



ugr

Universidad
de Granada

Sistemas Multidimensionales. **Práctica 3:**

*Herramientas ETL.
PDI (Pentaho Data Integration)*

Alberto Jesús Durán López

*Doble grado Ingeniería Informática y Matemáticas
Curso 2019-2020*

INDICE:

- Pasos previos	3
- Modificación de los campos	4
- Transformaciones	6
- Vista general Spoon	12
- Automatización mediante Jobs	13

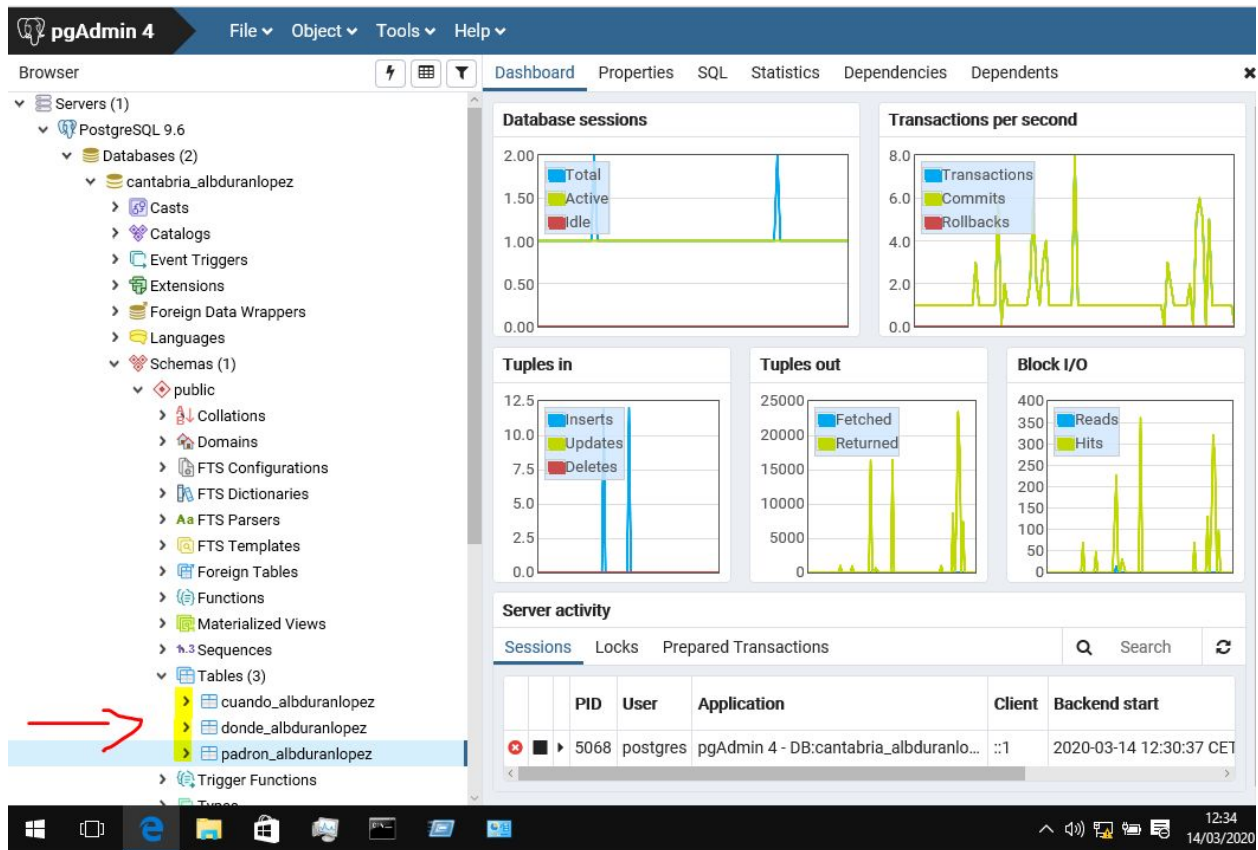
Transformaciones alternativas

- Ejercicio 2	15
- Ejercicio Libre.....	19
- Referencias	24

1- PASOS PREVIOS

Para la realización de la memoria se han incluido capturas de pantalla reflejando todos los pasos realizados.

Usando **pgAdmin**, creamos una base de datos (**cantabria_albduranlopez**) y, en el menú contextual asociado al esquema public, creamos las tablas que mostramos a continuación (**cuando**, **donde** y **padron** seguidos del prefijo **_albduranlopez**)



Una vez tenemos la BD, usamos Spoon para crear una nueva conexión con PostgreSQL y así aplicar transformaciones.

2- MODIFICACIÓN DE LOS CAMPOS

A continuación, creamos 3 entradas de Microsoft Excel y, para cada una de ellas usamos las hojas de Excel obtenidas de la práctica anterior como origen. Una vez hecho esto, sobrescribimos los campos usando minúsculas sin tilde y el carácter “_” como delimitador, es decir, usando el criterio *camel_case*. Mostramos el resultado de los tres input:

2.1- Cuando:

Step name: Input_cuando

#	Name	Type	Length	Precision	Trim type	Repeat	Format	Currency	Decimal	Grouping
1	id_cuando	Number			none	N				
2	año	String			none	N				
3	decada	String			none	N				

Get fields from header row...

Buttons: Help, OK, Preview rows, Cancel

2.2 - Donde:

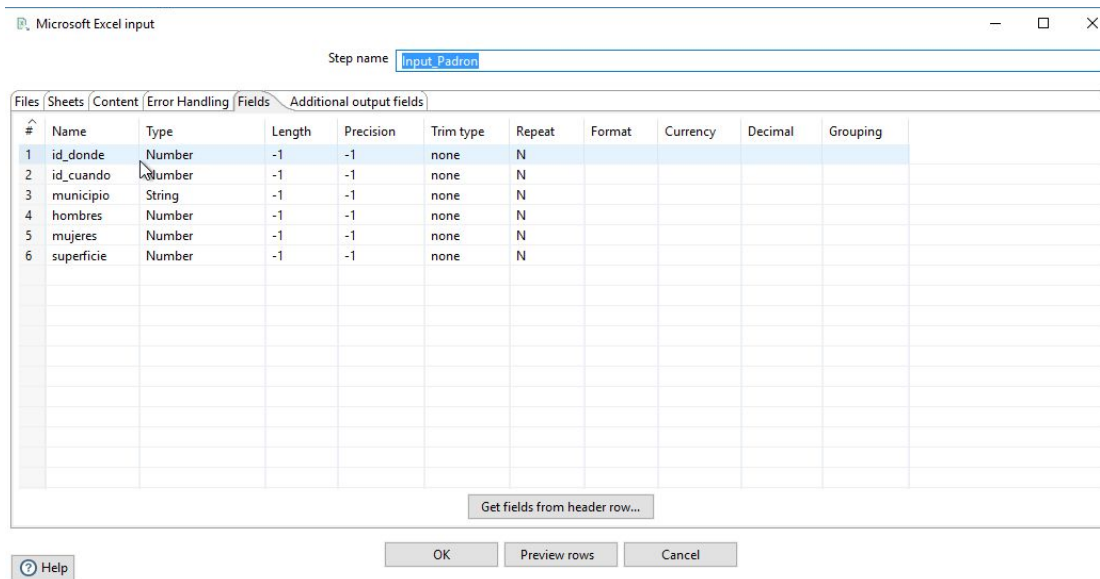
Step name: Input_donde

#	Name	Type	Length	Precision	Trim type	Repeat	Format	Currency	Decimal	Grouping
1	id_donde	Number			none	N				
2	cod_municipio	String			none	N				
3	municipio	String			none	N				
4	capital_municipio	String			none	N				
5	cod_comarca	Number			none	N				
6	comarca	String			none	N				
7	cod_provincia	String			none	N				
8	provincia	String			none	N				
9	capital_provincia	String			none	N				
10	cod_ca	String			none	N				
11	ca	String			none	N				
12	cod_tam_municipio	String			none	N				
13	tamaño_habitantes	String			none	N				
14	altitud	Number			none	N				
15	altitud_m	String			none	N				
16	superficie	Number			none	N				
17	superficie_km2	String			none	N				

Get fields from header row...

