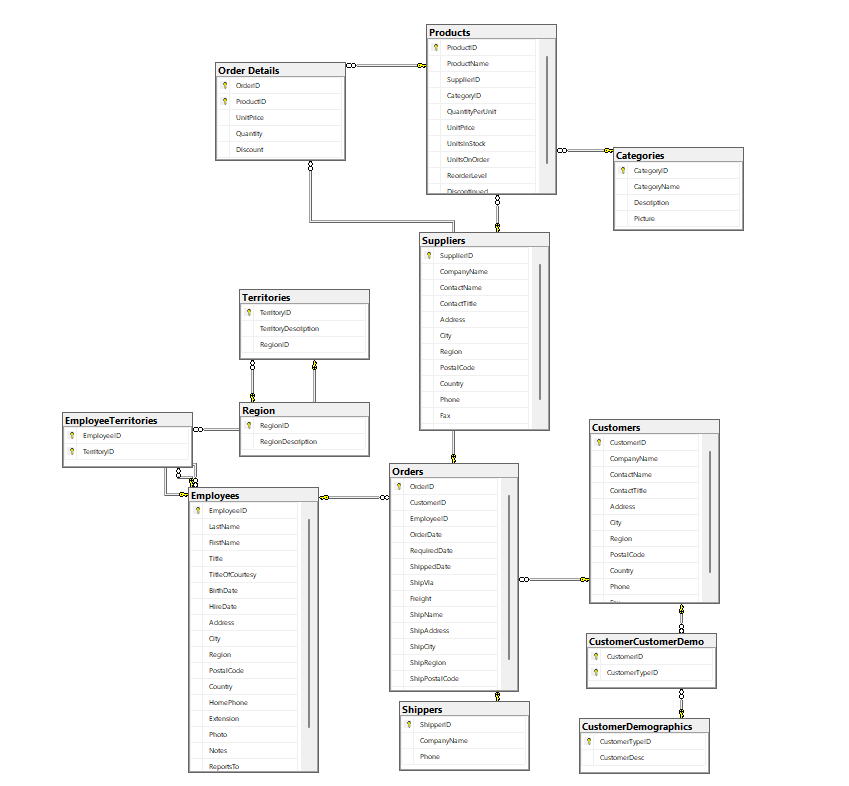
**Andres Esteban Muñoz Medina – 220034023**

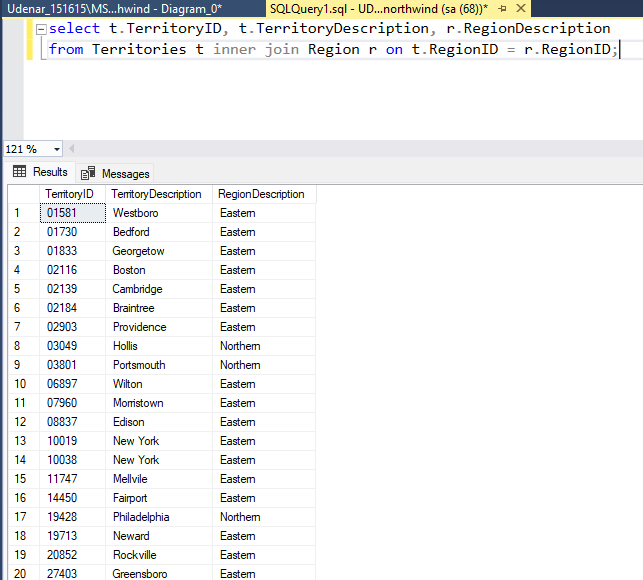
* Las relaciones entre las tablas se establecen mediante llaves foranes, el diagrama muestra de mejor manera la forma en como están relacionadas las tablas.



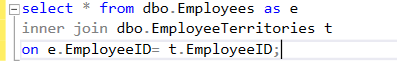
Podemos ver que todas las tablas están relacionadas entre sí por ejemplo Employess con Territories atraves de la tabla EmployeeTerritories donde están como llave primaria las 2 llaves de estas tablas siendo foráneas.

Ejemplo la tabla OrderDetails tiene como llave primaria una columna de orderId y además la llave foránea que provienen de las tablas Products para hacer la relación.

