYPE

El proceso de transformación de ***IN_DGT_TYPE*** contiene las siguientes operaciones:

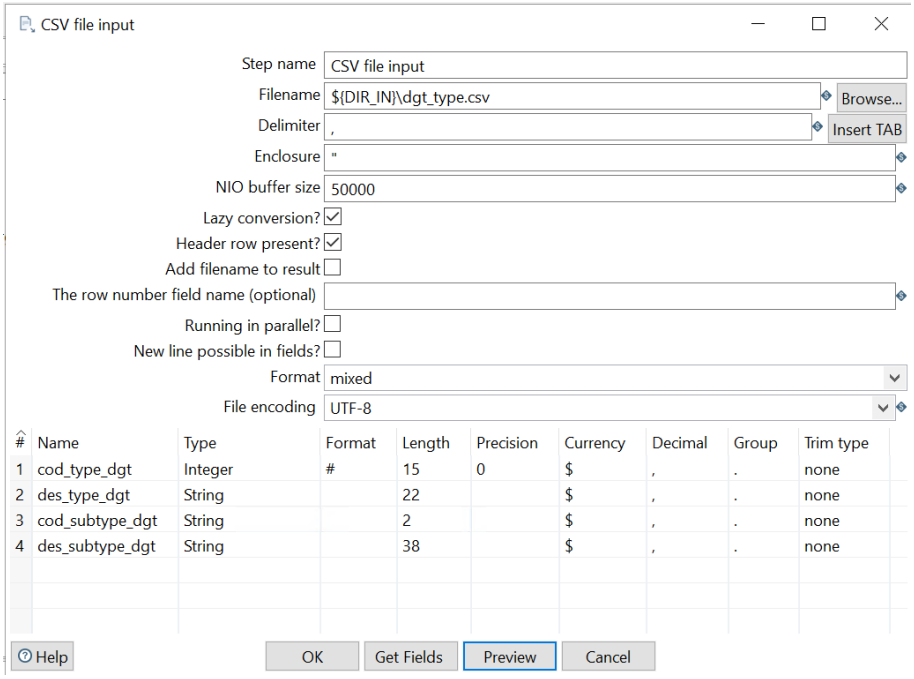
- Lectura del fichero de entrada *dgt_type.csv* (*CSV file input*)
- Carga a la tabla intermedia *stg_dgt_type* (*Table output*)

En primer lugar, se crea una nueva transformación:

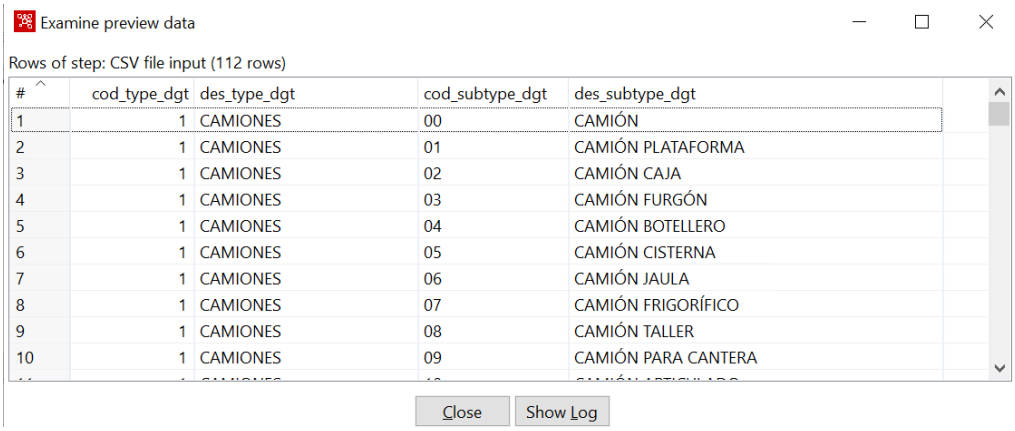


Como primer paso, se realizará la entrada del fichero **CSV**. Para ello, se usa el componente *CSV file input*. En este paso se indica el fichero desde donde se extraen los datos. Para ello, hay que utilizar la variable de entorno *DIR_IN*.

Para realizar correctamente la carga, se debe indicar el delimitador, si contiene cabecera y el resto de los parámetros como se muestra a continuación, se pulsará el botón *Get Fields* para obtener la estructura del fichero, y se pulsará el botón *Preview* para previsualizar los datos obtenidos:

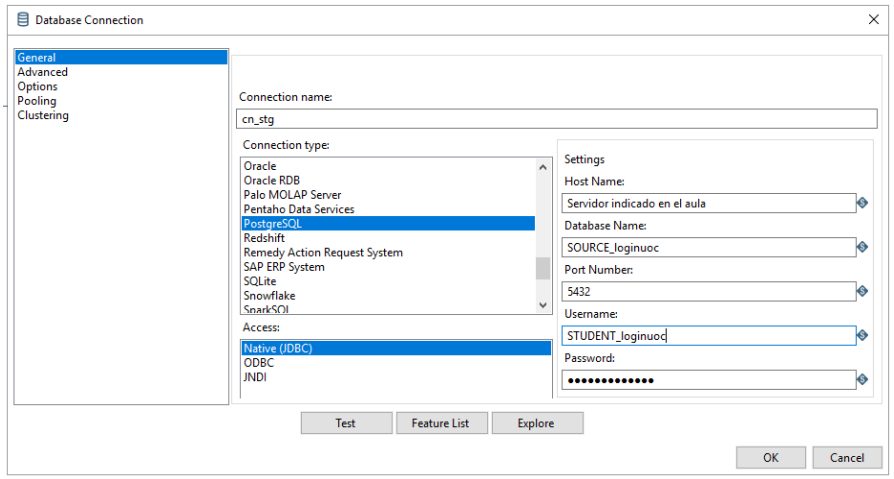


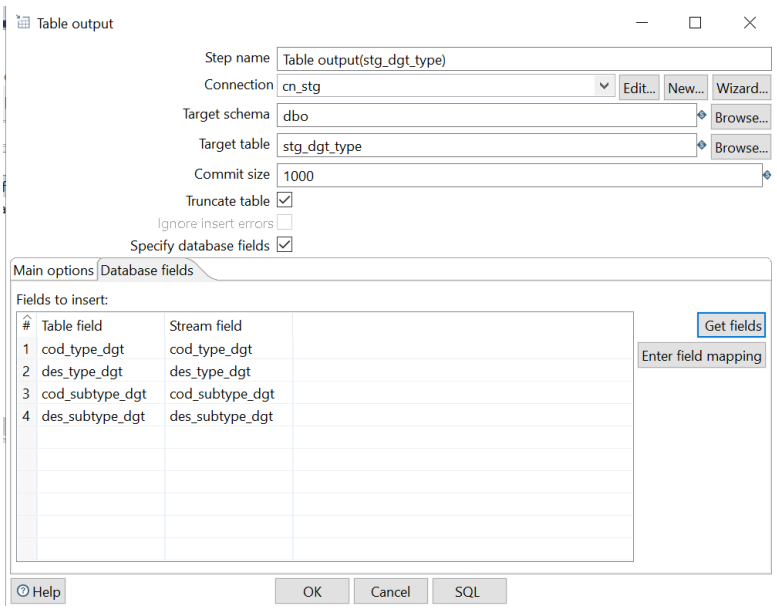
Importante: hay que indicar la codificación UTF-8



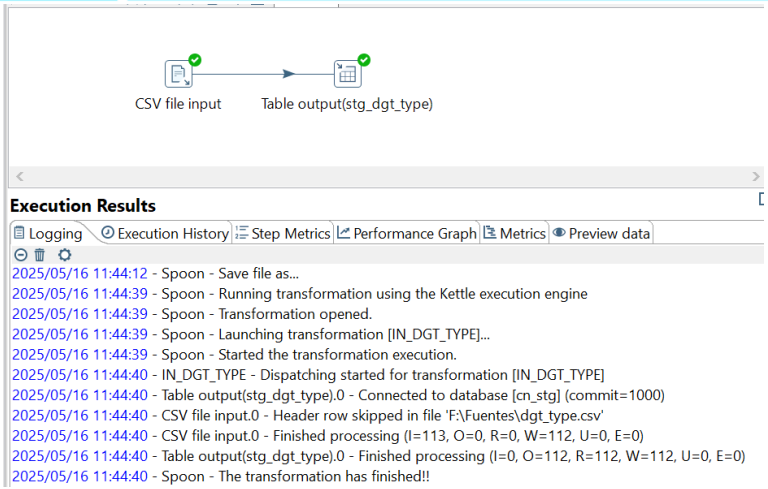
(la consulta muestra los 10 primeros registros)

Por último, se realizará la carga en la tabla intermedia *stg_dgt_type* del stage, utilizando el componente *Table Output*. Para ello se debe configurar la conexión, mapeando los campos disponibles en *Spoon* con los de la tabla de la base de datos y marcando las opciones *Truncate table* y *Specify database fields* (previo pulsar *Get Fields*) para poder indicar qué campos se quieren utilizar del flujo de la transformación *Stream fields* (datos que provienen del flujo de la transformación) y los campos correspondientes en la tabla de la base de datos *Table fields* (campos definidos en la tabla donde se alojarán los datos). En este último punto se deben ajustar el nombre de los campos a los campos de la tabla *stg_dgt_type*.





Una vez construida la transformación, se debe ejecutar para comprobar que funciona correctamente.



Así mismo se validará también que los datos hayan sido insertados correctamente en la base de datos PostgreSQL.

Query | Query History | Scratch Pad

```
SELECT * from dbo.stg_dgt_type
```

Data Output | Messages | Notifications

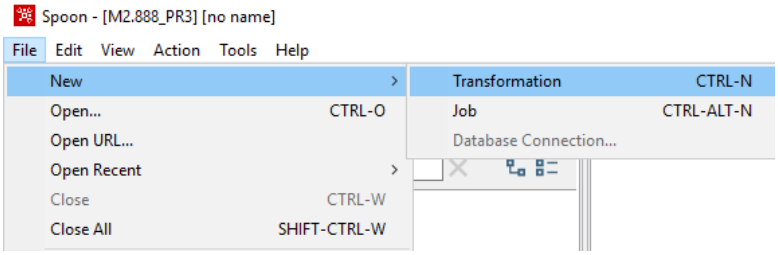
	cod_type_dgt character varying (64)	des_type_dgt character varying (64)	cod_subtype_dgt character varying (64)	des_subtype_dgt character varying (64)
1	1	CAMIONES	00	CAMIÓN
2	1	CAMIONES	01	CAMIÓN PLATAFORMA
3	1	CAMIONES	02	CAMIÓN CAJA
4	1	CAMIONES	03	CAMIÓN FURGÓN
5	1	CAMIONES	04	CAMIÓN BOTELLERO
6	1	CAMIONES	05	CAMIÓN CISTERNA
7	1	CAMIONES	06	CAMIÓN JAULA
8	1	CAMIONES	07	CAMIÓN FRIGORÍFICO
9	1	CAMIONES	08	CAMIÓN TALLER
10	1	CAMIONES	09	CAMIÓN PARA CANTERA

IN_MUNICIPALITIES

El proceso de transformación de ***IN_MUNICIPALITIES*** contiene las siguientes operaciones:

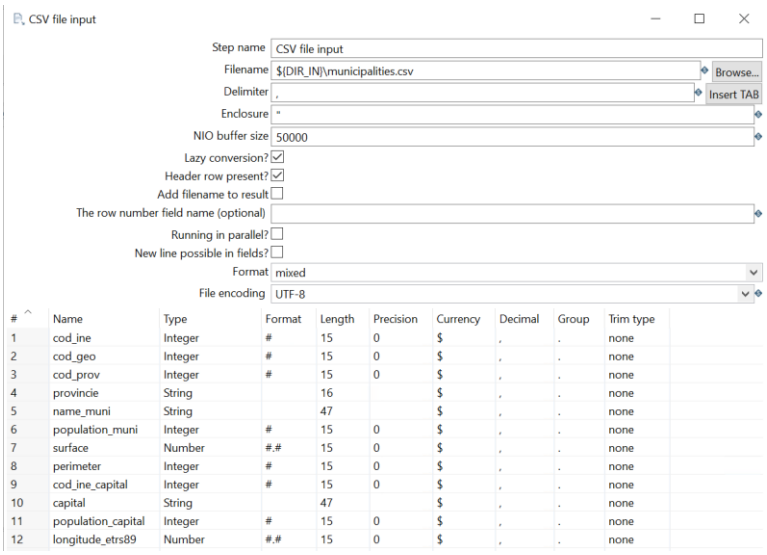
- Lectura del fichero de entrada *municipalities.csv* (*CSV file input*)
- Carga a la tabla intermedia *stg_municipalities* (*Table output*)

En primer lugar, se crea una nueva transformación:



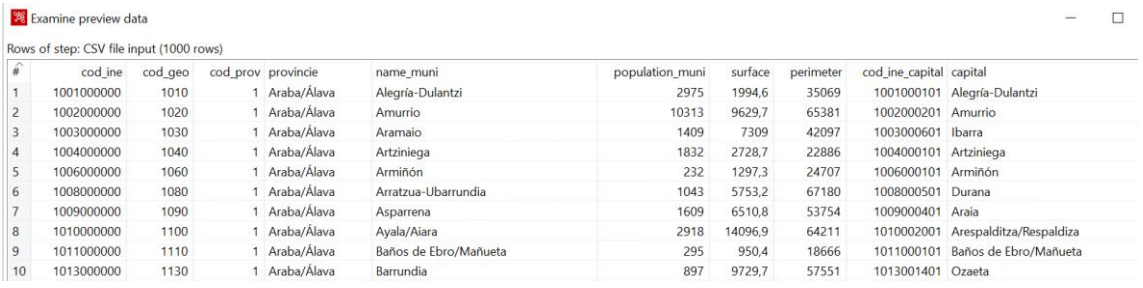
Como primer paso, se realizará la entrada del fichero **CSV**. Para ello, se usa el componente *CSV file input*. En este paso se indica el fichero desde donde se extraen los datos. Para ello, se debe utilizar la variable de entorno *DIR_IN*.

Para realizar correctamente la carga, se debe indicar el delimitador, si contiene cabecera y el resto de los parámetros como se muestra a continuación. Se pulsará el botón *Get Fields* para obtener la estructura del fichero, y se pulsará el botón *Preview* para previsualizar los datos obtenidos:



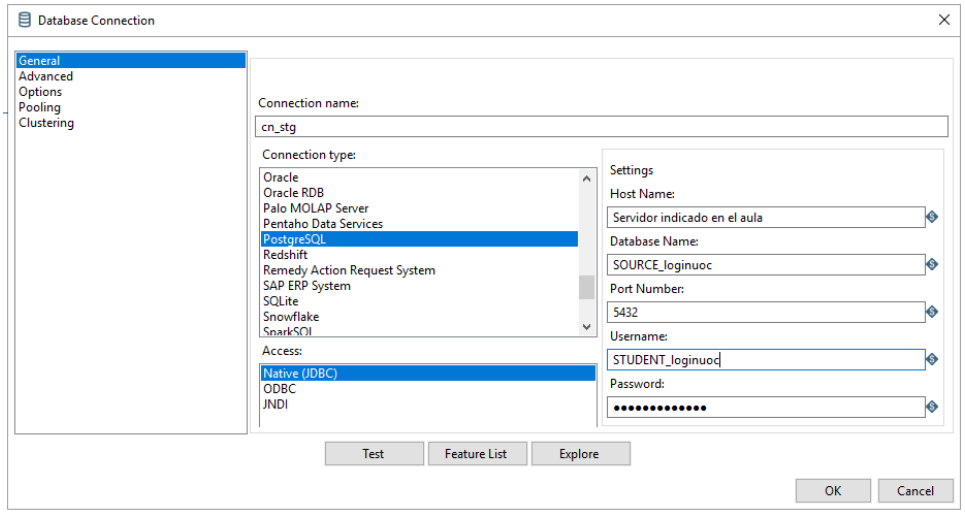
#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	cod_ine	Integer	#	15	0	$	,	.	none
2	cod_geo	Integer	#	15	0	$	,	.	none
3	cod_prov	Integer	#	15	0	$	,	.	none
4	provincie	String		16		$	,	.	none
5	name_muni	String		47		$	,	.	none
6	population_muni	Integer	#	15	0	$	,	.	none
7	surface	Number	#.#	15	0	$	,	.	none
8	perimeter	Integer	#	15	0	$	,	.	none
9	cod_ine_capital	Integer	#	15	0	$	,	.	none
10	capital	String		47		$	,	.	none
11	population_capital	Integer	#	15	0	$	,	.	none
12	longitude_etrs89	Number	#.#	15	0	$	,	.	none

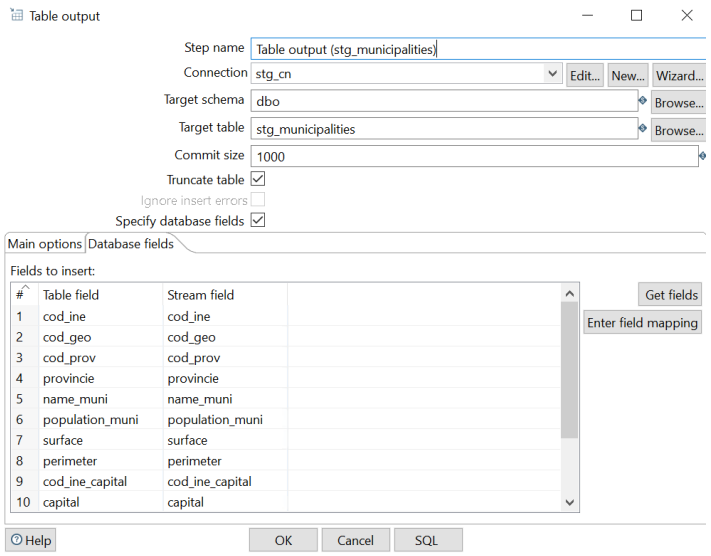
Importante: hay que indicar la codificación UTF-8



#	cod_ine	cod_geo	cod_prov	provincie	name_muni	population_muni	surface	perimeter	cod_ine_capital	capital
1	1001000000	1010	1	Araba/Álava	Alegría-Dulantzi	2975	1994,6	35069	1001000101	Alegría-Dulantzi
2	1002000000	1020	1	Araba/Álava	Amurrio	10313	9629,7	65381	1002000201	Amurrio
3	1003000000	1030	1	Araba/Álava	Aramaio	1409	7309	42097	1003000601	Ibarra
4	1004000000	1040	1	Araba/Álava	Artziniega	1832	2728,7	22886	1004000101	Artziniega
5	1006000000	1060	1	Araba/Álava	Armiñón	232	1297,3	24707	1006000101	Armiñón
6	1008000000	1080	1	Araba/Álava	Arratzua-Ubarrundia	1043	5753,2	67180	1008000501	Durana
7	1009000000	1090	1	Araba/Álava	Asparrena	1609	6510,8	53754	1009000401	Araia
8	1010000000	1100	1	Araba/Álava	Ayala/Aiara	2918	14096,9	64211	1010002001	Arespalditza/Respaldiza
9	1011000000	1110	1	Araba/Álava	Baños de Ebro/Mañueta	295	950,4	18666	1011000101	Baños de Ebro/Mañueta
10	1013000000	1130	1	Araba/Álava	Barrundia	897	9729,7	57551	1013001401	Ozaeta

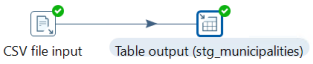
Por último, se realizará la carga en la tabla intermedia *stg_municipalities* del stage, utilizando el componente *Table Output*. Para ello se debe configurar la conexión, mapeando los campos disponibles en *Spoon* con los de la tabla de la base de datos y marcando las opciones *Truncate table* y *Specify database fields* (previo pulsar *Get Fields*) para poder indicar qué campos se quieren utilizar del flujo de la transformación *Stream fields* (datos que provienen del flujo de la transformación) y los campos correspondientes en la tabla de la base de datos *Table fields* (campos definidos en la tabla donde se alojarán los datos). En este último punto se deben ajustar el nombre de los campos a los campos de la tabla *stg_municipalities*.

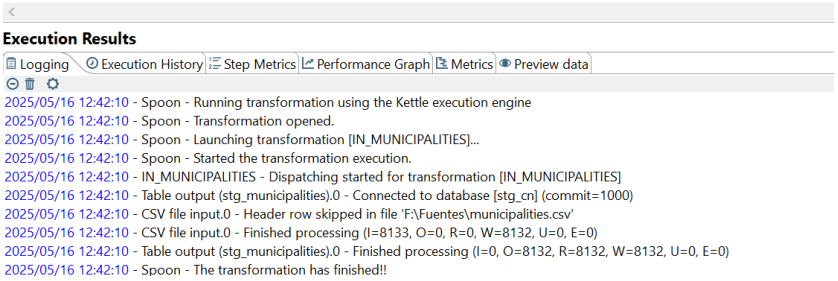




En las transformaciones de carga a las tablas intermedias (*stg_*) se debe activar la opción *Truncate table* del componente *Table Output*. Estas tablas no tienen relaciones de claves foráneas con otras tablas, por lo que pueden vaciarse sin restricciones antes de cargar nuevos datos.

Una vez construida la transformación, se ejecutará para comprobar su correcto funcionamiento.





También se debe validar que los datos se han insertado correctamente en la base de datos PostgreSQL.

```
SELECT * FROM dbo.stg_municipalities
```

Data Output | Messages | Notifications

	cod_ine character varying	cod_geo character varying (5)	cod_prov character varying (2)	provincie character varying (50)	name_muni character varying (50)
1	1001000000	1010	1	Araba/Álava	Alegría-Dulantzi
2	1002000000	1020	1	Araba/Álava	Amurrio
3	1003000000	1030	1	Araba/Álava	Aramaio
4	1004000000	1040	1	Araba/Álava	Artziniega
5	1006000000	1060	1	Araba/Álava	Armiñón
6	1008000000	1080	1	Araba/Álava	Arratzua-Ubarrundia
7	1009000000	1090	1	Araba/Álava	Asparrena
8	1010000000	1100	1	Araba/Álava	Ayala/Aiara
9	1011000000	1110	1	Araba/Álava	Baños de Ebro/Mañueta
10	1013000000	1130	1	Araba/Álava	Barrundia
11	1014000000	1140	1	Araba/Álava	Berantevilla
12	1016000000	1160	1	Araba/Álava	Bernedo

1.5. Ejercicios a realizar

De forma análoga a la carga de los ficheros cargados anteriormente (*municipalities.csv y dgt_types.csv)*, se solicita realizar el análisis del fichero ***vehicles.csv*** y su carga mediante la herramienta ETL ***Pentaho Spoon (Pentaho Data Integration)***.

En esta actividad se evaluará si el estudiantado es capaz de diseñar y ejecutar los procesos ETL para la carga a las tablas intermedias los ficheros *dgt_type.csv* y *municipalities.csv* utilizando el enunciado como guía. En la entrega deben de incluirse las capturas de pantalla de las métricas de la ejecución y su carga efectiva en las tablas PostgreSQL como evidencias de su realización. También en este ejercicio se evaluará el análisis de datos de la fuente *vehicles.csv* y la creación de la tabla intermedia y el diseño del proceso de ETL, incluyendo capturas de la realización de cada uno de estos apartados como evidencias y explicaciones de cada uno de los pasos mediante la rúbrica.