Buttons: Help, OK, Preview rows, Cancel

2.3 - Padron



Seleccionando el nodo de Input de Excel, podemos ver que se ha añadido correctamente pulsando sobre <<Preview data>> . Mostramos, por ejemplo, la tabla **padrón**:

Execution Results

Execution Results						
<div> Logging Execution History Step Metrics Performance Graph Metrics Preview data </div>						
<div> <input checked="" type="radio"/> First rows <input type="radio"/> Last rows <input type="radio"/> Off </div>						
#	id_donde	id_cuando	municipio	hombres	mujeres	superficie
1	1.0	1.0	Alfoz de Lloredo	1348.0	1298.0	46.5591
2	1.0	2.0	Alfoz de Lloredo	1332.0	1276.0	46.5591
3	2.0	1.0	Ampuero	1664.0	1673.0	32.466906
4	4.0	1.0	Arenas de Iguña	1084.0	1081.0	86.8871
5	9.0	1.0	Bárcena de Cicero	1140.0	1073.0	36.251423
6	3.0	1.0	Anievas	211.0	196.0	20.9664
7	5.0	1.0	Argoños	398.0	380.0	5.558
8	6.0	1.0	Arnuero	908.0	903.0	24.69626
9	7.0	1.0	Arredondo	364.0	304.0	46.881663
10	10.0	1.0	Bárcena de Pie de Concha	454.0	501.0	30.5447
11	8.0	1.0	Astillero, El	6408.0	6602.0	6.555353

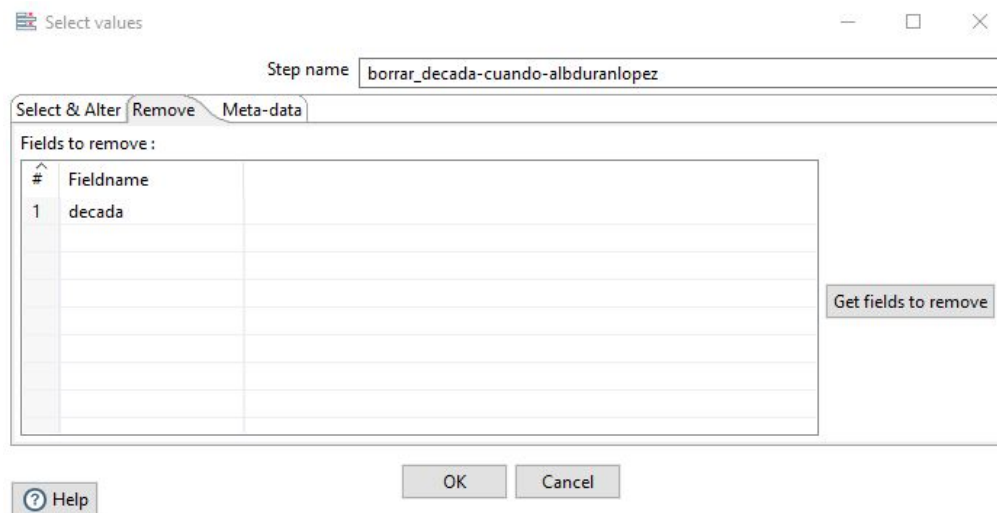
3- Transformaciones

Cabe destacar que para todas las transformaciones se ha tenido que ejecutar el botón de <<SQL>> y <<execute>> para ejecutar las sentencias de SQL. Además, se ha seleccionado la opción <<Truncate Table>> para que al realizar cualquier modificación se eliminen previamente los datos de la tabla.

3.1 - Transformación (1)

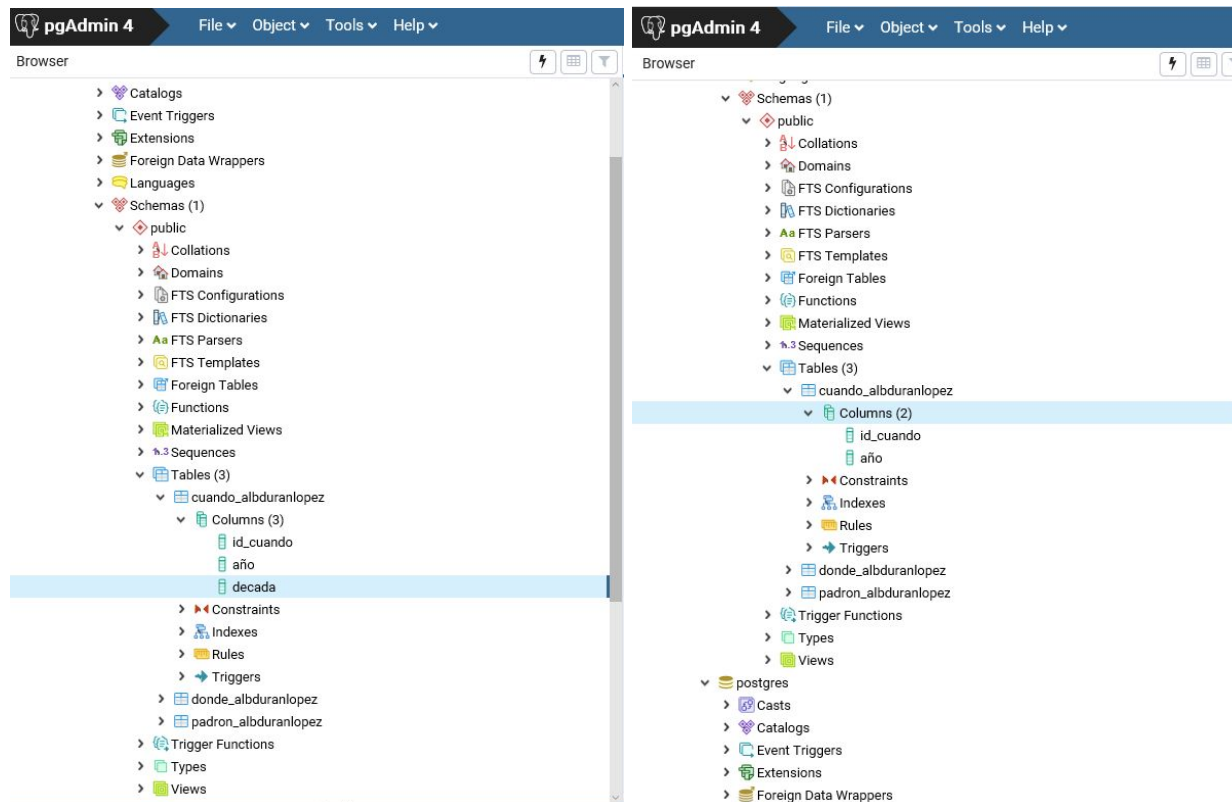


La primera transformación consiste en seleccionar las columnas que nos interesan. Nos quedamos con la columna **id_columna** y **año** por lo que **decada** se borraría. No hay pérdida de información porque ésta se puede obtener a partir del año. La configuración de la transformación **paso** es la siguiente:



Mostramos en la siguiente captura el resultado en **pgAdmin** tras ejecutar con **run** la transformación anterior. (hacer un previo <<SQL>> y <<execute>>)

En la captura de la izquierda se muestra el resultado antes de la transformación y en la derecha se muestra tras ejecutarla. Como podemos comprobar, se ha eliminado el campo **decada** de la tabla **cuando_albduranlopez**



3.2 - Transformación (2)



Esta transformación consistirá en separar un campo en otros 2. Observando el campo **altitud_m**, vemos que está formado por un código y por una descripción que se pueden separar por una 'coma'. Usamos la 'coma' como separador para obtener así dos nuevos campos que los llamaremos **cod_altitud** y **descripcion_altitud**.