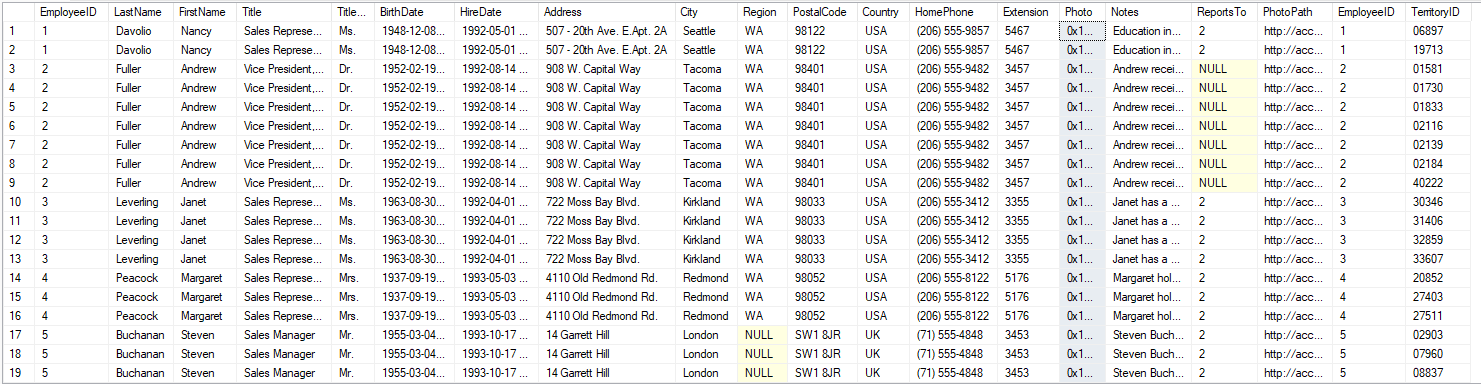
* Con la consulta detallada se muestran los siguientes resultados:



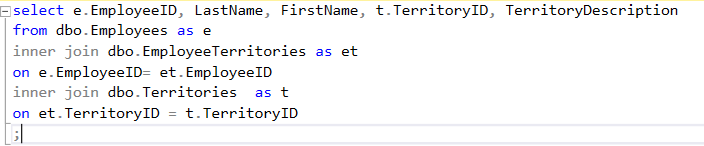
* Una consulta SQL que permita obtener todos los campos de empleado (select \*) y el ID del territorio se hace uso de inner join entre las tablas Employees y EmployeeTerritories:

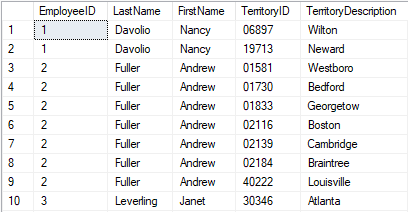


Esta consulta arroja todos los campos:



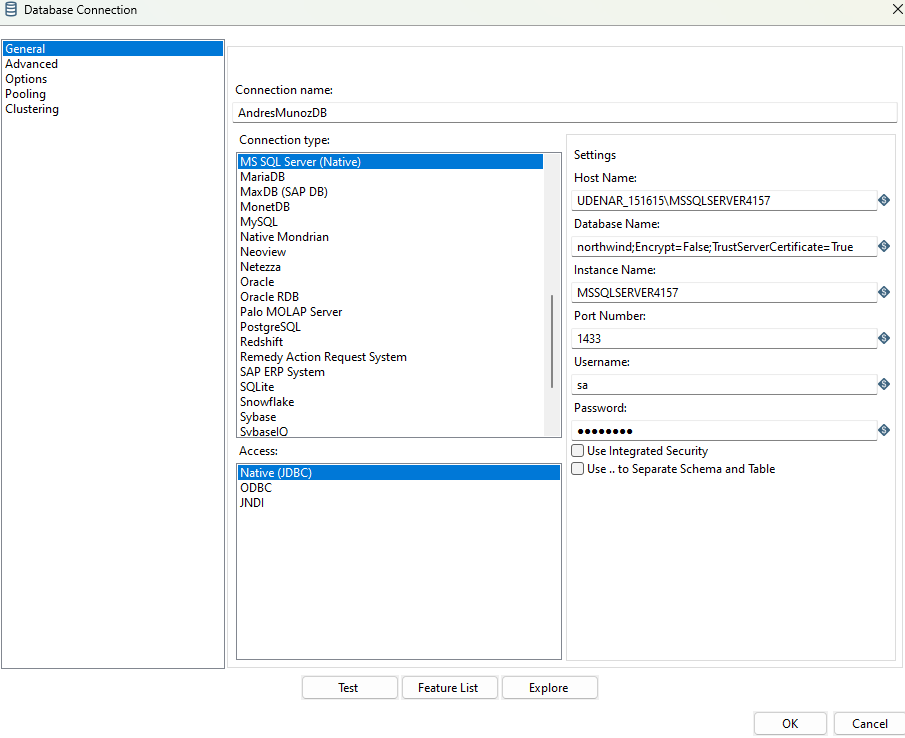
* Completando la query anterior para además mostrar la descripción del territorio se agrega la tabla Territories, limite la consulta a 5 columnas para evitar que muestre todas las columnas:





