ETL design training session – data and process quality

This document serves as a training session for using Pentaho Data Integration (PDI) tool (a.k.a. Kettle) for improving quality of data and quality of the ETL process.

Content:

a. In the third part, we will use the ETL design created in the previous training session to showcase in Pentaho Data Integration tool the process of improving the ETL data and process quality characteristics, based on different specified quality goals.

Data sources used in the training: All examples are created over the Learn-SQL system. For better understanding of internals of the system under the study, the schematic/diagrammatic representations of the domain ontology and the available data sources are available (see Sources/Explanations.pdf):

1) Diagrammatic/schematic representation of the Learn-SQL system
2) Moodle DB schema (IE notation), which captures the Learn-SQL database schema
   a. Follow the instructions in 'Sources/DB Moodle/ DBConnectionSetup.pdf' to set up the database connection and deploy the LearnSQL database.
3) Candidate's information in XML format, which is provided by Moodle
4) Evaluation results (candidates with final marks) in excel (dni, mark), which is the Excel sent to FIB at the end of the course

ETL design: training session – data and process quality

Part A: Enhancing data and process quality

Using the ETL process that we designed in the previous part, we will showcase how we can improve quality dimensions of the process, while maintaining the same functionality. Examples of such quality dimensions are performance, data quality, reliability etc.

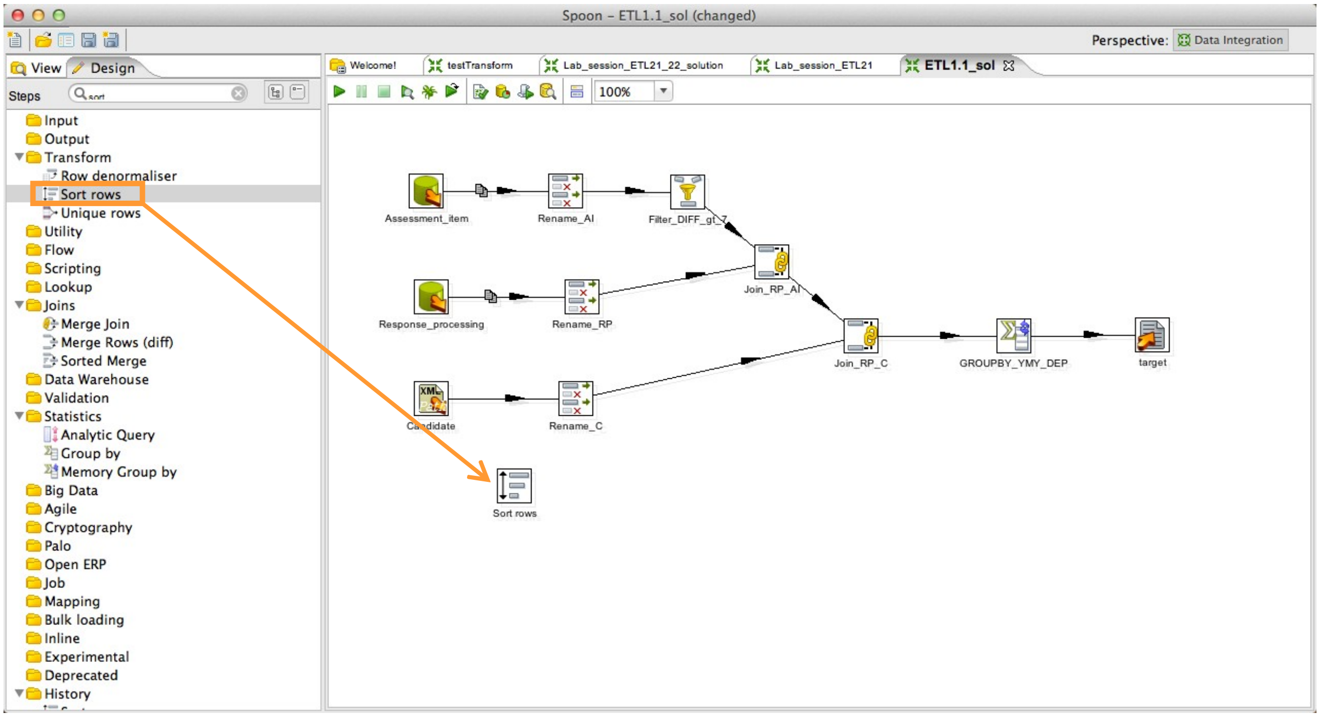
1. Improving Data Quality:

Data coming from the sources can be "dirty", as they can contain duplicates, incomplete or inconsistent entries etc. Dealing with those aspects of data quality can improve *data consistency* and *data completeness*. It is important to "cleanse" data as early as possible in the ETL process so that "dirty" data and their side-effects are not carried along. To achieve that using Pentaho Data Integration tool, we can follow the steps as described bellow.