El **cod_altitud** estará formado por un número entero [1,4], por tanto introducimos **length=1**

Bien es cierto que en la práctica anterior estos dos campos estaban por separado y se unieron, sin embargo, el objetivo de esta transformación no es otro sino el de mostrar la utilidad de esta herramienta.

Split fields

Step name:

Field to split:

Delimiter:

Enclosure:

Fields

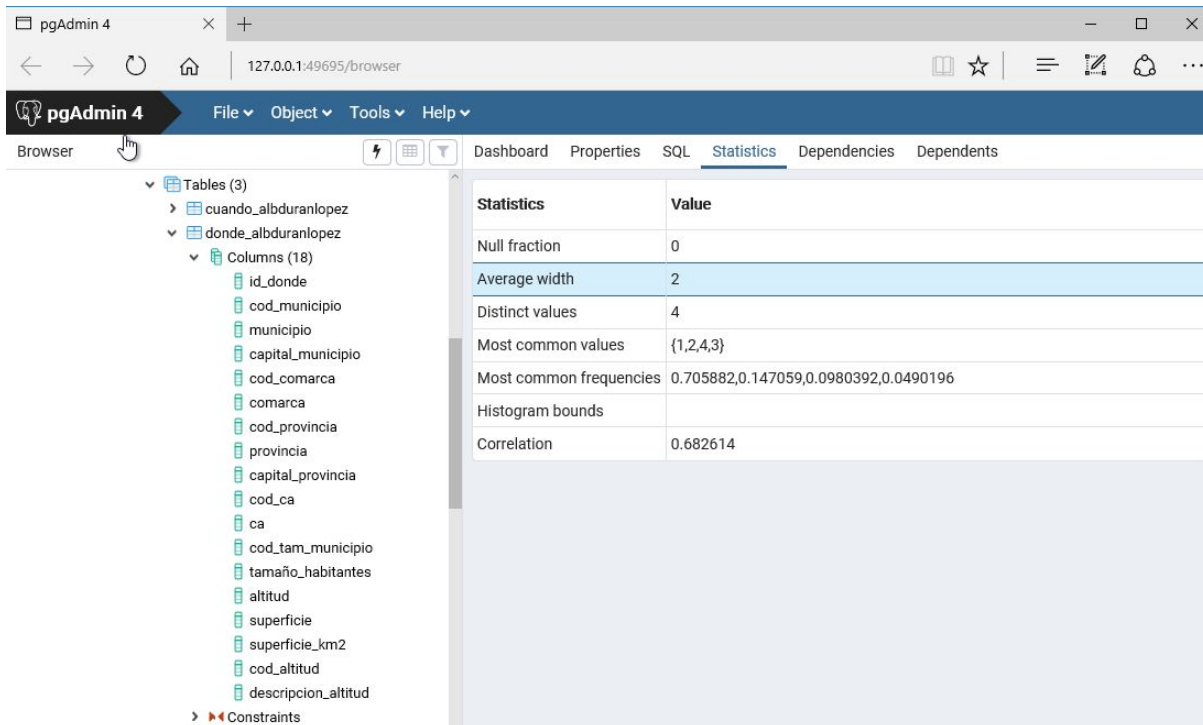
#	New field	ID	Remove ID?	Type	Length	Precision	Format	Group	Decimal	Currency	Nullif	Defau
1	cod_altitud		N	Integer	1							
2	descripcion_altitud		N	String								
3												
4												

Help OK Cancel

Comprobamos los resultados de la transformación en **pgAdmin**:

En la columna de la izquierda se puede ver como el campo **altitud_m** ha desaparecido y se han creado los dos nuevos campos comentados anteriormente.

Para ver que la transformación ha surgido efecto, mostramos *Statistics* por ejemplo de **cod_altitud** donde se refleja que los valores más comunes son enteros que se encuentran en el intervalo [1,4] (como ya habíamos comentado anteriormente)



The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays the database structure. Under 'donde_albduranlopez', the 'Columns (18)' list includes 'id_donde', 'cod_municipio', 'municipio', 'capital_municipio', 'cod_comarca', 'comarca', 'cod_provincia', 'provincia', 'capital_provincia', 'cod_ca', 'ca', 'cod_tam_municipio', 'tamaño_habitantes', 'altitud', 'superficie', 'superficie_km2', 'cod_altitud', and 'descripcion_altitud'. The 'Statistics' tab is selected, showing a table of statistics for the 'cod_altitud' column.

Statistics	Value
Null fraction	0
Average width	2
Distinct values	4
Most common values	{1,2,4,3}
Most common frequencies	0.705882,0.147059,0.0980392,0.0490196
Histogram bounds	
Correlation	0.682614

3.3 - Transformación (3)



Esta vez usamos una transformación en la que calculamos la suma de dos campos con el **paso Calculator**. Se tratará de realizar la suma de los campos **hombres** y **mujeres** y almacenarla en el campo **habitantes**.

Como consecuencia, se borran los dos campos anteriores.

La configuración del **paso** es la siguiente:

Calculator

Step name
suma-padron-albduranlopez

☒ Throw an error on non existing files

Fields:

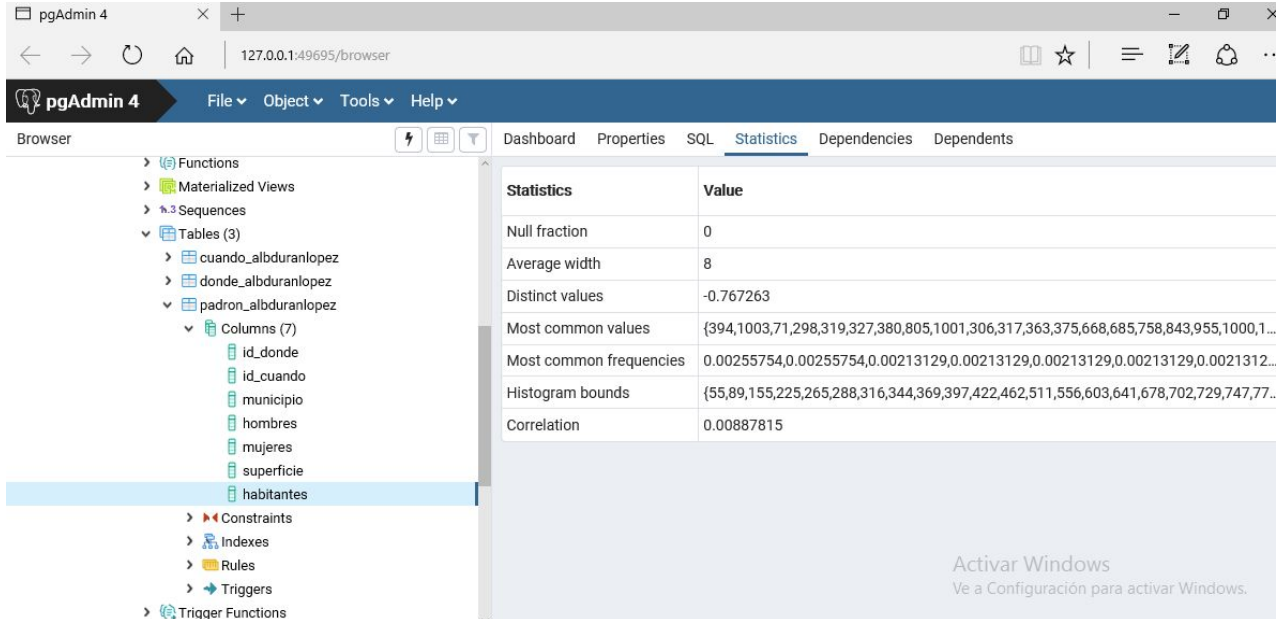
#	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision	Remove	Conversion mask
1	habitantes	A + B	hombres	mujeres		Number			N	