**Parte 2**

La configuración de la conexión de la base de datos en Pentaho quedo asi:



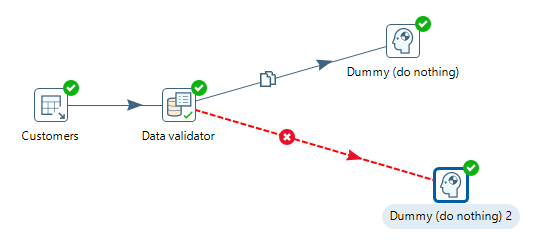
Cargamos las cuatros tablas con Table input seleccionando la base de datos AndresMunozDB.

Interfaz de usuario gráfica, Aplicación

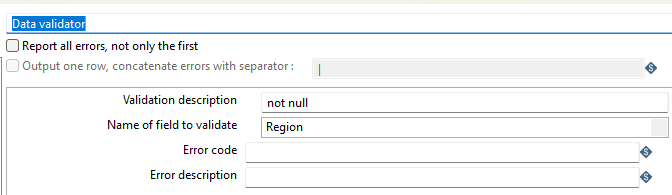
Descripción generada automáticamente

* Aplica un flujo que permita eliminar los registros que contienen datos null, visualízalos por medio de nodos Dummy.

Utilizamos el componente de data validator conectado a dos Dummy para visualizar las dos tablas:



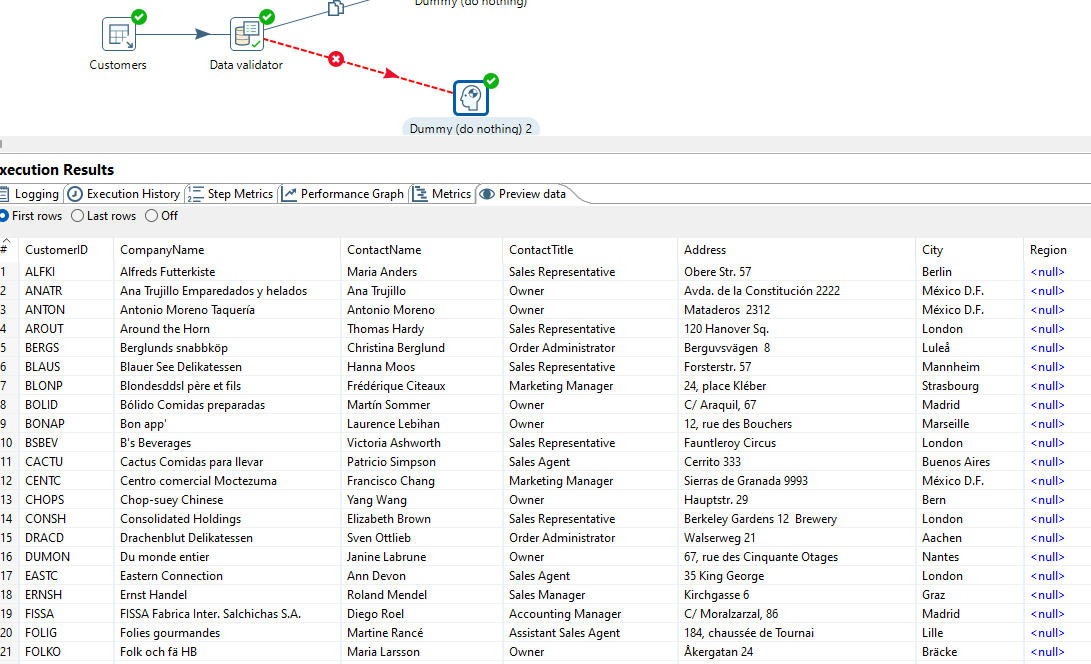
Validamos que en la columna Región de la tabla Customers no hayan null:





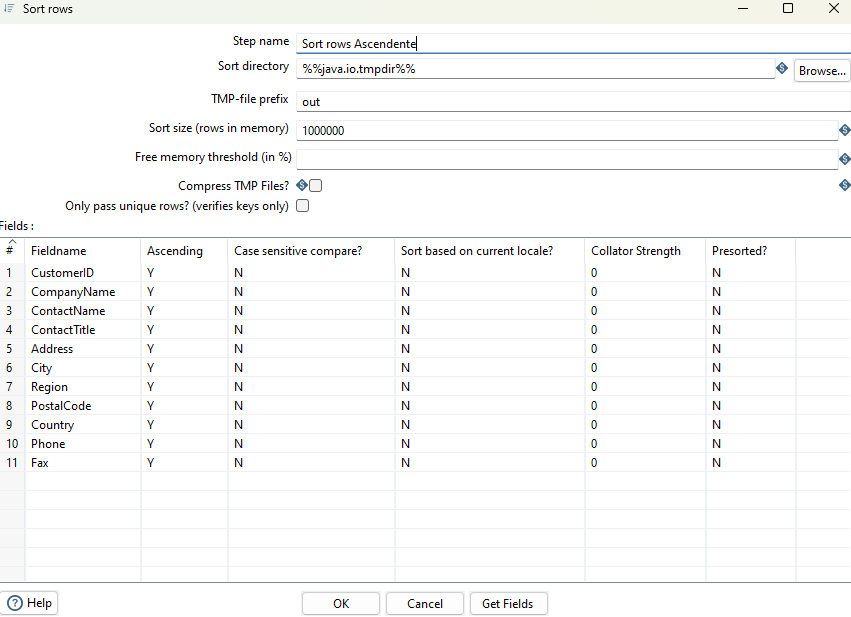


Los datos null salen en el summy inferior:

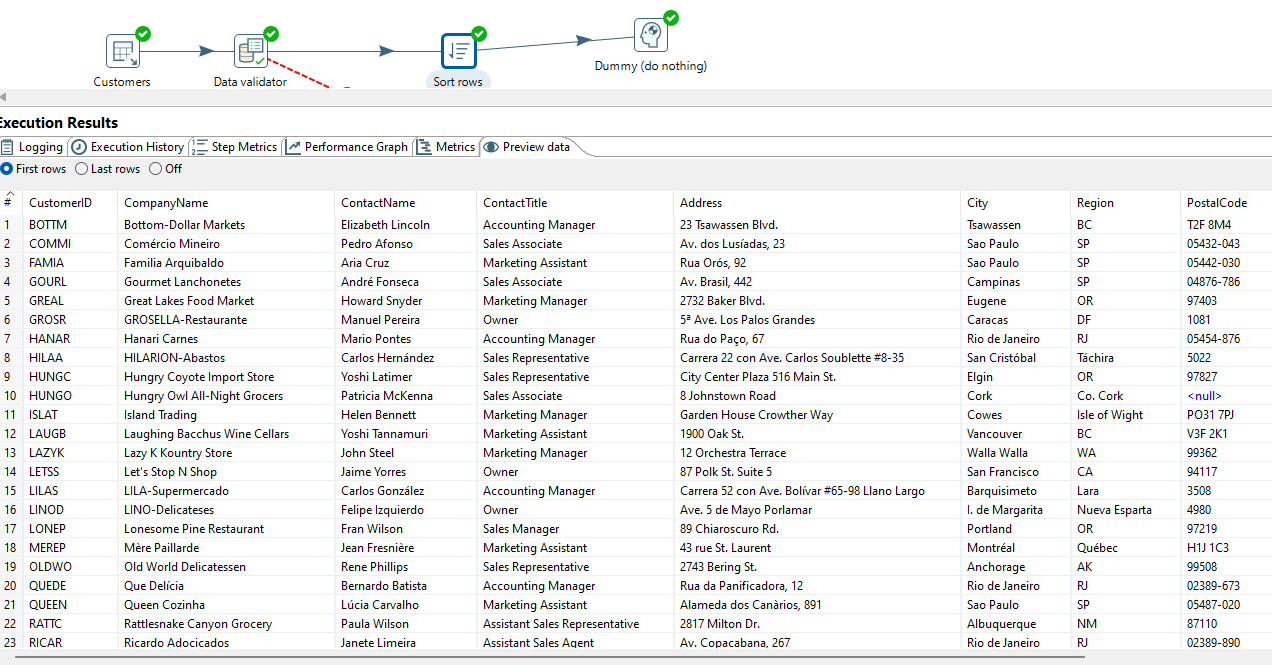


* Organiza los Rows en forma ascendente.

Utilizamos el componente Sort Rows para ordenar los datos en forma ascendente:

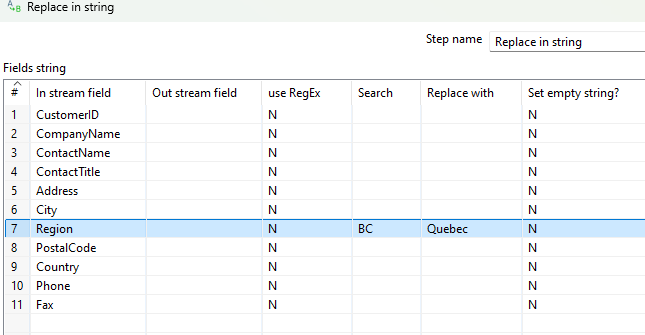


Ordena primero la columna CustomerID, CompanyName y así sucesivamente:

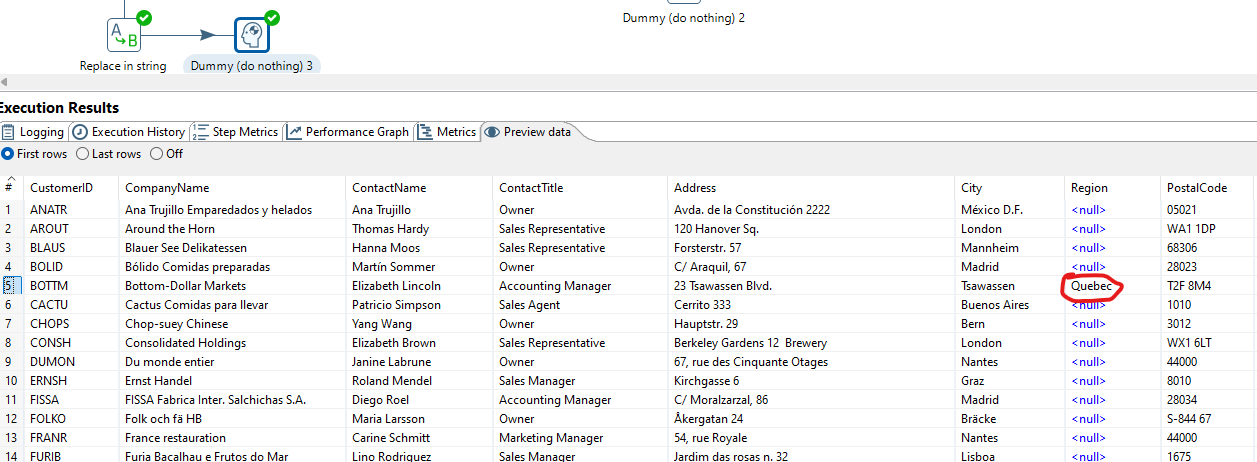


* Realice una operación de remplazar en cadena para los registros que quedaron del paso anterior, remplazando las regiones por Quebec.