Con la realización tanto de los apartados de la guía (1.2. al 1.4) como el ejercicio 1.5 el estudiantado va a poder demostrar un buen conocimiento del diseño de procesos de transformación y carga de datos desde los ficheros origen a las tablas intermedias.

Ejercicio 2. Análisis de la calidad de los datos (20%)

Objetivo:

Conocer los diferentes mecanismos que ofrecen las bases de datos para poder analizar la calidad de los datos almacenados en las tablas intermedias para asegurar su calidad.

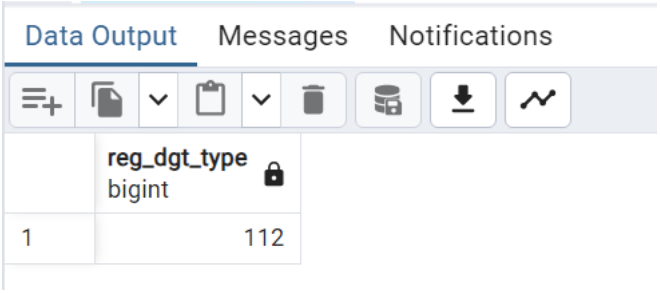
Indicaciones para el desarrollo del ejercicio

A continuación, se detallan los diferentes pasos necesarios para realizar el análisis de la calidad de datos cargados en la tabla *stg_dgt_type.* Para ello mediante consultas SQL en PostgreSQL se realiza el cálculo de la volumetría total, dado que esta tabla solo contiene columnas de tipo alfanumérico, se determina la completitud (volumetría y porcentaje de valores nulos por columna), las longitudes mínimas, máximas y valores únicos y por último, tras el análisis se realiza una la valoración de acciones a tener en cuenta en los procesos de integración y transformación para la carga de los datos de la tabla intermedia al modelo multidimensional.

- **Tabla *stg_dgt_type***

Para llevar a cabo la revisión de la completitud de datos (cantidad de los datos que están informados), es recomendable tener presente el número total de registros almacenados en la tabla *stg_dgt_type.*

```
SELECT
        count(1) as reg_dgt_type
FROM
        dbo.stg_dgt_type
```



Se calcula, por cada columna alfanumérica de stg_dgt_type, mediante expresiones comunes (*Common Table Expression WITH – CTE)*, la cantidad de valores no nulos,

nulos y únicos, así como las longitudes mínima y máxima de texto, cruzando esos resultados con el total de registros para calcular el porcentaje de nulos en cada columna.

Es interesante observar que se puede comparar la completitud y los valores únicos en varias columnas alfanuméricas de la misma tabla de una forma modular. La adición de la cláusula CROSS JOIN permite asegurar que la información de los nulos y el total de registros se obtenga para cada columna sin repetir cálculos. Es una operación de unión que devuelve el producto cartesiano de dos o más tablas.

```
WITH total_rows AS (

SELECT COUNT(*) AS total FROM dbo.stg_dgt_type
),
col_analysis AS (
-- Análisis de la calidad de datos de cod_type_dgt
SELECT
        'cod_type_dgt' AS columna,
        COUNT(cod_type_dgt) AS no_nulos,
        COUNT(distinct cod_type_dgt) AS valores_unicos,
        min(length(cod_type_dgt)) as min_length,
        max(length(cod_type_dgt)) as max_length
FROM
        dbo.stg_dgt_type
UNION ALL
        -- Análisis de la calidad de datos de des_type_dgt
SELECT
        'des_type_dgt' AS columna,
        COUNT(des_type_dgt) AS no_nulos,
        COUNT(distinct des_type_dgt) AS valores_unicos,
        min(length(des_type_dgt)) as min_length,
        max(length(des_type_dgt)) as max_length
FROM
        dbo.stg_dgt_type
UNION ALL
-- Análisis de la calidad de datos de cod_subtype_dgt
SELECT
        'cod_subtype_dgt' AS columna,
        COUNT(cod_subtype_dgt) AS no_nulos,
        COUNT(distinct cod_subtype_dgt) AS valores_unicos,
        min(length(cod_subtype_dgt)) as min_length,
        max(length(cod_subtype_dgt)) as max_length
FROM
        dbo.stg_dgt_type
UNION ALL
        -- Análisis de la calidad de datos de des_subtype_dgt
SELECT
        'des_subtype_dgt' AS columna,
        COUNT(des_subtype_dgt) AS no_nulos,
        COUNT(distinct des_subtype_dgt) AS valores_unicos,
        min(length(des_subtype_dgt)) as min_length,
```

```
        max(length(des_subtype_dgt)) as max_length
FROM
        dbo.stg_dgt_type
)
SELECT
  ca.columna,
  ca.no_nulos,
  tr.total - ca.no_nulos AS nulos,
  ROUND(100.0 * (tr.total - ca.no_nulos) / tr.total, 2) AS porcentaje_nulos,
  ca.valores_unicos,
  ca.min_length,
  ca.max_length
FROM col_analysis ca
CROSS JOIN total_rows tr
```

Data Output | Messages | Notifications

	columna text	no_nulos bigint	nulos bigint	porcentaje_nulos numeric	valores_unicos bigint	min_length integer	max_length integer
1	cod_type_dgt	112	0	0.00	8	1	1
2	des_type_dgt	112	0	0.00	9	5	22
3	cod_subtype_dgt	112	0	0.00	111	2	2
4	des_subtype_dgt	112	0	0.00	110	4	37

No se ha detectado ninguna columna con deficiencias de completitud (porcentaje_nulos = 0.00 en todas las columnas). No hay indicios preocupantes de duplicidad. No se observan valores atípicos en la longitud. La estructura de los datos es coherente y consistente con su significado esperado. Se aprecia que el campo *des_subtype_dgt* tiene casi valores únicos para cada fila (112 registros → 110 únicos), lo que indica una alta cardinalidad, esto es adecuado si cada descripción debe ser única (por ejemplo, nombres específicos), el resto de las columnas muestran cardinalidad baja o media, lo cual es típico de códigos o clasificaciones.

El análisis de valores únicos del campo *cod_subtype_dgt* (111 valores distintos), sugiere una revisión de la fuente de datos. Se detecta un caso específico de duplicidad: la presencia de los códigos s3 y S3, ambos asociados a la misma descripción *"SEMIRREMOLQUE FURGÓN"*. Esta duplicidad solo se diferencia por el uso de mayúsculas/minúsculas y no por significado, lo que indica un problema de estandarización en la fuente.

Esto implica la necesidad de aplicar un tratamiento específico en el proceso de carga de datos hacia la dimensión *dim_dgt_type*, normalizando el campo *cod_subtype_dgt* mediante conversión a mayúsculas.

En consecuencia, aunque el análisis de calidad no evidencia problemas generales de completitud o integridad, sí se requiere una acción correctiva de estandarización sobre *cod_subtype_dgt* para asegurar la consistencia y la integridad referencial durante los procesos de integración.

2.1 Ejercicios a realizar

Se solicita el análisis de la calidad de los datos stg_municipalities, stg_vehicles, teniendo presente cargados en las tablas *stg_municipalities, stg_vehicles* teniendo presente:

- la volumetría total
- el análisis de las columnas de tipo alfanumérico, de tipo numérico y fecha, determinando completitud (volumetría y porcentaje de valores nulos por columna), las longitudes mínimas, máximas y valores únicos
- el análisis de las columnas de tipo booleano, determinando la completitud (volumetría y porcentaje de valores nulos por columna) y valores únicos
- la valoración de acciones a tener en cuenta en los procesos de integración.

En esta actividad se evaluará la capacidad de realizar un análisis exhaustivo de la calidad de los datos almacenados en las tablas intermedias *stg_municipalities y stg_vehicles*. En particular, se valorará el cálculo de la volumetría total y el análisis detallado de cada columna según su tipo (alfanumérico, numérico, fecha, booleano), determinando la completitud (volumetría y porcentaje de valores nulos), longitudes mínimas y máximas, y número de valores únicos. Se valorará también la precisión en la detección de valores atípicos y la capacidad crítica en la elaboración de conclusiones claras y fundamentadas, que indiquen posibles acciones o transformaciones a realizar en futuros procesos de integración y transformación.

Ejercicio 3. Carga de una tabla de hechos y sus dimensiones en la base de datos (30%)

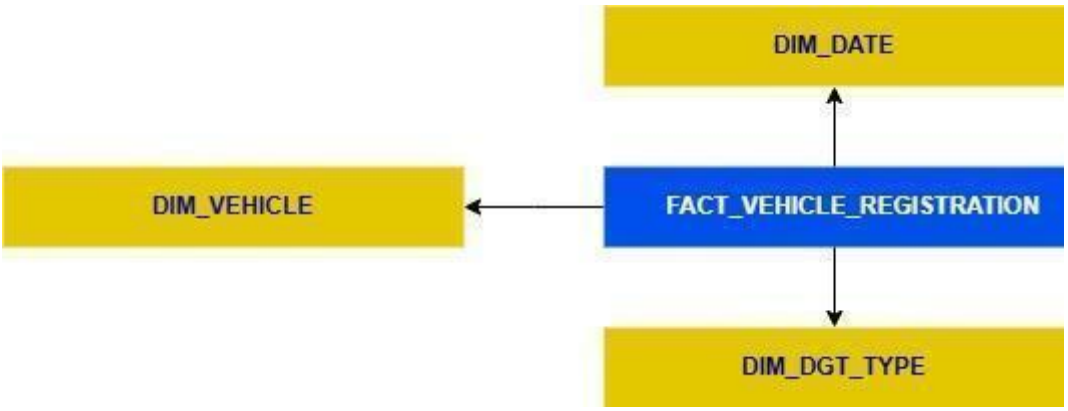
Objetivo:

Profundizar en el uso de la herramienta ETL Pentaho Spoon (Pentaho Data Integration), disponible en el entorno de laboratorio VDI, para diseñar y ejecutar procesos avanzados de integración y transformación de datos.

Desarrollo del ejercicio:

A continuación, se detallarán los diferentes pasos necesarios para la creación de una tabla de hechos y sus dimensiones vinculadas, así como trasladar los datos procedentes de la capa *staging* al modelo final.

En base a los datos origen y a los requerimientos, el modelo que se desea construir tiene el siguiente modelo conceptual, donde la tabla de hechos *FACT_VEHICLE_REGISTRATION* está relacionada en el diagrama de estrella con las dimensiones *DIM_DATE*, *DIM_VEHICLE*, *DIM_DGT_TYPE*.