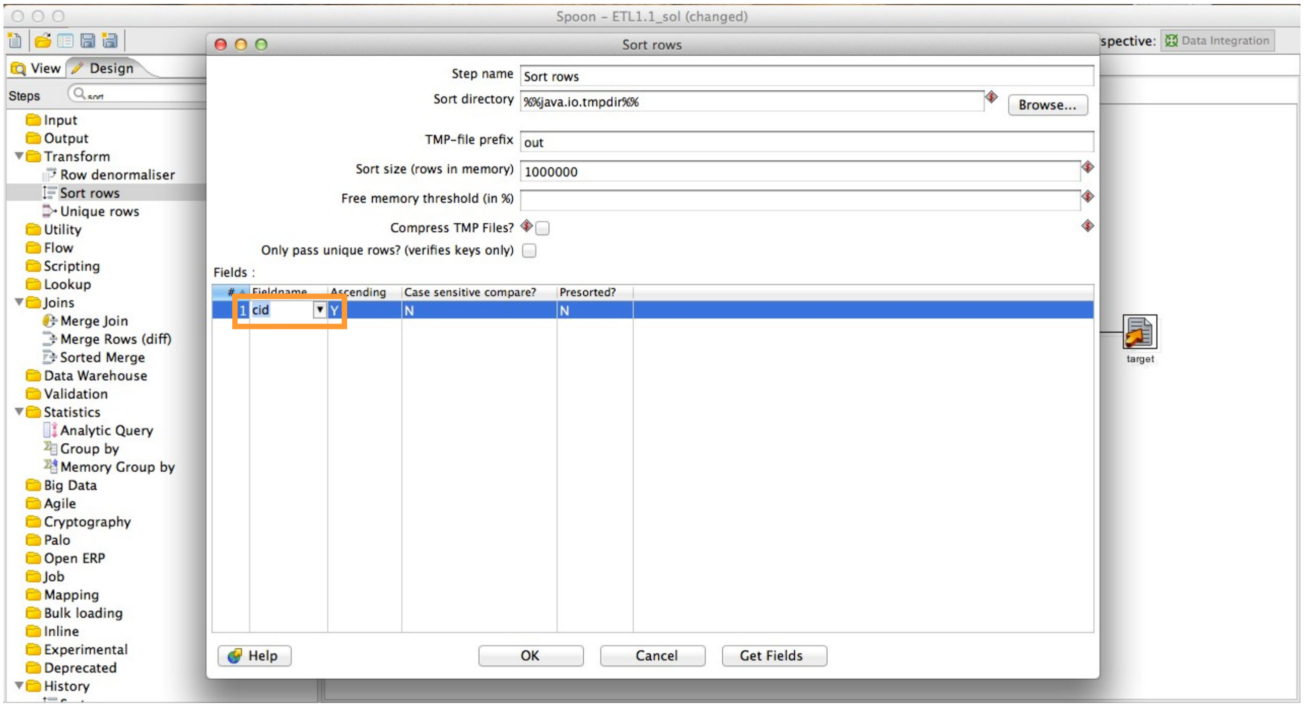
a. Removing Duplicates:

Duplicates are duplicate representations (i.e., repeated copies) of the same entity. For example, the same candidate (mdl_user) might be entered more than once as a **row** entry in the mdl_user.xml file. In Pentaho Data Integration tool we can use the "Unique rows" template to remove such duplicates. This template requires the input to be sorted on the specified keys that are compared for record matching.

(1) From the "Transform" category of the design palette choose "Sort rows" template and drag and drop it to the Spoon canvas as shown in the figure



(2) We need to parameterize the sort rows step considering the keys of the input. See figure below.