Utilizamos el componente de transformación Replace in string para buscar en la columna de Region “BC” y lo reemplace con “Quebec”.

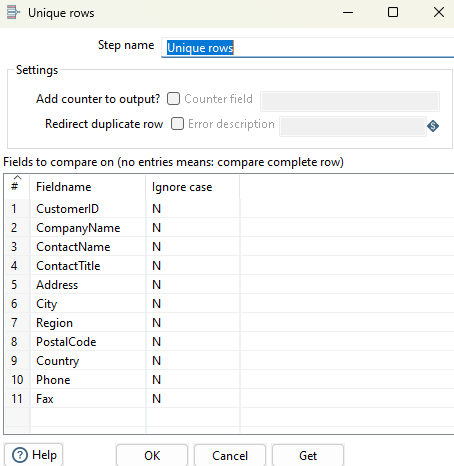


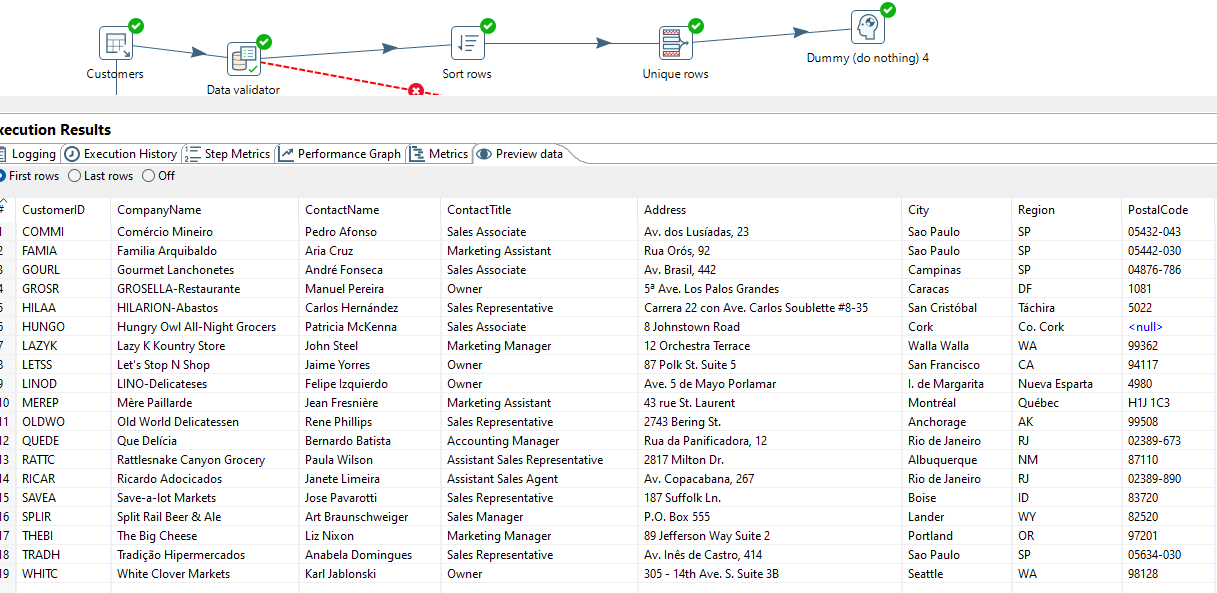
Probamos si se realizo el cambio de BC por Quebec:



* Aplique una validación que permita eliminar valores duplicados.