Help OK Cancel

Tras esto, en la tabla output ejecutamos las sentencias de SQL y le damos a **run** para que se realice la transformación

A continuación, nos dirigimos a **pgAdmin** para corroborar que se ha realizado correctamente.

En la siguiente captura mostramos la tabla **padron_albduranlopez** donde como vemos se han borrado los campos **hombres** y **mujeres** para crearse uno llamado **habitantes** donde se almacena la suma de ambos.



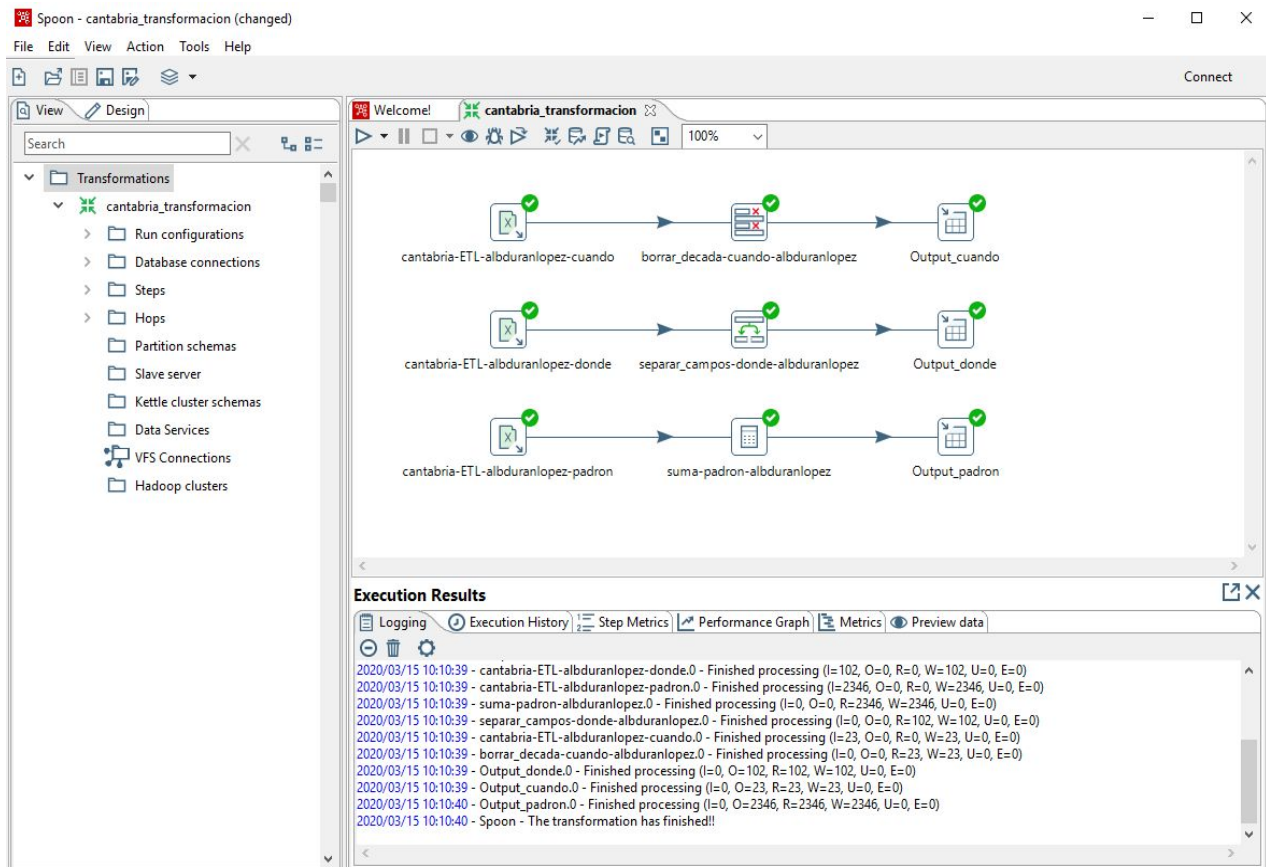
The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure for 'padron_albduranlopez', listing columns: id_donde, id_cuando, municipio, hombres, mujeres, superficie, and habitantes. The right pane shows the 'Statistics' tab for the selected table, displaying a table of statistical data.

Statistics	Value
Null fraction	0
Average width	8
Distinct values	-0.767263
Most common values	{394,1003,71,298,319,327,380,805,1001,306,317,363,375,668,685,758,843,955,1000,1...
Most common frequencies	0.00255754,0.00255754,0.00213129,0.00213129,0.00213129,0.00213129,0.0021312...
Histogram bounds	{55,89,155,225,265,288,316,344,369,397,422,462,511,556,603,641,678,702,729,747,77..
Correlation	0.00887815

Activar Windows
Ve a Configuración para activar Windows.

4 - VISTA GENERAL SPOON

Mostramos el resultado final de las transformaciones realizadas. Las transformaciones se pueden concatenar del mismo modo al realizado en la práctica, es decir, uniendo mediante flechas de entrada y/o salida los diferentes pasos



5 - AUTOMATIZACIÓN MEDIANTE JOBS

En la siguiente sección definiremos un trabajo (<<Jobs>>) para controlar la ejecución de las transformaciones definidas en los pasos anteriores. Nuestro objetivo es el de automatizar dicha ejecución para comprobar que todo funciones correctamente o, por el contrario, se produzcan errores.

En la barra superior (parte izquierda), pulsamos sobre *File > new > Job*

Una vez creado nuestro Job, nos aparecerá a la izquierda una columna donde podemos seleccionar **pasos**, igual que hemos hecho para la transformación.

Seleccionamos los pasos:

- Start**: Paso que inicia la automatización. Se puede configurar para que se ejecute cada cierto tiempo, aunque no es el caso.

- File exists**: Seleccionamos nuestro archivo .xlsx donde se va a realizar las transformaciones. En mi caso **cantabria-ETL-albduranlopez.xlsx**.

- Transformation**: Seleccionamos el archivo obtenido de las transformaciones del punto 4.

Controlar que todo va bien o se producen errores:

- Success**: Si el resultado de la ejecución anterior se muestra satisfactoriamente, dirigimos la salida a este paso

- Write to log**: Si, por el contrario, se producen errores en la ejecución, añadimos este nodo para guardar escribirlo en un **log**.