Step name: Sort rows

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

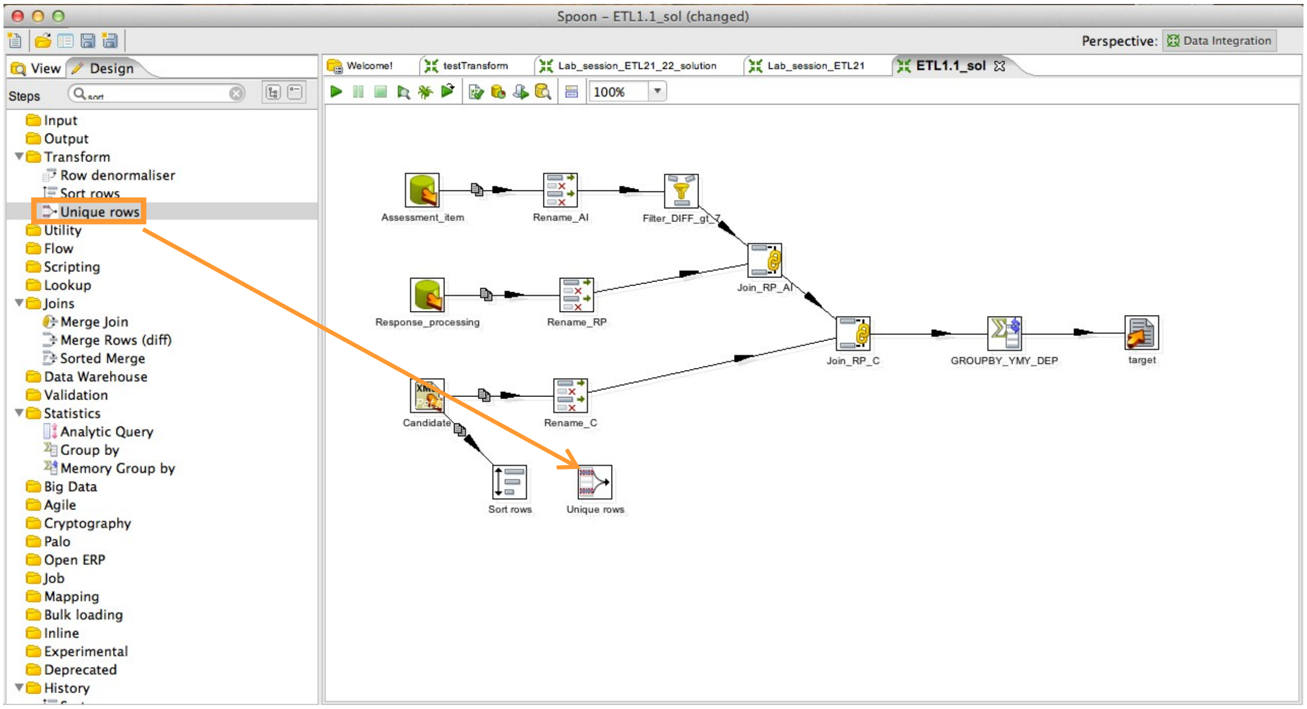
Compress TMP Files? □

Only pass unique rows? (verifies keys only) □

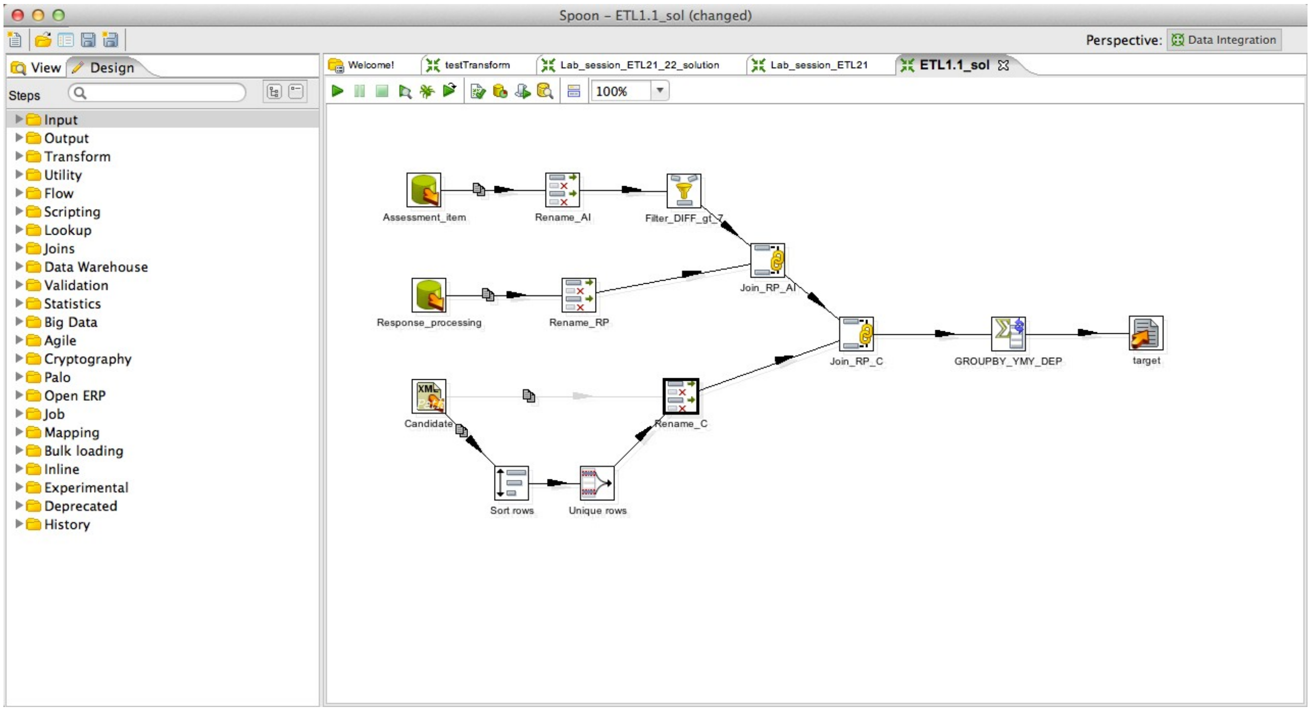
Fields :