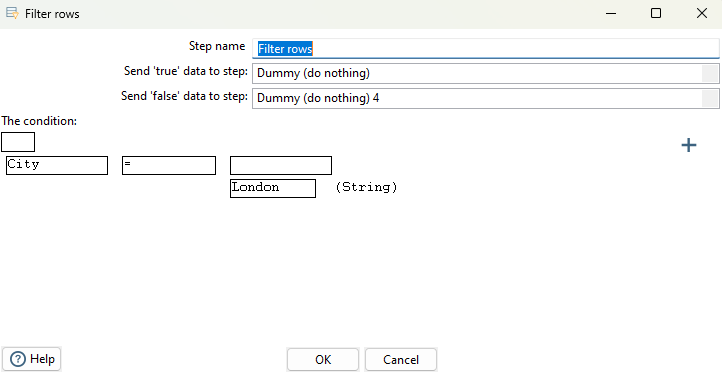
Utilizamos el componente de Unique rows para eliminar los duplicados y es necesario conectarlo después de un Sort Rows ya que es obligatorio ordenar los registro antes:



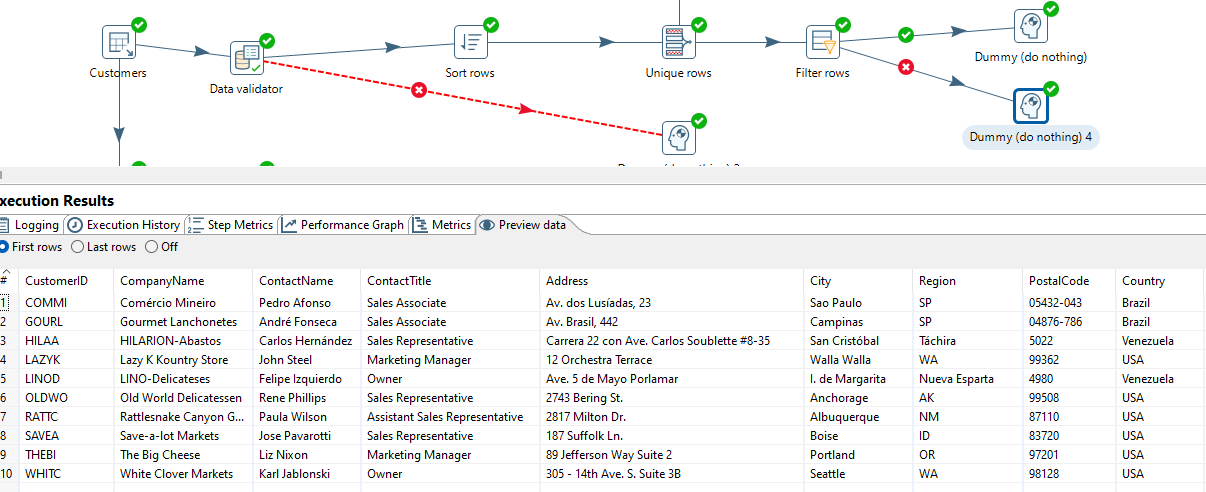


* Filtra los registros que sean diferentes de City London.

Usamos el componente Filter rows donde en la condición seleccionamos la columna de City evaluar los que sean igual a London:

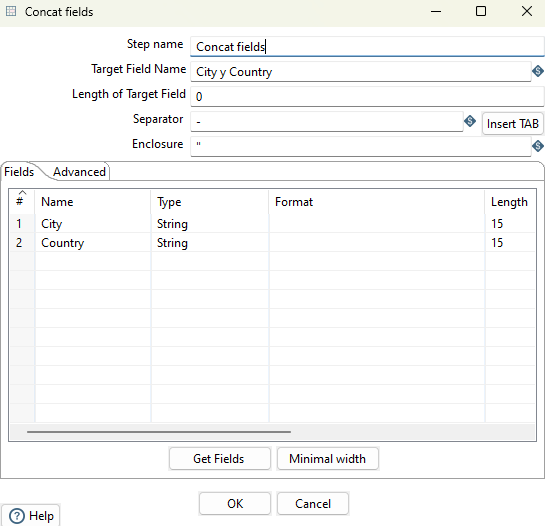


Miramos la salida False y en la vista de la data se ve que en city no aparece London