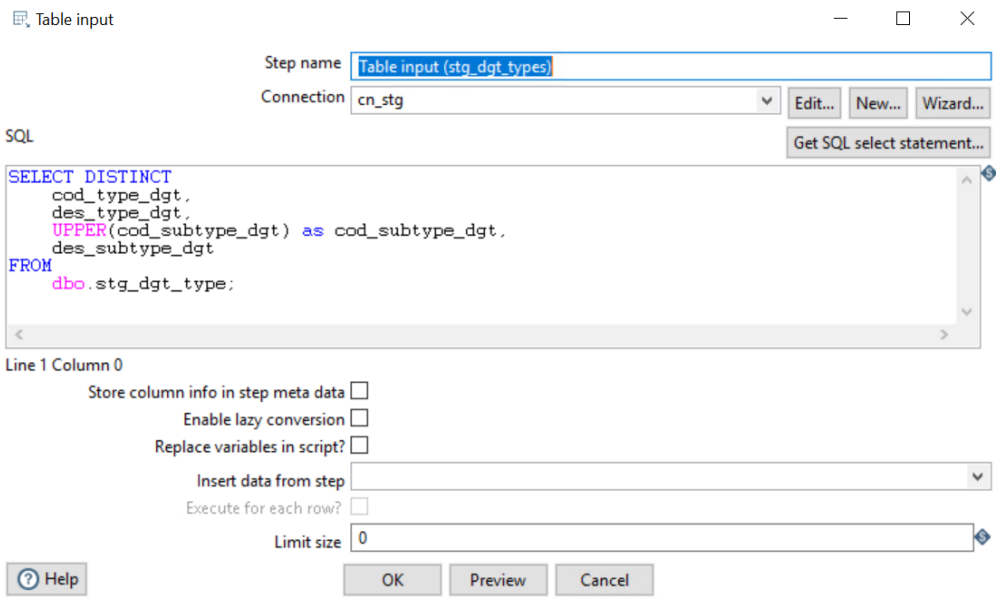
3.1 Transformación DIM_DGT_TYPE

Se describe el desarrollo de la transformación de *DIM_DGT_TYPES* utilizando *Spoon*. En esta transformación se cargarán los datos disponibles en la tabla *stg_dgt_type* a la tabla de dimensión *dim_dgt_type*. Para ello, se realizarán los tres pasos siguientes:

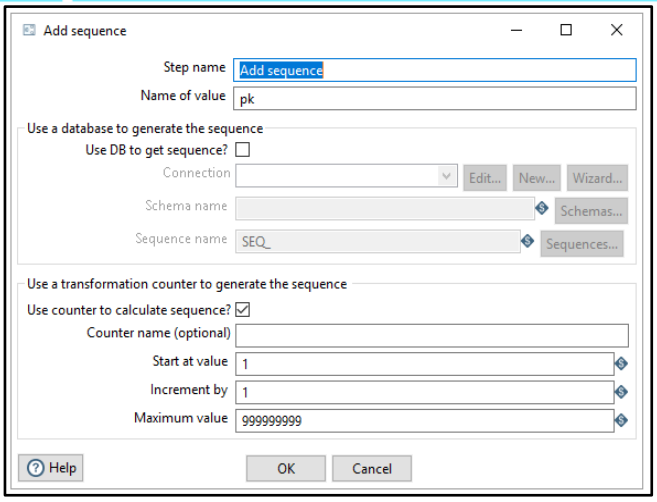
- Lectura de datos en la tabla *stg_dgt_type (Table input)*
- Añadir secuencia de datos (*Add sequence*)
- Insertar los datos en la tabla *dim_dgt_type (Table output)*.

Para realizar la lectura, se utilizará el componente ***Table Input***, que permite realizar una query directamente sobre la tabla de origen.

En este paso, además, se ha aplicado un proceso de normalización del campo *cod_subtype_dgt*, convirtiéndolo a mayúsculas mediante la función UPPER() en la propia query. Esta transformación garantiza la unificación de valores equivalentes (s3 / S3) antes de continuar con el flujo de carga hacia la dimensión, evitando duplicidades lógicas en el modelo analítico.



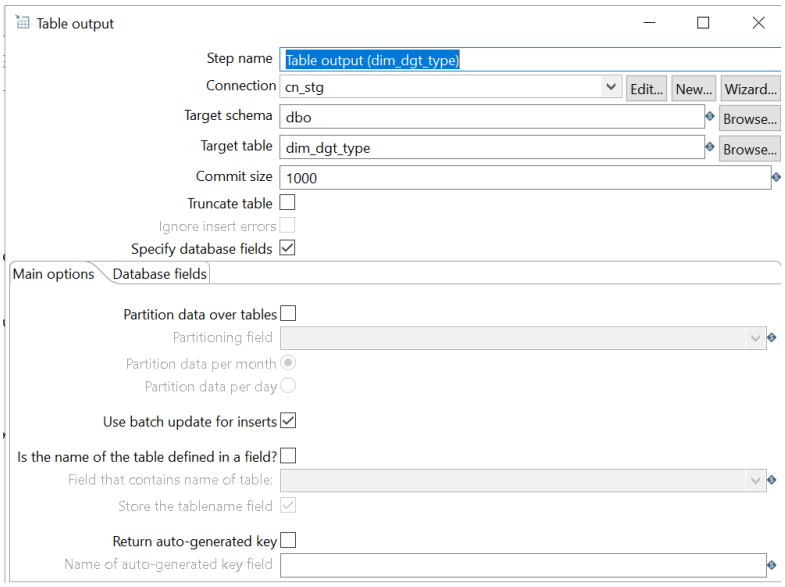
Luego se utiliza el componente ***Add sequence*** para añadir una secuencia que actuará como PK de la tabla.

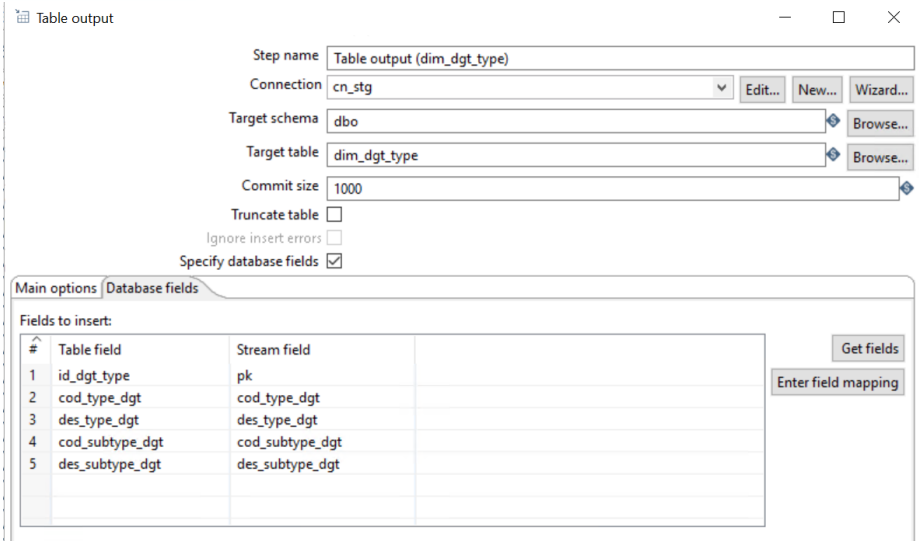


En base diseño del modelo en estrella, el script de creación de la tabla en la base de datos PostgreSQL será:

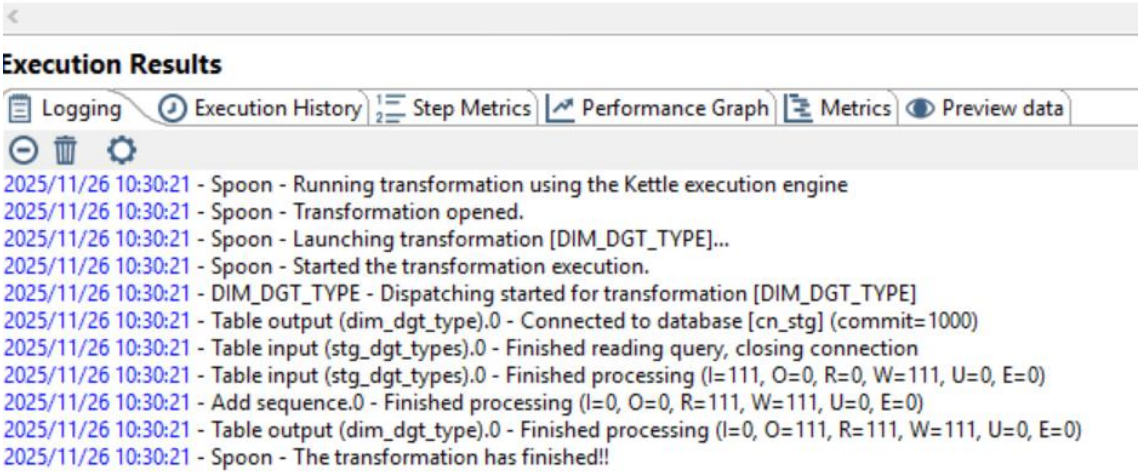
```
CREATE TABLE dbo.dim_dgt_type
(
   id_dgt_type INTEGER PRIMARY KEY,          -- Clave sustituta de la dimensión
   cod_type_dgt VARCHAR(1) NOT NULL,         -- Código del tipo DGT
   des_type_dgt VARCHAR(50) NOT NULL,        -- Descripción del tipo DGT
   cod_subtype_dgt VARCHAR(2) NOT NULL,      -- Código del subtipo DGT
   des_subtype_dgt VARCHAR(100) NOT NULL     -- Descripción del subtipo DGT
);
```

Finalmente, se realiza la carga de los datos en la tabla de dimensión correspondiente, especificando el mapeo de campos:





Una vez construida la transformación, debe ejecutarse para comprobar su correcto funcionamiento.



También se valida que los datos se han insertado correctamente en la base de datos PostgreSQL.

id_dgt_type [PK] integer	cod_type_dgt character varying (1)	des_type_dgt character varying (50)	cod_subtype_dgt character varying (2)	des_subtype_dgt character varying (100)
1	9	R Y S	SC	SEMIRREMOLQUE HORMIGONERA
2	9	R Y S	R3	REMOLQUE FURGÓN
3	9	CICLOMOTORES	90	CICLOMOTOR DE 2 RUEDAS
4	9	R Y S	S9	SEMIRREMOLQUE CANTERA
5	5	MOTOCICLETAS	51	MOTOCICLETA CON SIDECAR
6	9	R Y S	RE	REMOLQUE DE GRÚA
7	8	TRACTORES INDUSTRIALES	81	TRACTOCAMIÓN
8	1	CAMIONES	06	CAMIÓN JAULA
9	9	R Y S	SB	SEMIRREMOLQUE VIAJEROS O AUTOBÚS
10	0	OTROS vehiculos	7J	PALA CARGADORA-RETROEXCAVADORA
11	5	MOTOCICLETAS	60	VEHÍCULO DE MOVILIDAD PERSONAL
12	5	MOTOCICLETAS	54	CUATRICICLO PESADO

rows: 111 of 111 Query complete 00:00:00.719

3.2 Transformación DIM_DATE

A continuación, se describe el desarrollo de la transformación de *DIM_DATE* utilizando *Spoon*. En esta transformación se cargarán los datos que permiten realizar el análisis temporal por una fecha de registro del vehículo, por año, mes y día. La carga de esta dimensión es algo diferente a la carga de las dimensiones de datos. Hay diferentes opciones para cargar las tablas de tiempo. En esta solución, dado que se creará una dimensión temporal sencilla, se utilizará un *script* SQL para generar todos los registros necesarios.