#	Fieldname	Ascending	Case sensitive compare?	Presorted?
1	cid	Y	N	N

(3) From the "Transform" category of the design palette choose "Unique rows" template and drag and drop it to the Spoon canvas as shown in the figure.



(4) Remove the hop coming from the candidate xml file input and replace it with a new hop coming from unique rows, as shown in the figure.

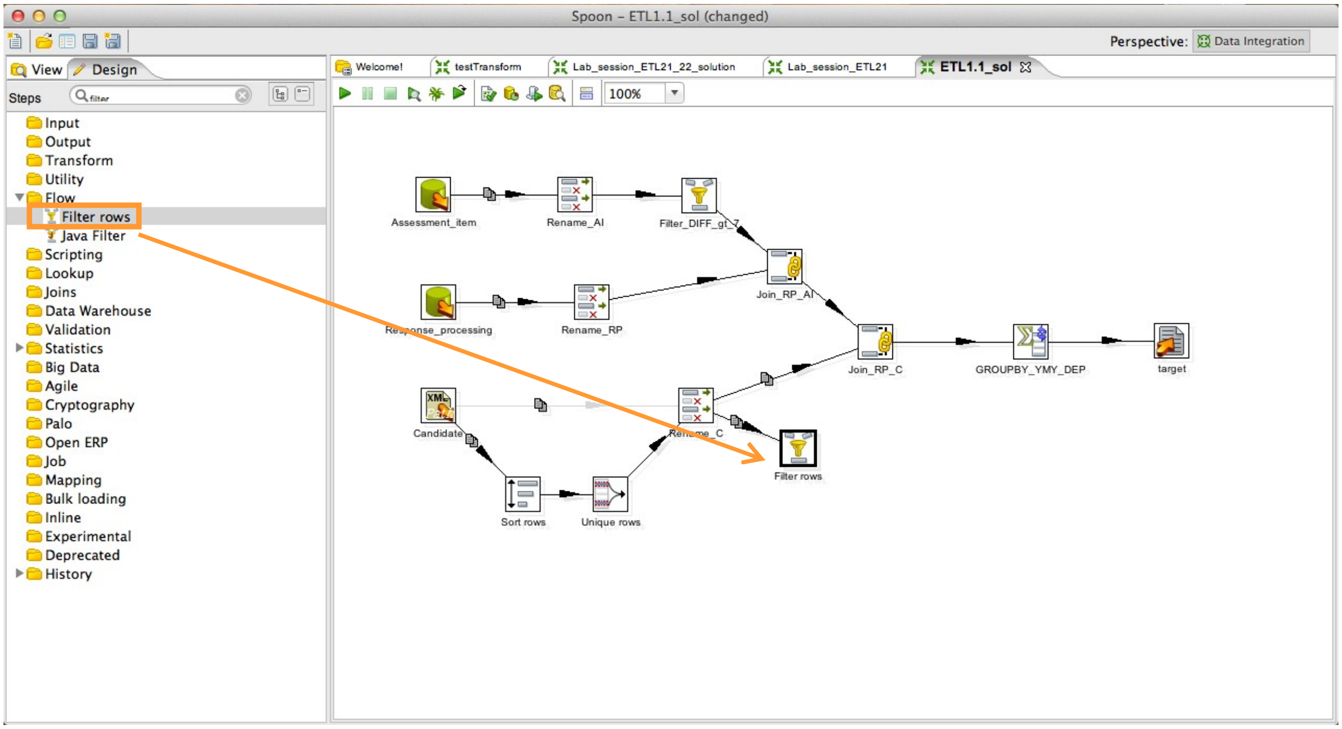


Note: This step will only eliminate identical entries, i.e., entries with the same values for all fields. In case there are entries for the same entity but with different values, more advanced entity resolution mechanisms have to be applied.