Spoon - job1

File Edit View Action Tools Help

Connect

View Design

Search

General

- Start
- Dummy
- Job
- Set variables
- Success
- Transformation

Mail

File management

Conditions

Scripting

Bulk loading

Big Data

Modeling

XML

Utility

Repository

File transfer

File encryption

Deprecated

Execution Results

Logging History Job metrics Metrics

2020/03/17 13:57:03 - cantabria_transformacion - Using legacy execution engine

2020/03/17 13:57:03 - cantabria_transformacion - Dispatching started for transformation [cantabria_transformacion]

2020/03/17 13:57:03 - Output_cuando.0 - Connected to database [Cantabria_INE] (commit=1000)

2020/03/17 13:57:03 - Output_donde.0 - Connected to database [Cantabria_INE] (commit=1000)

2020/03/17 13:57:03 - Output_padron.0 - Connected to database [Cantabria_INE] (commit=1000)

2020/03/17 13:57:04 - cantabria-ETL-albduranlopez-donde.0 - Finished processing (I=102, O=0, R=0, W=102, U=0, E=0)

2020/03/17 13:57:04 - separar_campos-donde-albduranlopez.0 - Finished processing (I=0, O=0, R=102, W=102, U=0, E=0)

2020/03/17 13:57:04 - cantabria-ETL-albduranlopez-padron.0 - Finished processing (I=2346, O=0, R=0, W=2346, U=0, E=0)

2020/03/17 13:57:04 - suma-padron-albduranlopez.0 - Finished processing (I=0, O=0, R=2346, W=2346, U=0, E=0)

2020/03/17 13:57:04 - cantabria-ETL-albduranlopez-cuando.0 - Finished processing (I=23, O=0, R=0, W=23, U=0, E=0)

2020/03/17 13:57:04 - borrar_decada-cuando-albduranlopez.0 - Finished processing (I=0, O=0, R=23, W=23, U=0, E=0)

2020/03/17 13:57:04 - Output_donde.0 - Finished processing (I=0, O=102, R=102, W=102, U=0, E=0)

2020/03/17 13:57:04 - Output_cuando.0 - Finished processing (I=0, O=23, R=23, W=23, U=0, E=0)

2020/03/17 13:57:04 - Output_padron.0 - Finished processing (I=0, O=2346, R=2346, W=2346, U=0, E=0)

2020/03/17 13:57:04 - job1 - Starting entry [Success]

2020/03/17 13:57:04 - job1 - Finished job entry [Success] (result=[true])

2020/03/17 13:57:04 - job1 - Finished job entry [Transformation] (result=[true])

2020/03/17 13:57:04 - job1 - Finished job entry [File exists] (result=[true])

2020/03/17 13:57:04 - job1 - Job execution finished

2020/03/17 13:57:04 - Spoon - Job has ended.

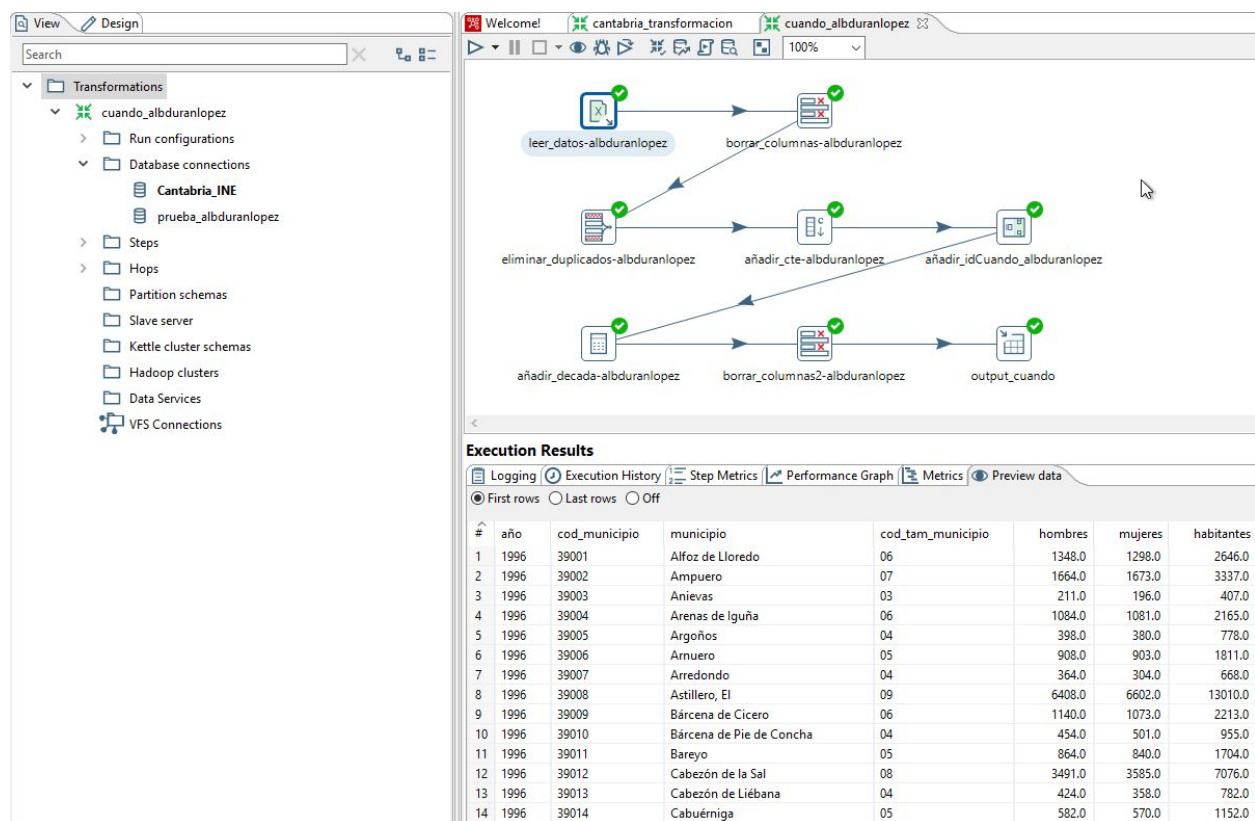
Transformaciones Alternativas

Ejercicio 2

Nos disponemos a hacer el ejercicio 2 de la sección <<Transformaciones alternativas>> del guión. Tenemos que obtener la tabla **cuando** usando como origen la hoja **provincia** obtenida con *Power Query* de la práctica anterior.

Creamos la BD *PostgreSQL*, de nombre **prueba_albduranlopez** y añadimos la tabla **cuando_albduranlopez**.

Las transformaciones usadas para obtener nuestro objetivo son las siguientes:



Pasos añadidos para obtener la transformación (ver imagen anterior):
(denotamos cada paso como: “función que realiza-albduranlopez”)

- El primer paso llamado **leer_datos** sirve para añadir el archivo de trabajo excel que estamos usando como origen.

Si pinchamos sobre él y observamos la terminal inferior podemos ver el contenido del archivo original (<<*Preview data*>>)

- **borrar_columnas**: Borrarnos todas las columnas que no nos interesen, quedándonos solamente con el campo *año*.

- **eliminar_duplicados**: Si observamos el campo *año*, vemos que los años aparecen repetidos, de hecho, el año 1996 se encuentra repetido en la captura anterior (el resto también). Con este paso nos aseguramos que no haya repetidos.

- **añadir_cte**: Añadimos un nuevo campo cuyo valor sea 10 en todas las filas/entradas (más adelante explicaremos su función)

- **añadir_idCuando**: En este paso creamos el nuevo campo *id_cuando* de forma que se incremente de 1 en 1 en cada fila.