Previamente a la carga de datos a la dimensión temporal, es necesaria la creación de la tabla en la base de datos PostgreSQL:

```
CREATE TABLE dbo.dim_date (
    date DATE PRIMARY KEY,   -- Clave primaria basada en la fecha
    year INTEGER NOT NULL,   -- Año (ej. 2024)
    month INTEGER NOT NULL,  -- Mes numérico (1-12)
    day INTEGER NOT NULL     -- Día del mes (1-31)
);
```

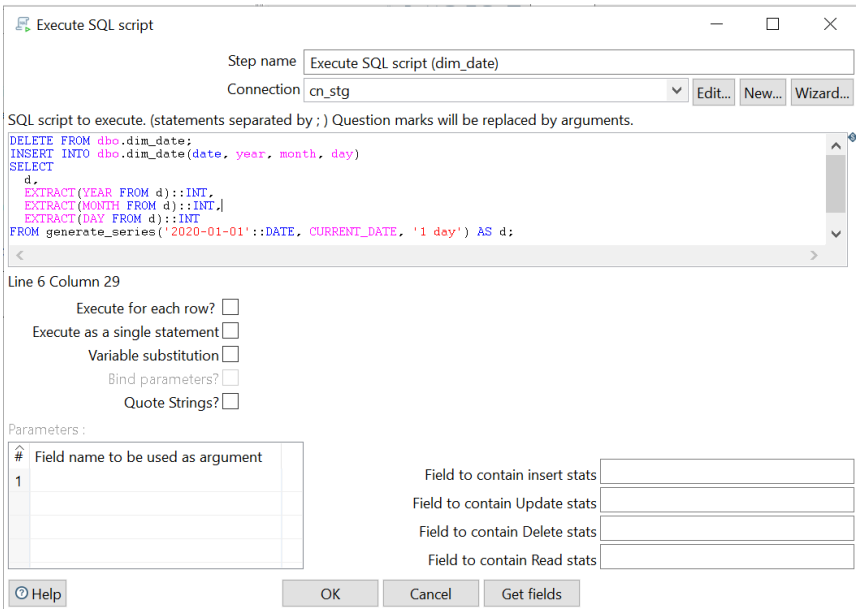
En esta transformación se cargarán los datos en DIM_DATE mediante la ejecución de un script teniendo en cuenta la mínima fecha de registro de vehículos (*registration_date*) hasta una fecha de fin que corresponderá al día de ejecución de la transformación.

Para ello, se realizarán un único paso: **Execute SQL script**

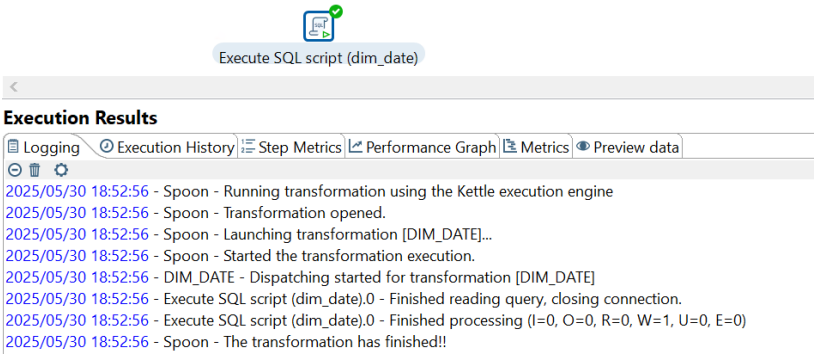
El script SQL completo es el siguiente:

```
DELETE FROM dbo.dim_date;
INSERT INTO dbo.dim_date(date, year, month, day)
SELECT
 d,
 EXTRACT(YEAR FROM d)::INT,
 EXTRACT(MONTH FROM d)::INT,
 EXTRACT(DAY FROM d)::INT
FROM generate_series('2020-01-01'::DATE, CURRENT_DATE, '1 day') AS d;
```

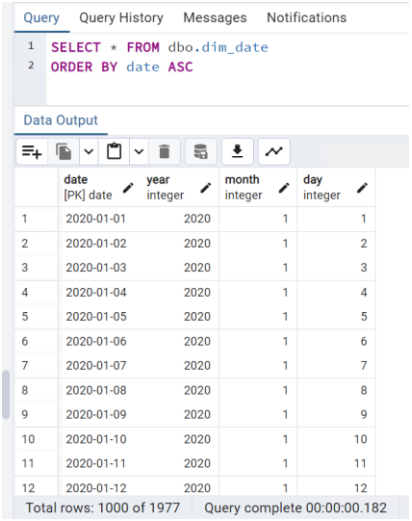
Para realizar la ejecución del script SQL se utilizará el componente **Execute SQL script**, que permite ejecutar una DML directamente para insertar los registros en una tabla de la base de datos.



Una vez construida la transformación, se ejecutará para comprobar su correcto funcionamiento.



Y se comprueba la correcta inserción de los datos en la tabla correspondiente en la base de datos PostgreSQL.

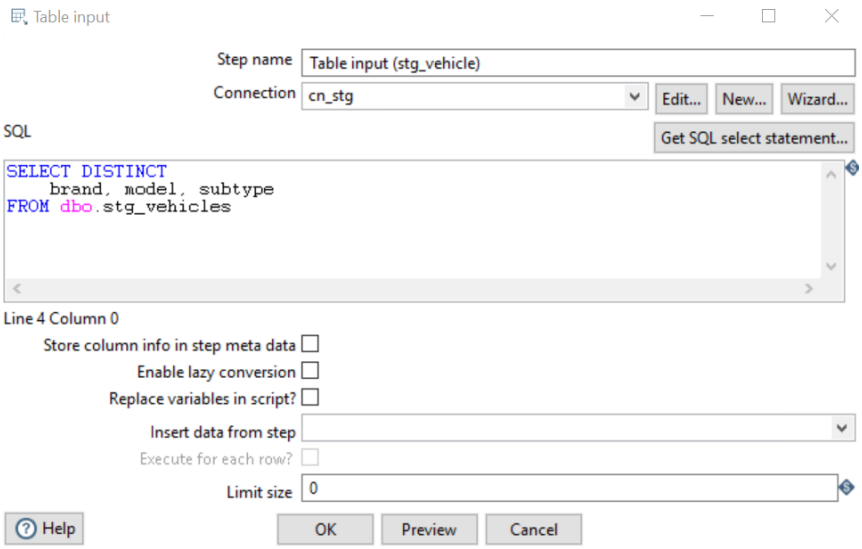


3.3 Transformación *DIM_VEHICLE*

En esta transformación se cargarán los datos disponibles en la tabla *stg_vehicles* a la tabla de dimensión *DIM_VEHICLE*. Para ello, se realizarán los tres pasos siguientes:

- Lectura de los datos en la tabla *stg_vehicles (Table Input)*
- Añadir secuencia de datos (*Add sequence*)
- Insertar los datos en la tabla *dim_vehicle* (*Table Output*)

Para realizar la lectura, se utilizará el componente ***Table Input***, que permite realizar una query directamente a la tabla de la base de datos.



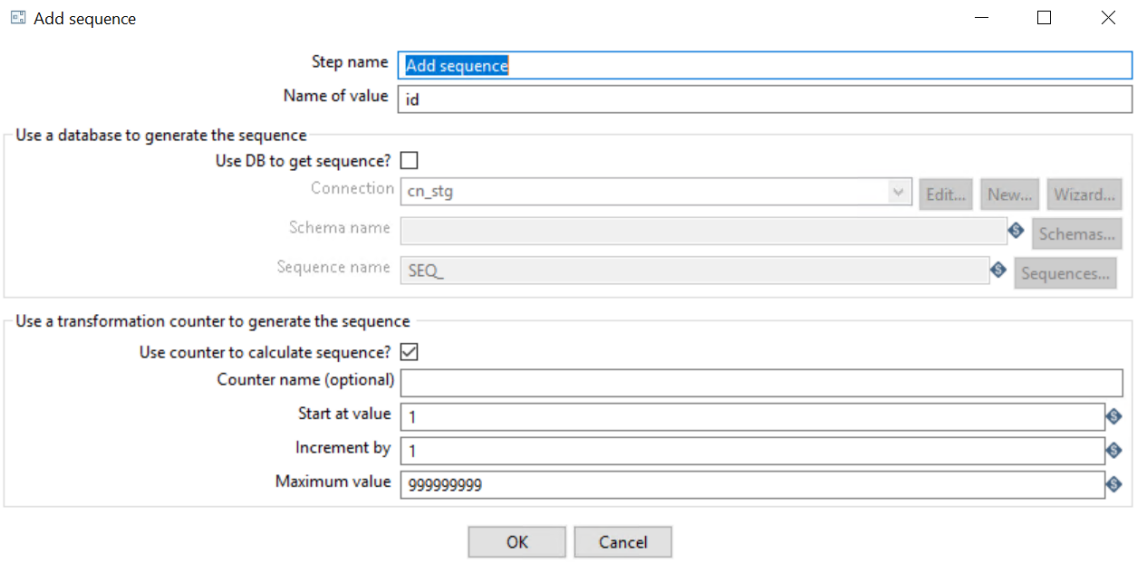
En base diseño del modelo en estrella, el script de creación de la tabla en la base de datos PostgreSQL será:

```
CREATE TABLE dbo.dim_vehicle (
   vehicle_id INTEGER PRIMARY KEY,      -- Clave
   brand VARCHAR(50),                   -- Marca del vehículo
   model VARCHAR(50),                   -- Modelo del vehículo
   subtype VARCHAR(2)                   -- Código de subtipo DGT (debe coincidir con cod_subtype_dgt)
);
```

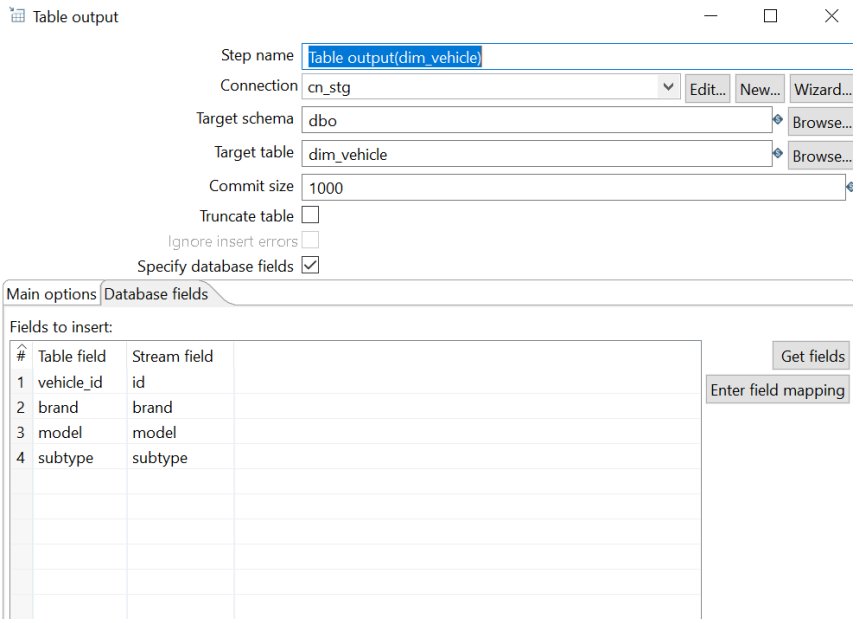
Notas:

- Se omiten campos como *municipality*, *registration_date* y new_used porque en un modelo en estrella esos atributos deben estar en otras tablas del modelo.
- Dado que la tabla *dim_dgt_type* define el campo *cod_subtype_dgt* como VARCHAR(2), y como el campo *subtype* de *stg_vehicles* corresponde a ese código conviene alinear su tamaño y tipo.

Luego se utiliza el componente ***Add sequence*** para añadir una secuencia que actuará como PK de la tabla.