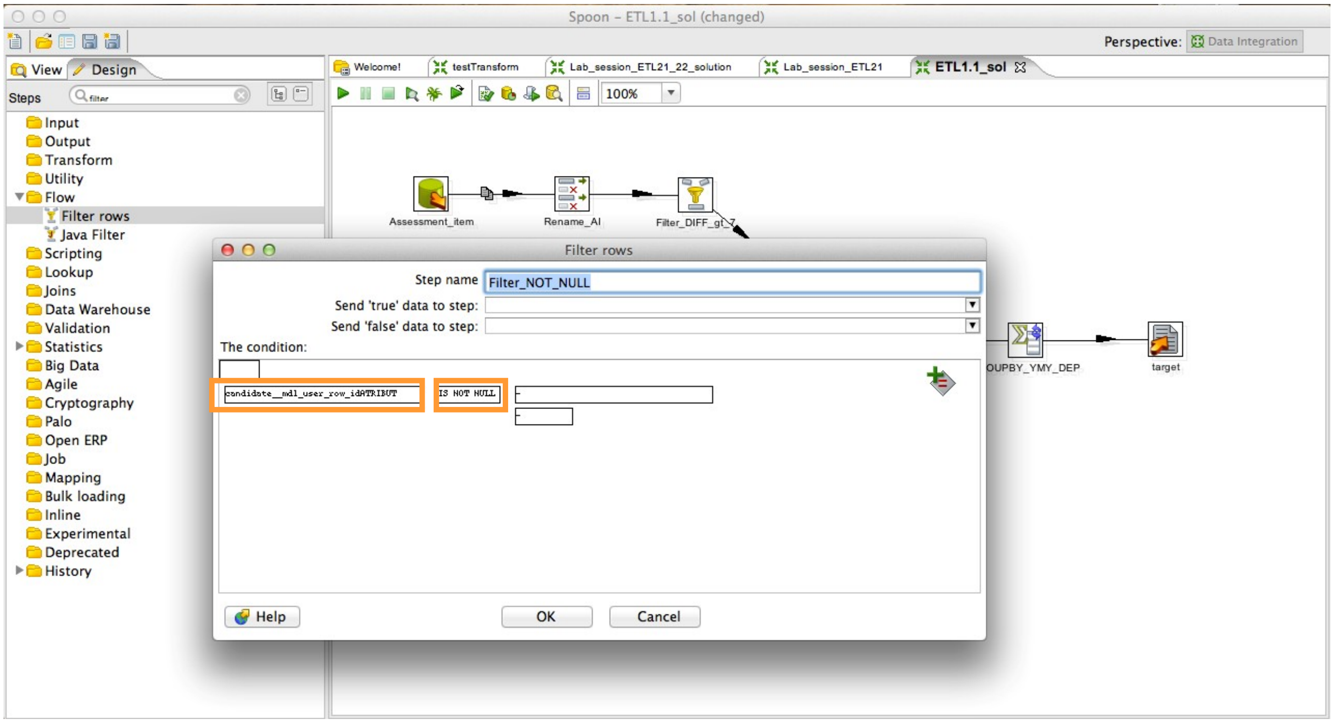
b. Removing incomplete records:

Incomplete records are entries with incomplete or missing data, due to poor maintenance, processing errors or other causes. For example, in the mdl_user.xml file there could be row entries with missing important information such as the candidate id, making these entries useless. In Pentaho Data Integration tool we can use the “Filter” template to remove such incomplete entries.

(1) From the “Flow” category of the design palette choose “Filter rows” template and drag and drop it to the Spoon canvas as shown in the figure.



(2) We need to parameterize the filter rows step by configuring the fields to be considered. See figure below.



Step name: Filter_NOT_NULL

Send 'true' data to step:

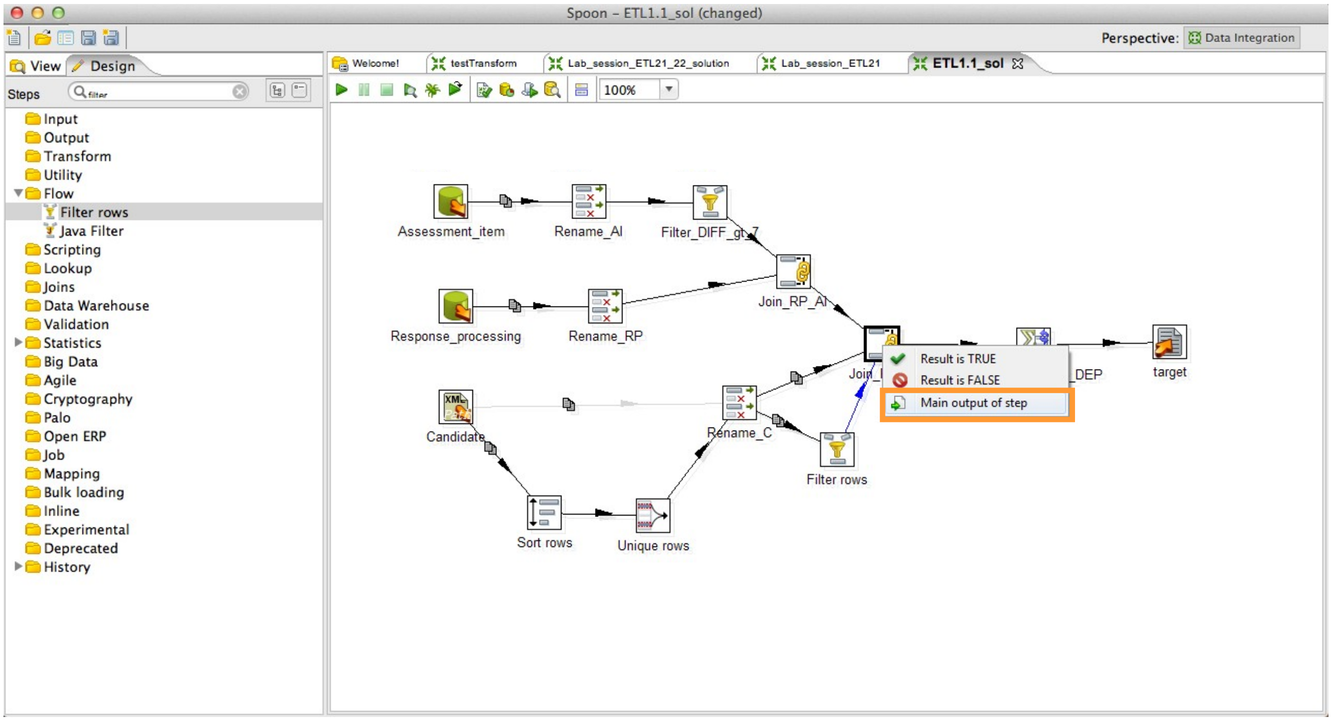
Send 'false' data to step:

The condition:

candidate__mdl_user_row_idATRIBUT IS NOT NULL

(3) Remove the hop coming from the Rename_C step and replace it with a new hop coming from the filter, selecting the "Main output of step" option, as shown in the figure.

Important note: *filtering steps like "Filter rows" and "Java Filter" also allows controlling the rejected rows of the step (e.g., in data cleaning operations to investigate further on the erroneous rows). In that case, instead of the "Main output of the step", we will provide two outputs of the filtering step, (1) "Result is TRUE" to send correct (clean) data to the further step for data processing, and (2) "Result is FALSE" to store erroneous data for later further investigation. Notice that in the case you select "Result is TRUE" output hop, you must obligatory provide the output hop for "Result is FALSE".*