- **añadir_decada**: Añadimos el campo *decada*. Es la transformación que más a costado realizar, por ello, se merece una explicación detallada:

Calculator

Step name:

☒ Throw an error on non existing files

Fields:

#	New field	Calculation	Field A	Field B	Field C	Value type	Length
1	año2	Create a copy of field A	año			Integer	
2	decada2	A / B	año2	id		Integer	
3	decada	A * B	decada2	id		Integer	

Help OK Cancel

En el paso previo teníamos el campo *año* y, como bien sabemos, dividiendo este valor por 10 (nos quedamos con la parte entera) y, multiplicando el valor obtenido por 10, obtenemos un valor que se corresponde con la década.

Para realizar esto en **Spoon**, creamos un nuevo campo *año2* que sea una copia exacta del campo *año* (no dejaba trabajar sobre el campo original).

Después, creamos un campo *decada2* donde almacenamos el resultado de la división entera de *año2* y el valor 10, obtenido en el paso creado anteriormente llamado **añadir_cte**. Por último, multiplicamos los campos *decada2* y el valor 10, obteniendo la década.

- **borrar_columnas2**: Borramos todas las columnas adicionales que hemos necesitado crear en el paso anterior.

- **output**: Sirve para generar el contenido de la tabla de la BD

El resultado obtenido se puede ver seleccionando sobre el último nodo y pinchando en <<Preview Data>>

The screenshot displays the Pentaho Data Integration (Spoon) interface. On the left, the 'Transformations' tree shows a project named 'cuando_albduranlopez' with various sub-items like 'Run configurations', 'Database connections', 'Steps', 'Hops', 'Partition schemas', 'Slave server', 'Kettle cluster schemas', 'Hadoop clusters', 'Data Services', and 'VFS Connections'. The main workspace shows a data flow diagram with the following steps:

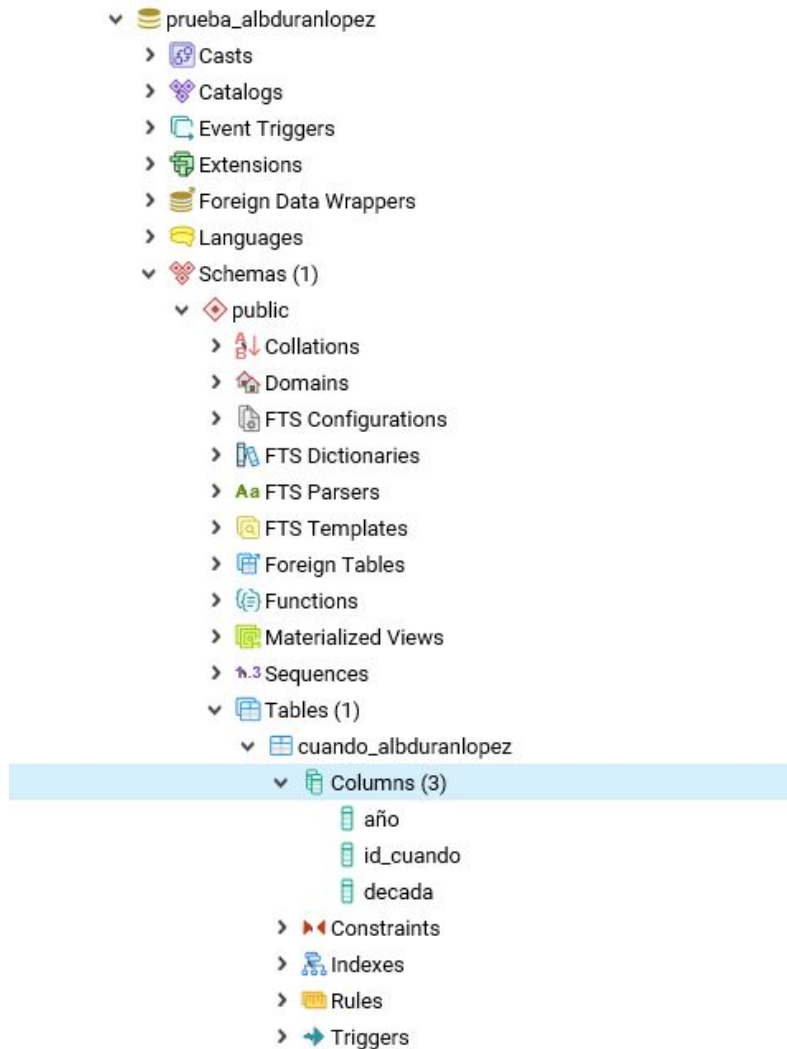
- leer_datos-albduranlopez** (Input) connects to **borrar_columnas-albduranlopez** (Output).
- eliminar_duplicados-albduranlopez** (Filter) connects to **añadir_cte-albduranlopez** (Calculator).
- añadir_idCuando_albduranlopez** (Input) connects to **añadir_cte-albduranlopez**.
- añadir_decada-albduranlopez** (Calculator) connects to **borrar_columnas2-albduranlopez** (Output).
- borrar_columnas2-albduranlopez** connects to **output_cuando** (Output).

Below the diagram, the 'Execution Results' section is visible, showing a table of data:

#	id_cuando	año	decada
1	1	1996	1990
2	2	1998	1990
3	3	1999	1990
4	4	2000	2000
5	5	2001	2000
6	6	2002	2000
7	7	2003	2000
8	8	2004	2000
9	9	2005	2000
10	10	2006	2000
11	11	2007	2000
12	12	2008	2000
13	13	2009	2000
14	14	2010	2010

Comprobamos que en **pgAdmin** se han actualizado las tablas y se ha finalizado el proceso sin errores.

Cabe destacar (aunque no se pide) que este proceso se puede automatizar mediante un *Job*. De hecho, el mismo *Job* realizado en la actividad anterior serviría para controlar la ejecución de las transformaciones realizadas.



Ejercicio libre

Es este último ejercicio realizaremos la transformación 2, pero considerando el archivo original de nuestra provincia **cantabria**.

Nos encontramos en un problema algo difícil ya que no partimos de nuestro archivo generado con *Power Query*, pero nada que no podamos realizar!

Primeramente, necesitamos borrar las primera líneas de cabecera y pie de nuestro archivo original, tal y como hicimos en la práctica 2, por tanto usaremos *Power Query* (sólo para esto)

Configuración de la consulta

PROPIEDADES

Nombre
tabla-2893

Todas las propiedades

PASOS APLICADOS

- Origen
- Navegación
- Filas inferiores quitadas
- Filas superiores quitadas**

Column1	Column2	Column3	Column4
Total	2019	2018	2017
39 Cantabria	581078	580229	580295
39001 Alfoz de Lloredo	2416	2408	2446
39002 Ampuero	4288	4219	4222
39003 Anievas	267	279	291
39004 Arenas de Iguña	1679	1676	1690
39005 Argoños	1748	1723	1713
39006 Arnauero	2103	2108	2098
39007 Arredondo	452	470	472
39008 Astillero, El	18111	18108	18120
39009 Bárcena de Cicero	4246	4186	4132
39010 Bárcena de Pie de Concha	678	680	687
39011 Bareyo	1950	1972	1987
39012 Cabezón de la Sal	8301	8349	8326
39013 Cabezón de Liébana	616	592	595
39014 Cabuérniga	1000	999	1001
39015 Camaleño	936	946	970
39016 Camargo	30260	30263	30556
39027 Campoo de Enmedio	3714	3750	3757

73 COLUMNAS, 105 FILAS

VISTA PREVIA DESCARGADA A LAS 23:36

Esta captura es lo que se pedía en la P2. Sin embargo, necesitamos borrar además las filas 1 y 2 que se muestran, pero con cuidado de recordar en qué columnas están **“Total, columna2”**, **“Hombres, columna26”** y **“Mujer, columna50”** y, recordando que la secuencia de años se repite 3 veces, desde 2019 hasta 1996.