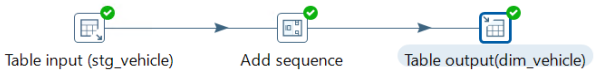


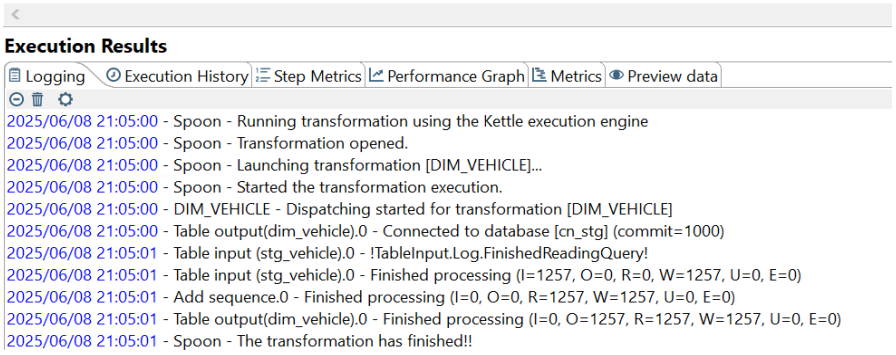
Finalmente, se realiza la carga de los datos en la tabla de dimensión correspondiente, especificando el mapeo de campos:



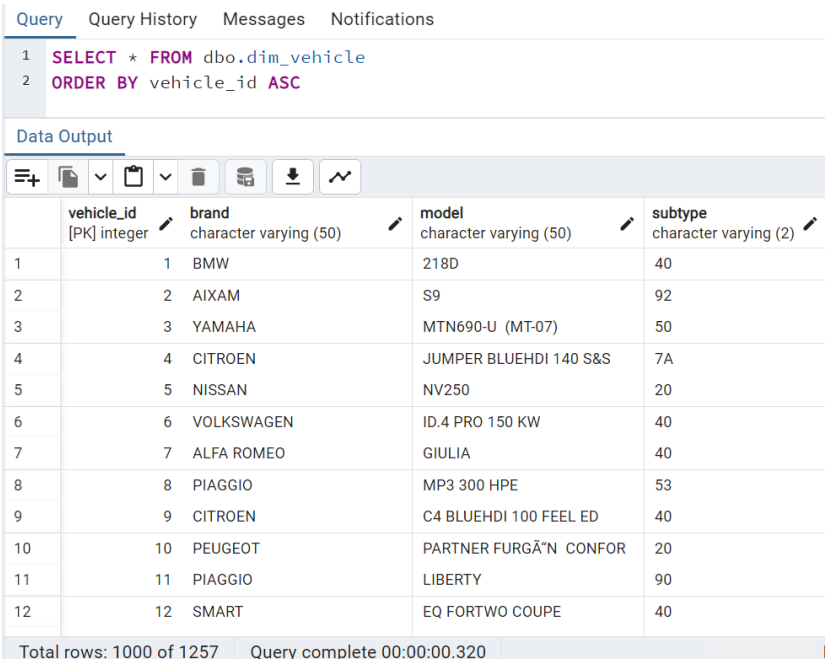
En las transformaciones de carga a las dimensiones (*dim_**), no se debe activar la opción de *Truncate table*. Estas tablas están referenciadas por la tabla de hechos a través de claves foráneas, y si se intenta truncarlas directamente se producirá un error de integridad referencial. Por ello, el vaciado de las dimensiones (y de la tabla de hechos) se realiza de forma controlada mediante una sentencia TRUNCATE TABLE…CASCADE.

Una vez construida la transformación, se ejecutará para comprobar su correcto funcionamiento.





Y se comprueba la correcta inserción de los datos en la tabla correspondiente en la base de datos PostgreSQL.



Query | Query History | Messages | Notifications

```
1 SELECT * FROM dbo.dim_vehicle
2 ORDER BY vehicle_id ASC
```

Data Output

	vehicle_id [PK] integer	brand character varying (50)	model character varying (50)	subtype character varying (2)
1	1	BMW	218D	40
2	2	AIXAM	S9	92
3	3	YAMAHA	MTN690-U (MT-07)	50
4	4	CITROEN	JUMPER BLUEHDI 140 S&S	7A
5	5	NISSAN	NV250	20
6	6	VOLKSWAGEN	ID.4 PRO 150 KW	40
7	7	ALFA ROMEO	GIULIA	40
8	8	PIAGGIO	MP3 300 HPE	53
9	9	CITROEN	C4 BLUEHDI 100 FEEL ED	40
10	10	PEUGEOT	PARTNER FURGÃ"N CONFOR	20
11	11	PIAGGIO	LIBERTY	90
12	12	SMART	EQ FORTWO COUPE	40

Total rows: 1000 of 1257 Query complete 00:00:00.320

3.4 Transformación FACT_VEHICLE_REGISTRATION

En este punto se describe el desarrollo de la transformación para la carga inicial de la tabla de hechos al almacén desde las tablas intermedias del *staging area*.

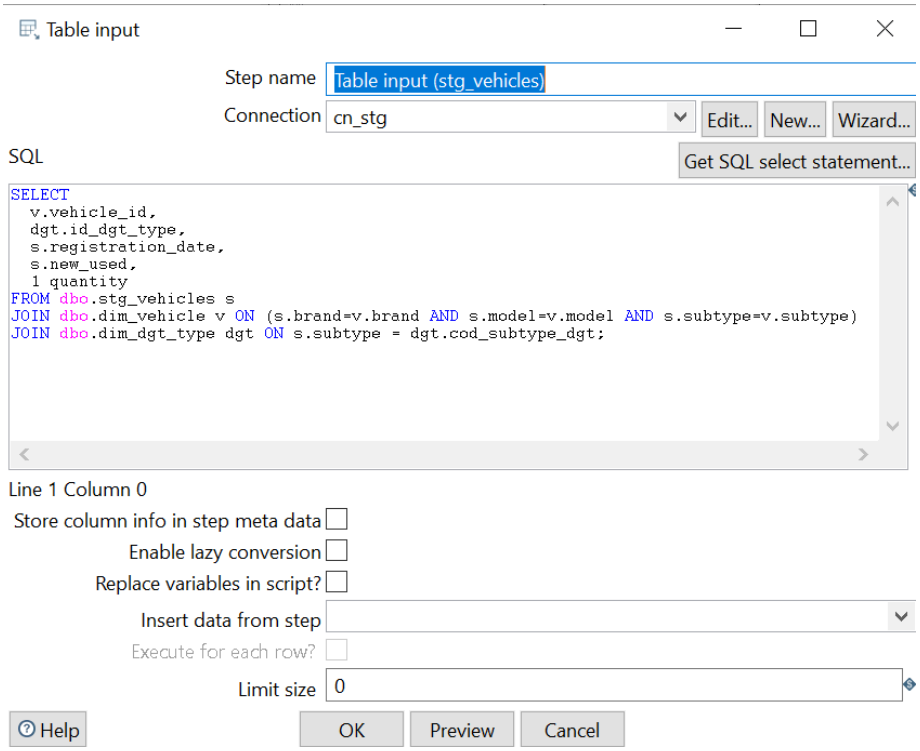
Con la implementación y ejecución de los procesos de carga de dimensiones se tendrá una gran cantidad de datos en nuestro modelo dimensional y se podrá pasar entonces a añadir los datos al modelo multidimensional, haciendo referencia a las dimensiones disponibles mediante sus claves foráneas.

El proceso de transformación de *FACT_VEHICLE_REGISTRATION* contiene las siguientes operaciones:

- Lectura de datos en la tabla *stg_vehicles (Table Input)*
- Añadir secuencia de datos (*Add sequence*)
- Carga en la tabla de hechos *fact_vehicle_registration* (Table output)

Con el primer paso se leen y preparan los datos desde la tabla de staging *stg_vehicles*, haciendo un INNER JOIN con *dim_vehicle* para recuperar la clave primaria vehicle_id,

y con dim_dgt_type para recuperar la clave *id_dgt_type* desde la dimensión *dim_dgt_type*, a partir del código de subtipo DGT (*subtype*). En la lectura también se añade el campo de medida *quantity* = 1 que usará la tabla de hechos y representa la cantidad de matriculaciones registradas para un vehículo, en una fecha concreta, con un determinado tipo DGT.

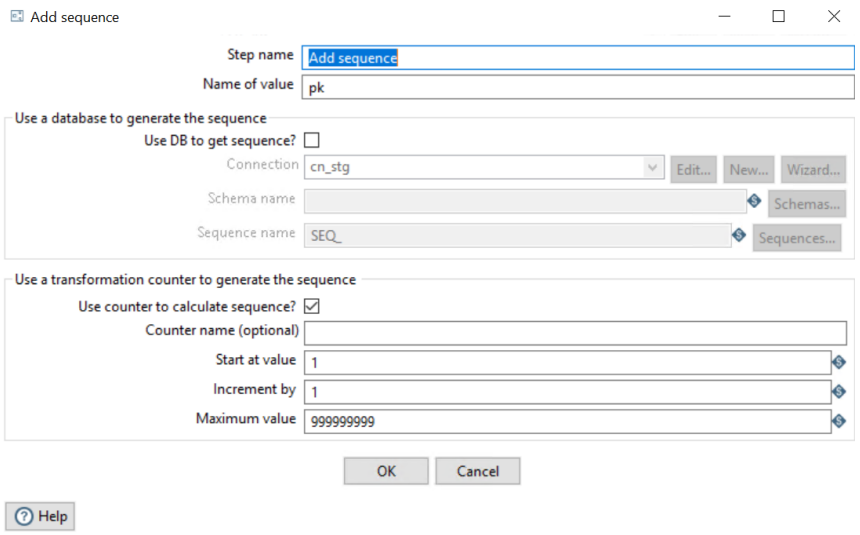


En base diseño del modelo en estrella, el script de creación de la tabla en la base de datos PostgreSQL será:

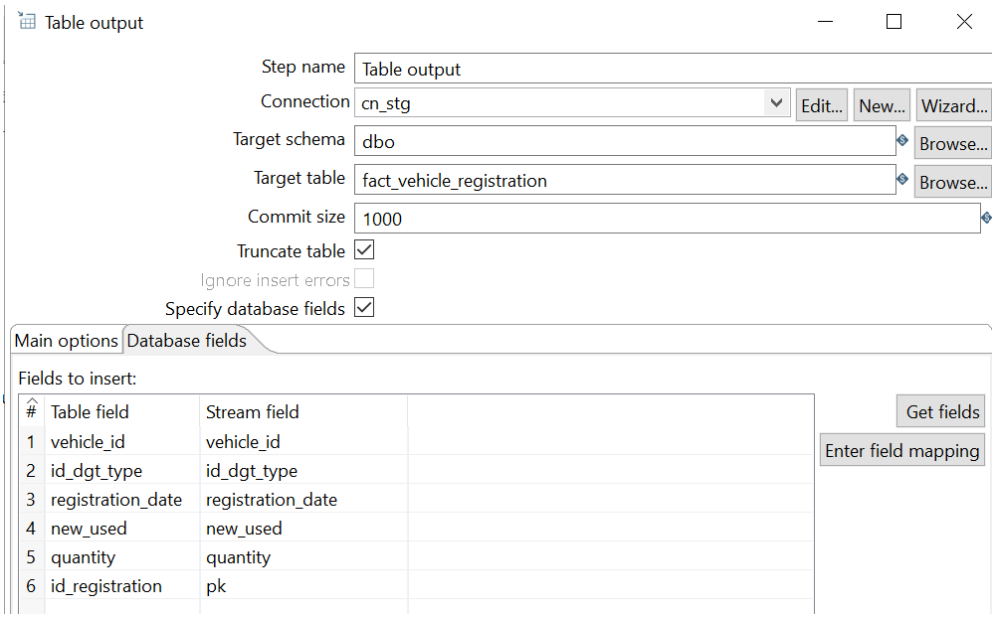
```
DROP TABLE IF EXISTS dbo.fact_vehicle_registration;
CREATE TABLE dbo.fact_vehicle_registration (
   id_registration  INT NOT NULL PRIMARY KEY,        --Clave
   registration_date date NOT NULL,                  -- Clave foránea hacia dim_date (clave natural)
   vehicle_id INT NOT NULL,                          -- Clave foránea hacia dim_vehicle
   id_dgt_type INT NOT NULL,                         -- Clave foránea hacia dim_dgt_type
   new_used     BOOLEAN,                             -- Si es nuevo o usado
   quantity INT NOT NULL DEFAULT 1,                  -- Medida (1 por cada registro de alta)

   FOREIGN KEY (registration_date) REFERENCES dbo.dim_date(date),
   FOREIGN KEY (vehicle_id) REFERENCES dbo.dim_vehicle(vehicle_id),
   FOREIGN KEY (id_dgt_type) REFERENCES dbo.dim_dgt_type(id_dgt_type)
);
```

Se utiliza el componente ***Add sequence*** para añadir una secuencia que actuará como PK de la tabla.