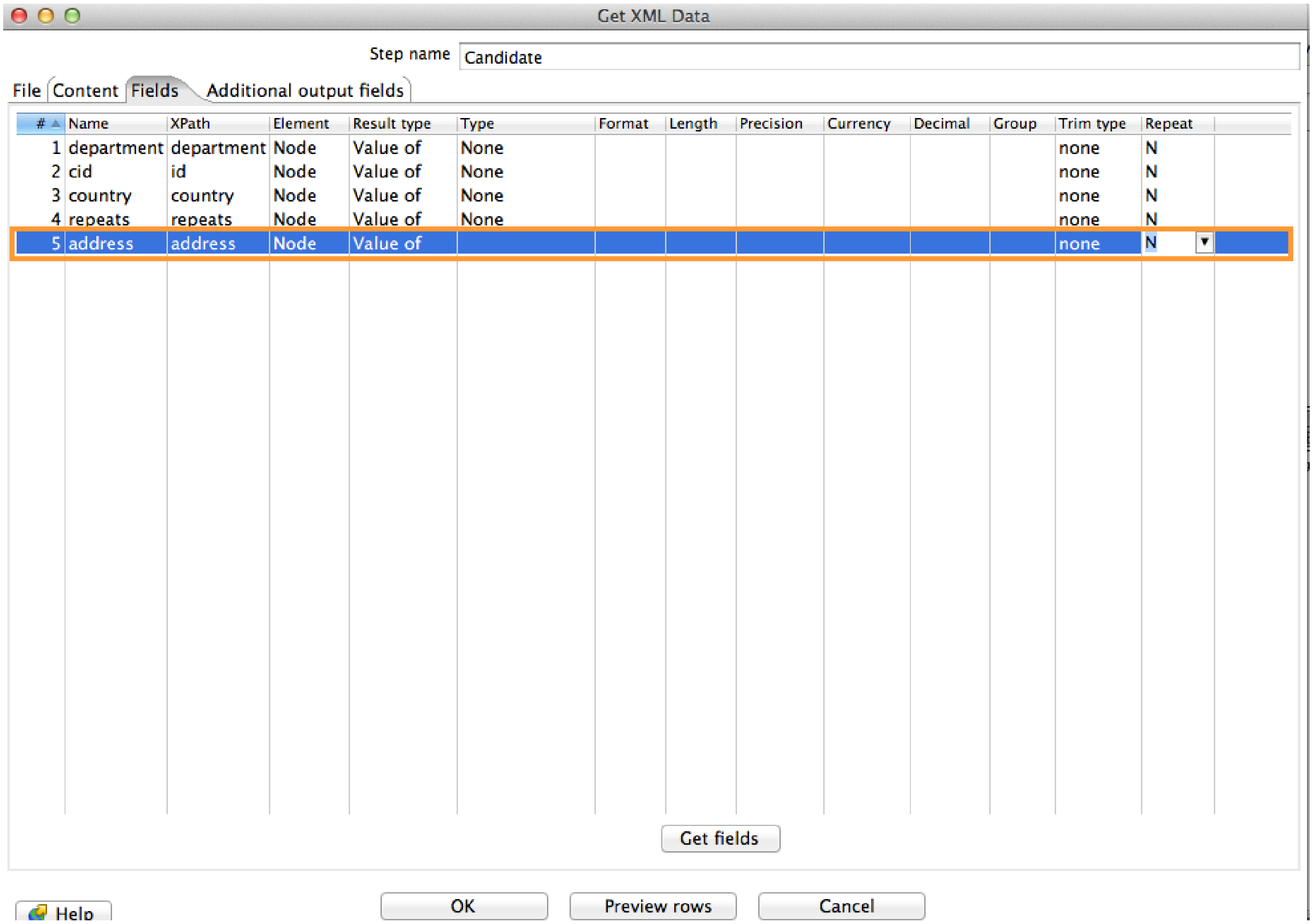


c. Crossing multiple sources:

One technique that is frequently used in order to improve data consistency and data completeness is crossing data from one source, with data from other source(s) available. In this approach, after conducting entity resolution and deciding which entries from all available sources correspond to the same objects in the real world, the entries can be compared and completed. For example, missing values from the entries' fields can be completed with found values from another source. Other choices could be to keep only the entries corresponding to the same object that contain the most information and disregard the rest, or apply some more complex functions and aggregations to the field values coming from multiple sources (min/max/avg value, most recent value etc.).

For the following example we showcase the simple first case, when there are missing values in one field that can possibly be found in the fields of another source. For this purpose, we will use another available source file that contains Candidate's names and addresses in XML format (mdl_user_master.xml) and includes the candidate ids, first and last names and addresses. The candidate id for one user is the same as the candidate id that is found in the records of mdl_user.xml, allowing for the records for the same user to be matched. We will assume that we are interested in having at the output of the ETL process, the correct address of the candidates.

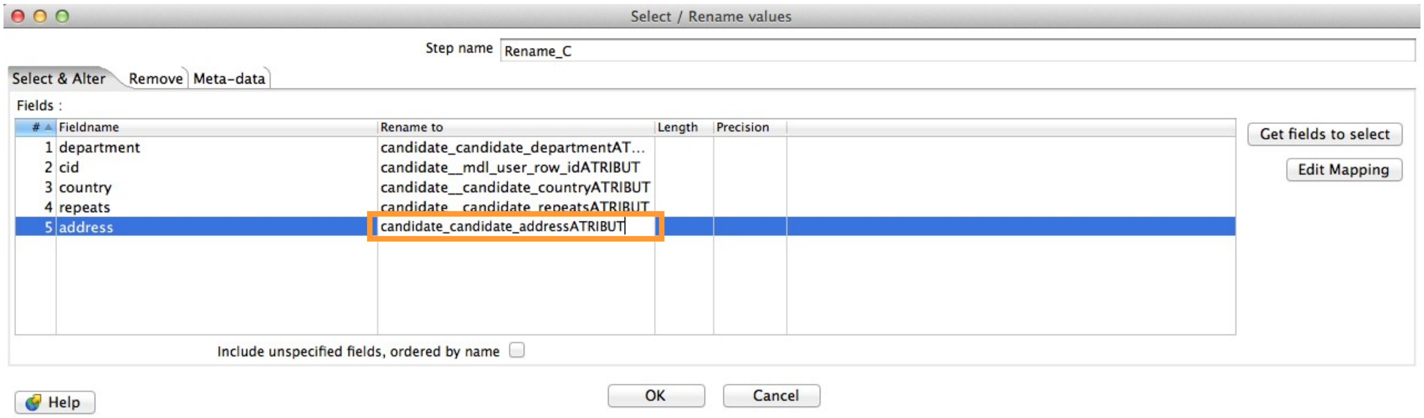
(1) In the "Candidate" source input step, in the "Fields" tab, add the address attribute, as shown in the following figure.