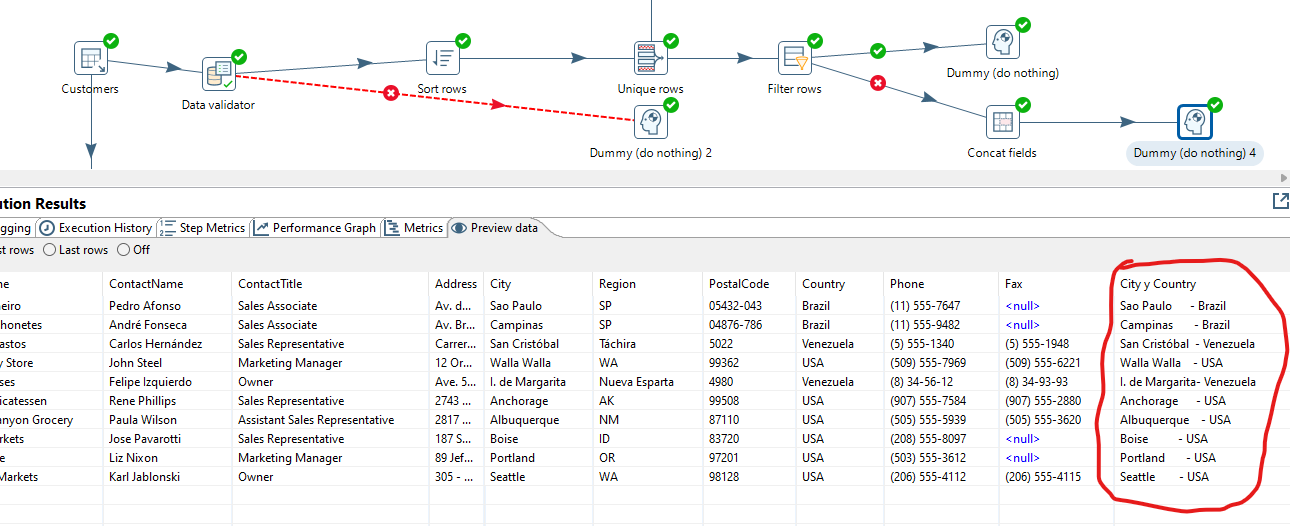


* Como funcionaria la operación concat fields, genera un nuevo campo que concatene los campos City y country separado por un guion “-”.

Con Concat fields agregamos los Fields (columnas) que queremos concatenar y en Separator ponemos el guion “-“.

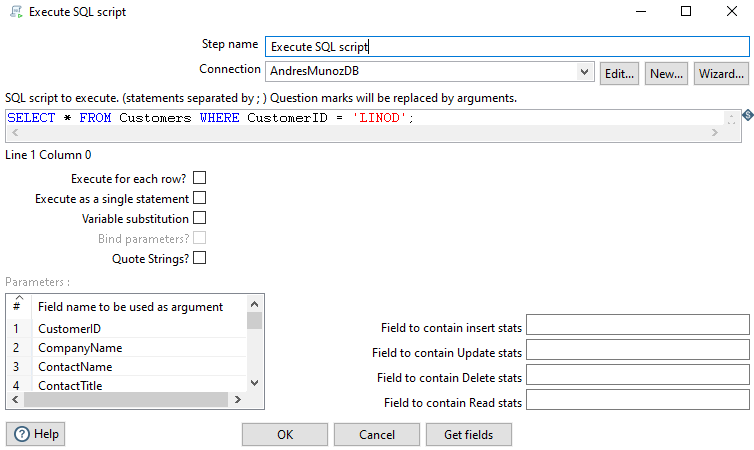


Visualizamos la tabla y se muestra la tabla concatenada City y Country como ultima columna:

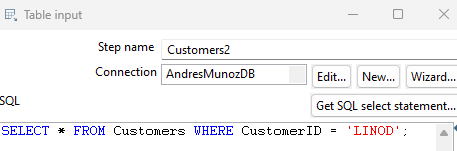


* Cómo se agregaría un nodo de Execute SQL script para ejecutar un select que filtre por el CustormerID.

Utilizamos la característica de Execute SQL para realizar consultas en sql:



En este caso la caracteristica Execute sql no realiza la consulta necesaria por lo tanto usaremos la caracteristica table input agregando la consulta sql que queremos realizar en la base de datos:



Nos realiza la consulta con la condición de buscar en CustomerID el dato que sea igual a ‘LINOD’