Finalmente, se realizará la carga de los datos en la tabla de hechos especificando el mapeo de campos y activando la opción *Truncate table*:



Una vez construida la transformación, se ejecutará para comprobar su correcto funcionamiento.

Execution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

```
2025/06/08 22:07:01 - Table output.0 - linenr 1450000
2025/06/08 22:07:03 - Table input (stg_vehicles).0 - !TableInput.Log.LineNumber!
2025/06/08 22:07:03 - Add sequence.0 - linenr 1500000
2025/06/08 22:07:04 - Table output.0 - linenr 1500000
2025/06/08 22:07:06 - Table input (stg_vehicles).0 - !TableInput.Log.LineNumber!
2025/06/08 22:07:06 - Add sequence.0 - linenr 1550000
2025/06/08 22:07:07 - Table output.0 - linenr 1550000
2025/06/08 22:07:08 - Table input (stg_vehicles).0 - !TableInput.Log.LineNumber!
2025/06/08 22:07:09 - Add sequence.0 - linenr 1600000
2025/06/08 22:07:10 - Table output.0 - linenr 1600000
2025/06/08 22:07:11 - Table input (stg_vehicles).0 - !TableInput.Log.LineNumber!
2025/06/08 22:07:12 - Add sequence.0 - linenr 1650000
2025/06/08 22:07:13 - Table output.0 - linenr 1650000
2025/06/08 22:07:14 - Table input (stg_vehicles).0 - !TableInput.Log.LineNumber!
2025/06/08 22:07:15 - Add sequence.0 - linenr 1700000
2025/06/08 22:07:16 - Table output.0 - linenr 1700000
2025/06/08 22:07:16 - Table input (stg_vehicles).0 - !TableInput.Log.FinishedReadingQuery!
2025/06/08 22:07:16 - Table input (stg_vehicles).0 - Finished processing (I=1721704, O=0, R=0, W=1721704, U=0, E=0)
2025/06/08 22:07:16 - Add sequence.0 - Finished processing (I=0, O=0, R=1721704, W=1721704, U=0, E=0)
2025/06/08 22:07:17 - Table output.0 - Finished processing (I=0, O=1721704, R=1721704, W=1721704, U=0, E=0)
2025/06/08 22:07:17 - Spoon - The transformation has finished!!
```

Luego se comprueba la correcta inserción de los datos en la tabla correspondiente en la base de datos PostgreSQL.

```
1 SELECT * FROM dbo.fact_vehicle_registration
2 ORDER BY id_registration ASC
```

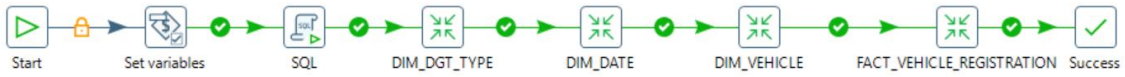
Data Output

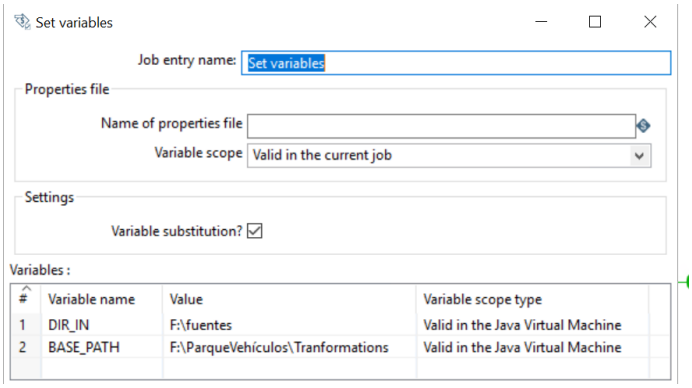
	id_registration [PK] integer	date date	vehicle_id integer	id_dgt_type integer	new_used boolean	quantity integer
1	1	2021-01-06	769	34	false	1
2	2	2021-01-06	769	34	false	1
3	3	2021-01-06	769	34	false	1
4	4	2021-01-06	769	34	false	1
5	5	2021-01-06	769	34	false	1
6	6	2021-01-06	769	34	false	1
7	7	2021-01-06	769	34	false	1
8	8	2021-01-06	769	34	false	1
9	9	2021-01-06	769	34	false	1
10	10	2021-01-06	769	34	false	1
11	11	2020-02-06	1008	34	false	1
12	12	2020-02-06	1008	34	false	1

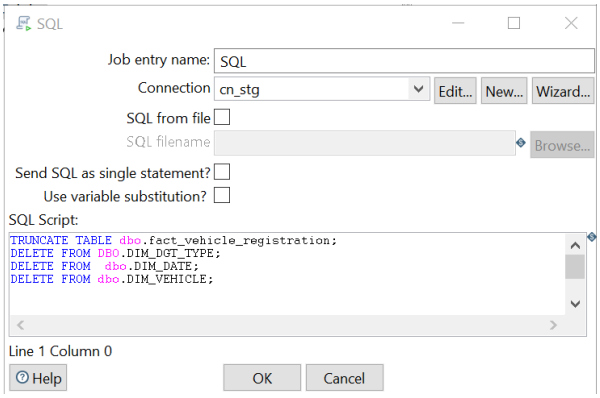
Total rows: 1000 of 1721704 Query complete 00:00:03.048

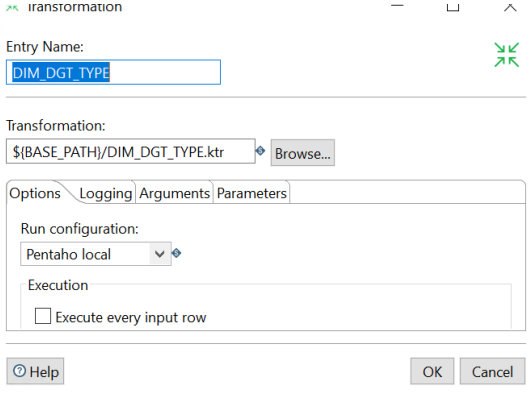
3.5 Job *JOB_LOAD*

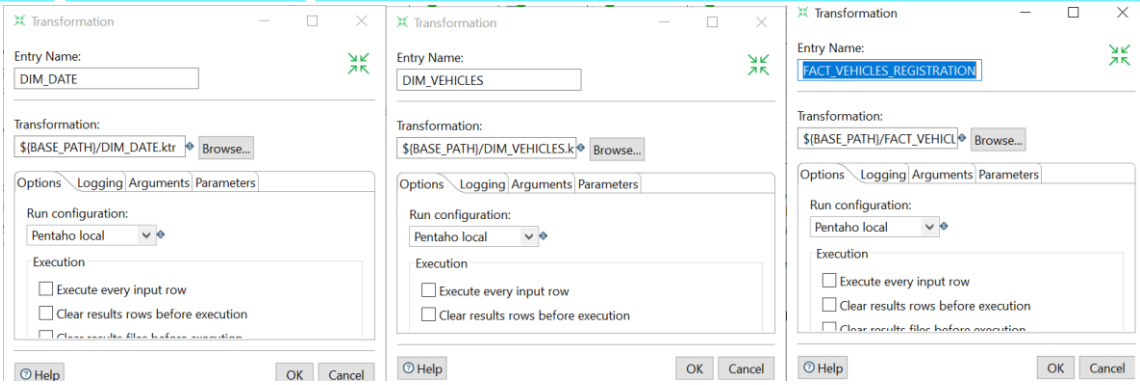
En esta sección, se diseñan los trabajos (*jobs*) mediante *Spoon*. Éstos van a permitir la ejecución secuencial de todos los procesos de ETL incluidos en cada bloque definido que permitan la carga automatizada de las dimensiones y tablas de hechos al *data mart*. El trabajo definido contiene como pasos cada una de las transformaciones implementadas en el apartado anterior de diseño de ETL.



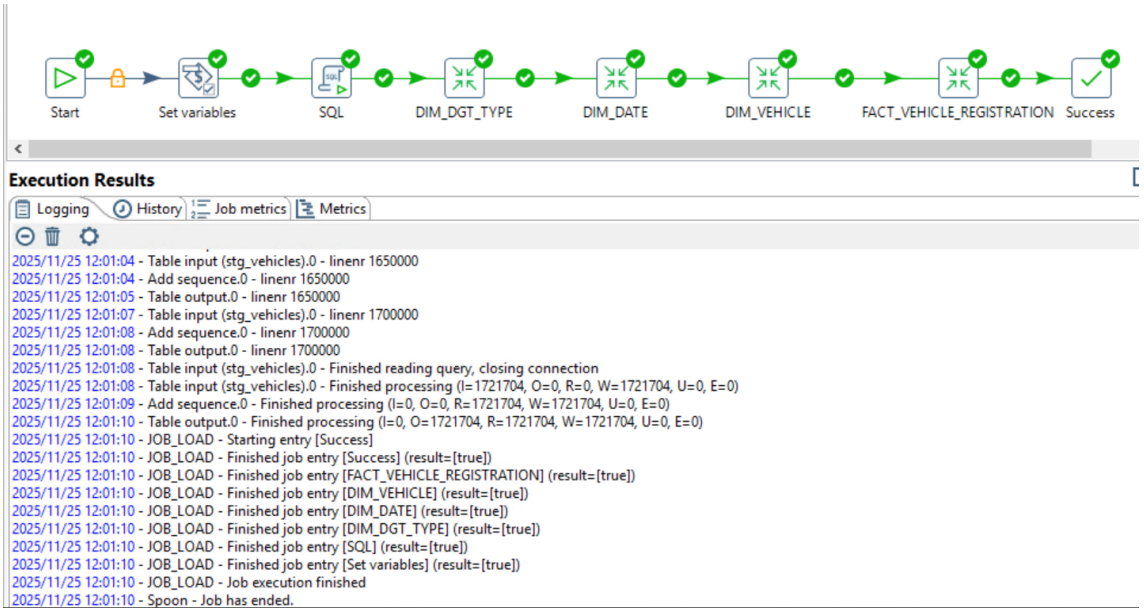








Una vez construida la transformación, se ejecutará para comprobar su correcto funcionamiento.



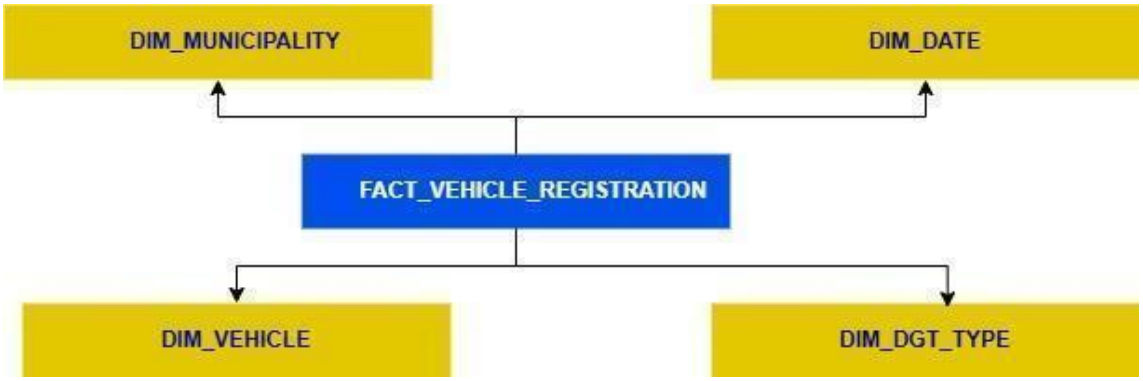
3.6 Ejercicios a realizar

1. Reproducir los puntos de la guía 3.1 a 3.3 incluyendo en la entrega las capturas de las métricas correspondientes a la ejecución correcta de cada una de las transformaciones para carga de datos a las dimensiones *DIM_DGT_TYPE, DIM_DATE* y *DIM_VEHICLE*.