Con el archivo nuevo obtenido, lo importamos en un nuevo **paso** de input archivo Excel en nuestra máquina del aula virtual. Una vez importado el resultado es el siguiente:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

First rows Last rows Off

#	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15	Col
1	39 Cantabria	581078.0	580229.0	580295.0	582206.0	585179.0	588656.0	591888.0	593861.0	593121.0	592250.0	589235.0	582138.0	572824.0	568091.0	56
2	39001 Alfoz de Lloredo	2416.0	2408.0	2446.0	2466.0	2485.0	2499.0	2530.0	2514.0	2484.0	2493.0	2496.0	2502.0	2529.0	2560.0	
3	39002 Ampuero	4288.0	4219.0	4222.0	4181.0	4184.0	4228.0	4235.0	4281.0	4230.0	4183.0	4179.0	4052.0	3951.0	3837.0	
4	39003 Anievas	267.0	279.0	291.0	314.0	326.0	327.0	333.0	351.0	363.0	363.0	366.0	361.0	360.0	376.0	
5	39004 Arenas de Iguña	1679.0	1676.0	1690.0	1711.0	1752.0	1775.0	1811.0	1792.0	1806.0	1814.0	1837.0	1830.0	1867.0	1882.0	
6	39005 Argoños	1748.0	1723.0	1713.0	1720.0	1744.0	1705.0	1711.0	1696.0	1712.0	1663.0	1650.0	1614.0	1535.0	1444.0	
7	39006 Arnuero	2103.0	2108.0	2098.0	2091.0	2125.0	2107.0	2122.0	2115.0	2101.0	2117.0	2122.0	2141.0	2073.0	2026.0	
8	39007 Arredondo	452.0	470.0	472.0	480.0	498.0	507.0	506.0	522.0	515.0	517.0	525.0	532.0	546.0	547.0	
9	39008 Astillero, El	18111.0	18108.0	18120.0	18134.0	18297.0	18282.0	18005.0	17938.0	17675.0	17545.0	17360.0	17065.0	16393.0	16032.0	1
10	39009 Bárcena de Cicero	4246.0	4186.0	4132.0	4124.0	4127.0	4080.0	4078.0	4118.0	4074.0	3988.0	3784.0	3613.0	3407.0	3139.0	
11	39010 Bárcena de Pie de Concha	678.0	680.0	687.0	710.0	728.0	735.0	739.0	758.0	770.0	788.0	805.0	787.0	794.0	800.0	
12	39011 Bareyo	1950.0	1972.0	1987.0	1999.0	2011.0	2016.0	2021.0	2060.0	2075.0	2053.0	2060.0	2003.0	1928.0	1895.0	
13	39012 Cabezón de la Sal	8301.0	8349.0	8326.0	8345.0	8353.0	8369.0	8350.0	8234.0	8303.0	8322.0	8372.0	8234.0	8086.0	8032.0	
14	39013 Cabezón de Liébana	616.0	592.0	595.0	601.0	622.0	628.0	650.0	707.0	699.0	690.0	699.0	685.0	688.0	686.0	
15	39014 Cabezón de Toranzo	1000.0	999.0	1001.0	1011.0	998.0	1016.0	1046.0	1066.0	1077.0	1086.0	1093.0	1093.0	1093.0	1096.0	

Nos preparamos para normalizar nuestra tabla con el **paso** <<Row Normaliser>> cuya configuración mostramos a continuación:

Row normaliser

Step name:

Type field:

Fields

#	Fieldname	Type	new field
1	Column2	2019	Total
2	Column3	2018	Total
3	Column4	2017	Total
4	Column5	2016	Total
5	Column6	2015	Total
6	Column7	2014	Total
7	Column8	2013	Total
8	Column9	2012	Total
9	Column10	2011	Total
10	Column11	2010	Total
11	Column12	2009	Total
12	Column13	2008	Total
13	Column14	2007	Total
14	Column15	2006	Total
15	Column16	2005	Total
16	Column17	2004	Total
17	Column18	2003	Total
18	Column19	2002	Total
19	Column20	2001	Total
20	Column21	2000	Total
21	Column22	1999	Total
22	Column23	1998	Total
23	Column25	1996	Total
24	Column26	2019	Hombres
25	Column27	2018	Hombres
26	Column28	2017	Hombres
27	Column29	2016	Hombres
28	Column30	2015	Hombres
29	Column31	2014	Hombres
30	Column32	2013	Hombres
31	Column33	2012	Hombres
32	Column34	2011	Hombres
33	Column35	2010	Hombres
34	Column36	2009	Hombres
35	Column37	2008	Hombres
36	Column38	2007	Hombres

Help OK Cancel Get Fields

Para la configuración de este paso, añadimos las columnas que queremos crear en el campo <<*new Field*>>, en nuestro caso son: **Total, Hombres y Mujeres**.

Por otro lado, añadimos en el campo <<*Type*>> la secuencia de los años (2019-1996)

Cabe destacar que se han tenido que borrar la columna1 y las columnas referentes al año 1997 ya que ese año el *INE* no realizó un muestreo y, como consecuencia, el campo estaba vacío, reflejándose en un error. Mostramos resultados tras este paso:

Execution Results

Logging Execution History Step Metrics Performance Graph Metrics Preview data

☒ First rows ☐ Last rows ☐ Off

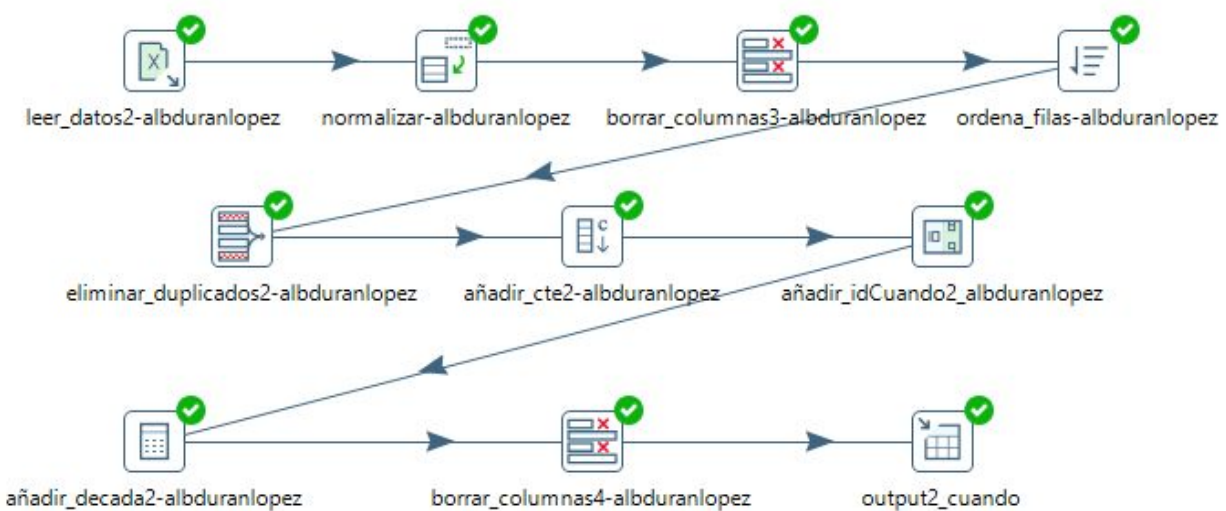
#	Column1	Type	Total	Hombres	Mujeres
1	39 Cantabria	2019	581078.0	281801.0	299277.0
2	39 Cantabria	2018	580229.0	281564.0	298665.0
3	39 Cantabria	2017	580295.0	281808.0	298487.0
4	39 Cantabria	2016	582206.0	282988.0	299218.0
5	39 Cantabria	2015	585179.0	284788.0	300391.0
6	39 Cantabria	2014	588656.0	286782.0	301874.0
7	39 Cantabria	2013	591888.0	288643.0	303245.0
8	39 Cantabria	2012	593861.0	289999.0	303862.0
9	39 Cantabria	2011	593121.0	289872.0	303249.0
10	39 Cantabria	2010	592250.0	289931.0	302319.0
11	39 Cantabria	2009	589235.0	288735.0	300500.0
12	39 Cantabria	2008	582138.0	285469.0	296669.0
13	39 Cantabria	2007	572824.0	280283.0	292541.0
14	39 Cantabria	2006	568091.0	277869.0	290222.0
15	39 Cantabria	2005	562309.0	274797.0	287512.0