2. Reproducir el punto de la guía 3.4 incluyendo en la entrega las capturas de las métricas correspondientes a la ejecución correcta de la transformación para la carga de datos a la tabla de hechos *FACT_VEHICLE_REGISTRATION*.

3. Reproducir el punto de la guía 3.5 incluyendo en la entrega las capturas de las métricas correspondientes a la ejecución correcta de la transformación para la carga de tabla de hechos *JOB_LOAD*.

4. Siguiendo la guía del enunciado, diseñar y ejecutar la transformación DIM_MUNICIPALITY que permita incorporar al modelo estrella la dimensión para la carga de los datos geográficos de los vehículos registrados en el parque móvil de la tabla intermedia *stg_municipalities*.



5. Adaptación de la transformación *FACT_VEHICLE_REGISTRATION* para obtener la vinculación entre la tabla de hechos y la dimensión *DIM_MUNICIPALITY*.

   Se deberán realizar las siguientes tareas:

   (a) Modificar la estructura de la tabla *FACT_VEHICLE_REGISTRATION* para incluir la nueva columna que vinculará la tabla de hechos con la tabla de dimensiones *DIM_MUNICIPALITY.*

   (b) Adaptar la transformación *FACT_VEHICLE_REGISTRATION* para cargar la clave principal de la dimensión *DIM_MUNICIPALITY* en la tabla de hechos.

   (c) Adaptar el *JOB_LOAD* para incorporar la carga de *DIM_MUNICIPALITY* y *FACT_VEHICLE_REGISTRATION modificada.*

6. En último lugar, con el objetivo de disponer de los datos almacenados en el *data warehouse* en la capa de publicación (*datamart*), se desea crear un nuevo proceso de integración que traslade toda la información almacenada en hechos

y dimensiones en el *datawarehouse* instalado en la base de datos **PostgreSQL** al *data mart* instalado en la base de datos **SQL Server**. Para ello se deberán seguir los siguientes pasos:

1) Creación de las tablas destino en SQLServer

```
use SOURCE_loginuoc; -- Cambiar por el usuario de la UOC

-- Eliminar tablas si existen previamente
IF OBJECT_ID('fact_vehicle_registration', 'U') IS NOT NULL DROP TABLE
fact_vehicle_registration;
IF OBJECT_ID('dim_dgt_type', 'U') IS NOT NULL DROP TABLE dim_dgt_type;
IF OBJECT_ID('dim_date', 'U') IS NOT NULL DROP TABLE dim_date;
IF OBJECT_ID('dim_vehicle', 'U') IS NOT NULL DROP TABLE dim_vehicle;
IF OBJECT_ID('dim_municipality', 'U') IS NOT NULL DROP TABLE dim_municipality;

CREATE TABLE dim_dgt_type
(
   id_dgt_type INTEGER PRIMARY KEY,              -- Clave sustituta de la dimensión
   cod_type_dgt VARCHAR(1) NOT NULL,              -- Código del tipo DGT
   des_type_dgt VARCHAR(50) NOT NULL,             -- Descripción del tipo DGT
   cod_subtype_dgt VARCHAR(2) NOT NULL,           -- Código del subtipo DGT
   des_subtype_dgt VARCHAR(100) NOT NULL          -- Descripción del subtipo DGT
);

CREATE TABLE dim_date (
   date DATE PRIMARY KEY,                        -- Clave primaria basada en la fecha
   year INTEGER NOT NULL,                         -- Año (ej. 2024)
   month INTEGER NOT NULL,                       -- Mes numérico (1-12)
   day INTEGER NOT NULL                          -- Día del mes (1-31)
);

CREATE TABLE dim_vehicle (
   vehicle_id INTEGER PRIMARY KEY,               -- Clave sustituta
   brand VARCHAR(50),                             -- Marca del vehículo
   model VARCHAR(50),                             -- Modelo del vehículo
  subtype VARCHAR(2)     -- Código de subtipo DGT (debe coincidir con cod_subtype_dgt)
);

CREATE TABLE dim_municipality (
 id_municipality INTEGER PRIMARY KEY,
 cod_ine VARCHAR(11),
 cod_geo VARCHAR(5),
 cod_prov VARCHAR(2),
 provincie VARCHAR(50),
 name_muni VARCHAR(50),
 population_muni INTEGER,
 surface VARCHAR(50),
 perimeter INTEGER,
 cod_ine_capital VARCHAR(11),
 capital VARCHAR(50),
 population_capital INTEGER,
```

```
 longitude_etrs89 VARCHAR(50),
 latitude_etrs89 VARCHAR(50),
 altitude INTEGER
);

CREATE TABLE fact_vehicle_registration (
   id_registration INT NOT NULL PRIMARY KEY,
   registration_date DATE NOT NULL,                -- Clave foránea hacia dim_date (clave
natural)
   vehicle_id INTEGER NOT NULL,                    -- Clave foránea hacia dim_vehicle
   new_used BIT,                                   -- Si es nuevo o usado
   id_dgt_type INTEGER NOT NULL,                   -- Clave foránea hacia dim_dgt_type
   id_municipality INTEGER NOT NULL,               -- Clave foránea hacia dim_municapality
   quantity INT NOT NULL DEFAULT 1                 -- Medida (puede ser 1 por cada registro
de alta)
);

 ALTER TABLE [fact_vehicle_registration] WITH CHECK ADD CONSTRAINT
[FK_FACT_DIM_VEHICLES] FOREIGN KEY([vehicle_id])
 REFERENCES [dim_vehicle] ([vehicle_id]);
 ALTER TABLE [fact_vehicle_registration] WITH CHECK ADD CONSTRAINT
[FK_FACT_DIM_DGT_TYPE] FOREIGN KEY([id_dgt_type])
 REFERENCES [DIM_DGT_TYPE] ([id_dgt_type]);
 ALTER TABLE [fact_vehicle_registration] WITH CHECK ADD CONSTRAINT
[FK_FACT_DIM_MUNICIPALITY] FOREIGN KEY([id_municipality])
 REFERENCES [DIM_MUNICIPALITY] ([id_municipality]);
ALTER TABLE fact_vehicle_registration WITH CHECK ADD CONSTRAINT
FK_FACT_DIM_DATE FOREIGN KEY (registration_date) REFERENCES dim_date (date);
```

2) Creación de las cuatro transformaciones de copia: DM_DIM_DATE, DM_DIM_VEHICLE, DM_DIM_MUNICIPALITY, DM_DIM_DGT_TYPE y DM_FACT_VEHICLE_REGISTRATION
3) Creación del job *JOB_DM_LOAD* que integre toda la ejecución de copia desde el Data Warehouse al *data mart*.

Ejercicio 4. Uso de consultas Common Table Expressions (CTE) para analizar los datos (30%)

Objetivo:

Conocer, entender y generar consultas *Common Table Expressions*, y las características y funcionalidades de las funciones analíticas, así como saber aplicarlas para la obtención de cálculos complejos.

4.1 Ejercicios a realizar:

Se solicitan las sentencias SQL en SQL Server que den respuesta a las siguientes cuestiones:

1. Conocer los diez primeros (top ten) municipios con mayor número de matriculaciones en la provincia de Barcelona ordenados de forma descendente por número de matriculaciones.
2. Repetir la consulta anterior para conocer el municipio con mayor número de matriculaciones de CAMIONES nuevos.
3. Conocer el total de matriculaciones de vehículos de tipo 'FURGONETAS' y de 'TURISMOS' por cada municipio de la provincia de Barcelona, ordenado por número de matriculaciones de 'FURGONETAS' y número de matriculaciones de 'TURISMOS' de forma descendente.
4. Desde un punto de vista estadístico, se desea analizar los municipios de la provincia de Barcelona para calcular en una única consulta:
   - El número total de matriculaciones por municipio.
   - El número de matriculaciones correspondientes a vehículos de tipo **MOTOCICLETAS** en cada municipio.

   A partir de las matriculaciones de MOTOCICLETAS por municipio, se calcula:
   - El porcentaje que representan sobre el total de matriculaciones del propio municipio.
   - El porcentaje que representan sobre el total general de matriculaciones de la provincia.

   Los resultados deben ordenarse de forma descendente según el número de matriculaciones de MOTOCICLETAS por municipio
5. Analizar desde un punto de vista estadístico la evolución de las matriculaciones del subtipo de vehículo "TURISMO".

   Para ello en una única consulta, calcular:
   - El total anual de matriculaciones de la provincia.
   - El total mensual de matriculaciones, desglosado por año y mes.
   - El total mensual de matriculaciones del subtipo "TURISMO", también por año y mes.

   A partir de estos valores, calcular:
   - El porcentaje que representan las matriculaciones mensuales de TURISMOS sobre el total de matriculaciones del mismo mes.

- El porcentaje que representan las matriculaciones mensuales de TURISMOS sobre el total de matriculaciones del año correspondiente.

Los resultados deben presentarse ordenados cronológicamente por año y mes.

En esta actividad se evaluará la capacidad de redactar consultas SQL utilizando CTE que respondan a las consultas planteadas sobre los datos almacenados. Se valorará la aplicación correcta de funciones agregadas, filtros, ordenaciones y cálculos estadísticos, así como la presentación clara y justificada y legible de cada consulta SQL. Se deberá incluir capturas de pantalla que muestran claramente la ejecución de cada consulta junto con los resultados obtenidos.

Criterios de evaluación

Parte teórica (30%)

Se detallan al inicio del cuestionario accesible desde el apartado *Contenidos del aula, **Entrega cuestionario PR3***.

Parte práctica (70%)

- Se valorará mediante un documento en formato Word o PDF que muestre evidencias de cumplimiento de los pasos indicados y aportados por el estudiante al resolver los ejercicios solicitados.
- Los aspectos a tener en cuenta están descritos al final de cada ejercicio a realizar.
- En el aula está disponible la rúbrica de corrección.

Nota final:

La nota final se calculará de acuerdo con esta fórmula:

```
NF = NotaCuestionario (30%) + NotaEjerciciosPrácticos (70%)

donde:
```

NotaEjerciciosPrácticos = Ejercicio 1 (10%) + Ejercicio 2 (10%) + Ejercicio 3 (30%) + Ejercicio 4 (20%)

Formato y fecha de entrega

La solución de la PR3 se entregará, a través del espacio "**Contenidos del aula**", en dos partes:

1. **Entrega Cuestionario PR3**
2. **Entrega práctica PR3.** Envío del documento solución de la parte práctica, en formato Word o PDF, a través del espacio **"Contenidos"** del aula. El nombre del fichero a entregar debe seguir el patrón: **BDA_PR3_Apellido1_Apellido2_Nombre.extensión.**

IMPORTANTE: Tal y como indica el plan docente:

- Es responsabilidad única del estudiante asegurarse que envía el cuestionario y entrega el documento con la solución en el espacio del aula habilitado para ese cometido.
- Entregas realizadas fuera de los canales indicados se considerarán como No presentadas.
- Entregas pasadas las 23:59 h de la fecha límite no serán aceptadas y por tanto, no podrán ser evaluadas.

La fecha máxima de entrega es el **17/12/2025** a las 23:59 horas.