En total hay 73 columnas y el proceso para configurar el nodo de normalización es algo tedioso pero, como es lógico, tanto **Total, Hombres y Mujeres** siguen la misma estructura.

Recordemos el objetivo de este ejercicio, el cual era el de obtener la tabla **cuando_albduranlopez** a partir del archivo original.

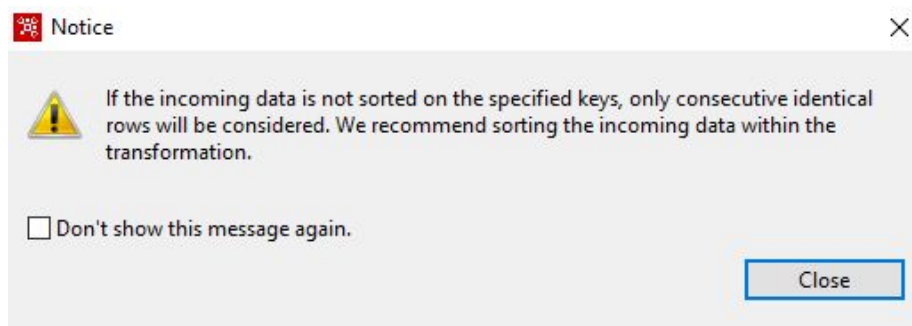
Hemos recortado cabeceras y pie de hoja, normalizado y mostrado los datos. Echando un vistado a la tabla de arriba vemos que tenemos el campo <<Type>> que muestra los años que justamente son necesarios para la tabla que debemos obtener, **cuando**.

Por ello, simplemente necesitamos borrar el resto de campos, renombrar el campo <<Type>> al nombre <<Año>>, ordenar los años, borrar duplicados y añadir los campos <<id_cuando>> y <<decada>>, igual que se hizo en el apartado anterior:



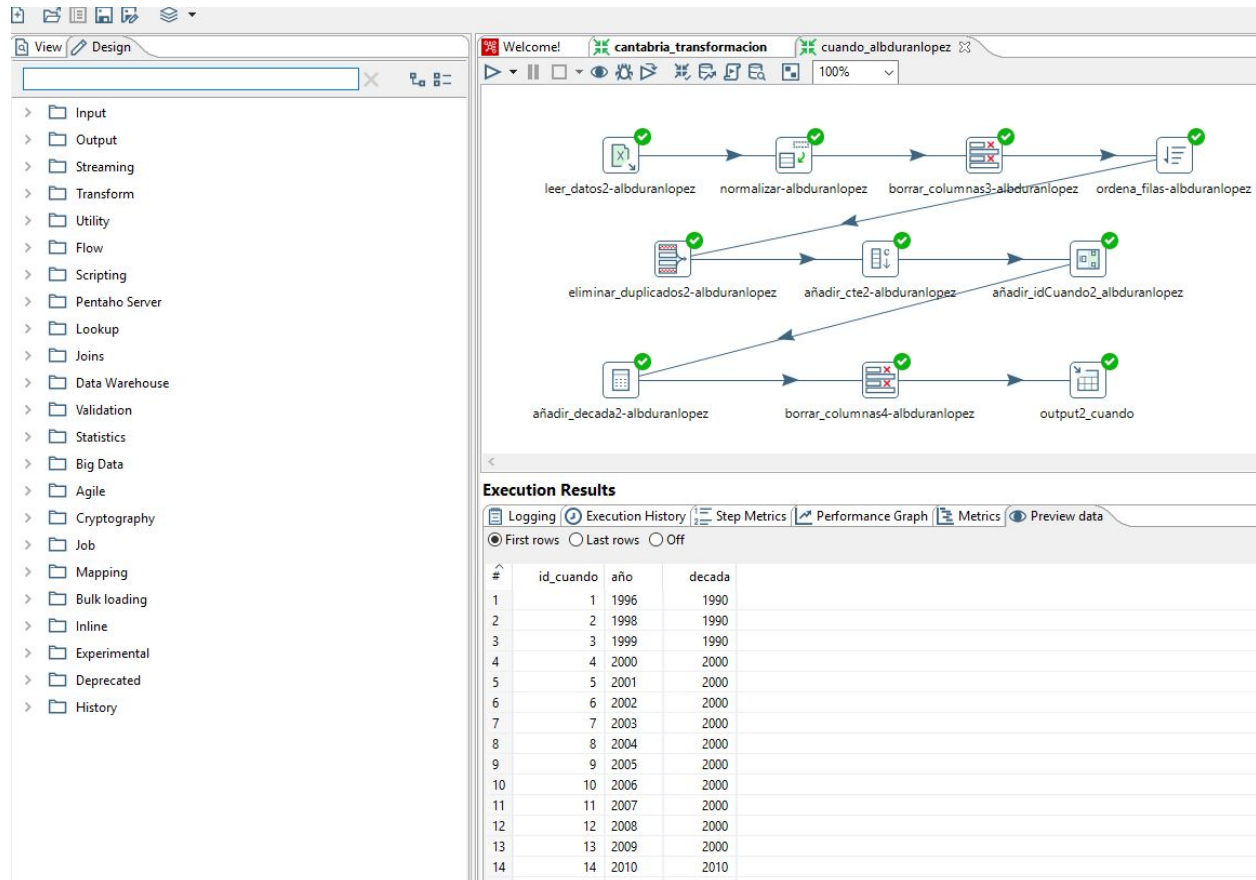
Siguiendo el nombre de cada paso, el proceso seguido es muy visual.

Una vez obtenida la tabla normalizada, y tal y como hemos comentado anteriormente, borramos las columnas que no queramos, las ordenamos (importante ya que el <<elimina duplicados>> sólo funciona con filas consecutivas)



Luego, eliminamos repetidos, y añadimos el campo `<<id_cuando>>` con un paso que añada una columna que actúa como un contador, desde 1 hasta el número de filas que tengamos.

Por último, para obtener el campo `<<decada>>`, dividimos el año entre 10 (nos quedamos con la parte entera) y después, multiplicamos el resultado por 10.



Como resultado, obtenemos la tabla **cuando_albduranlopez**, que podemos comprobar que está bien creada desde **pgAdmin**. Sin más dilación y, al igual que en el ejercicio anterior, destacamos que este proceso se puede automatizar mediante un `<<Job>>` y así controlar la ejecución de las transformaciones realizadas.

REFERENCIAS

- Guión de la prácticas 3
- Documentación oficial Spoon
- [StackOverflow](#)
- [Pentaho Documentation](#)
- [Pentaho](#)