Step name: Candidate

#	Name	XPath	Element	Result type	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	department	department	Node	Value of	None							none	N
2	cid	id	Node	Value of	None							none	N
3	country	country	Node	Value of	None							none	N
4	repeats	repeats	Node	Value of	None							none	N
5	address	address	Node	Value of								none	N

(2) Add the address attribute to the Rename_C step, as shown below.



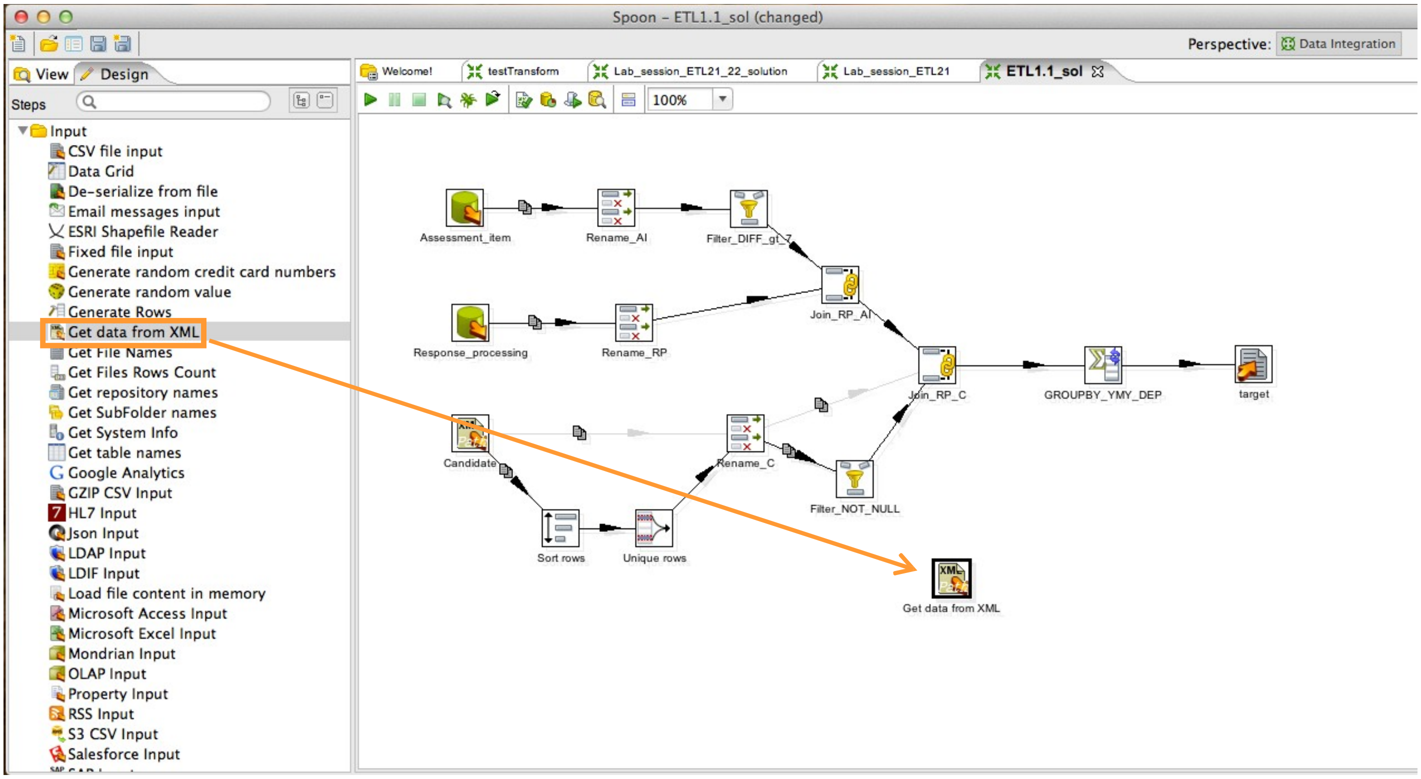
Step name: Rename_C

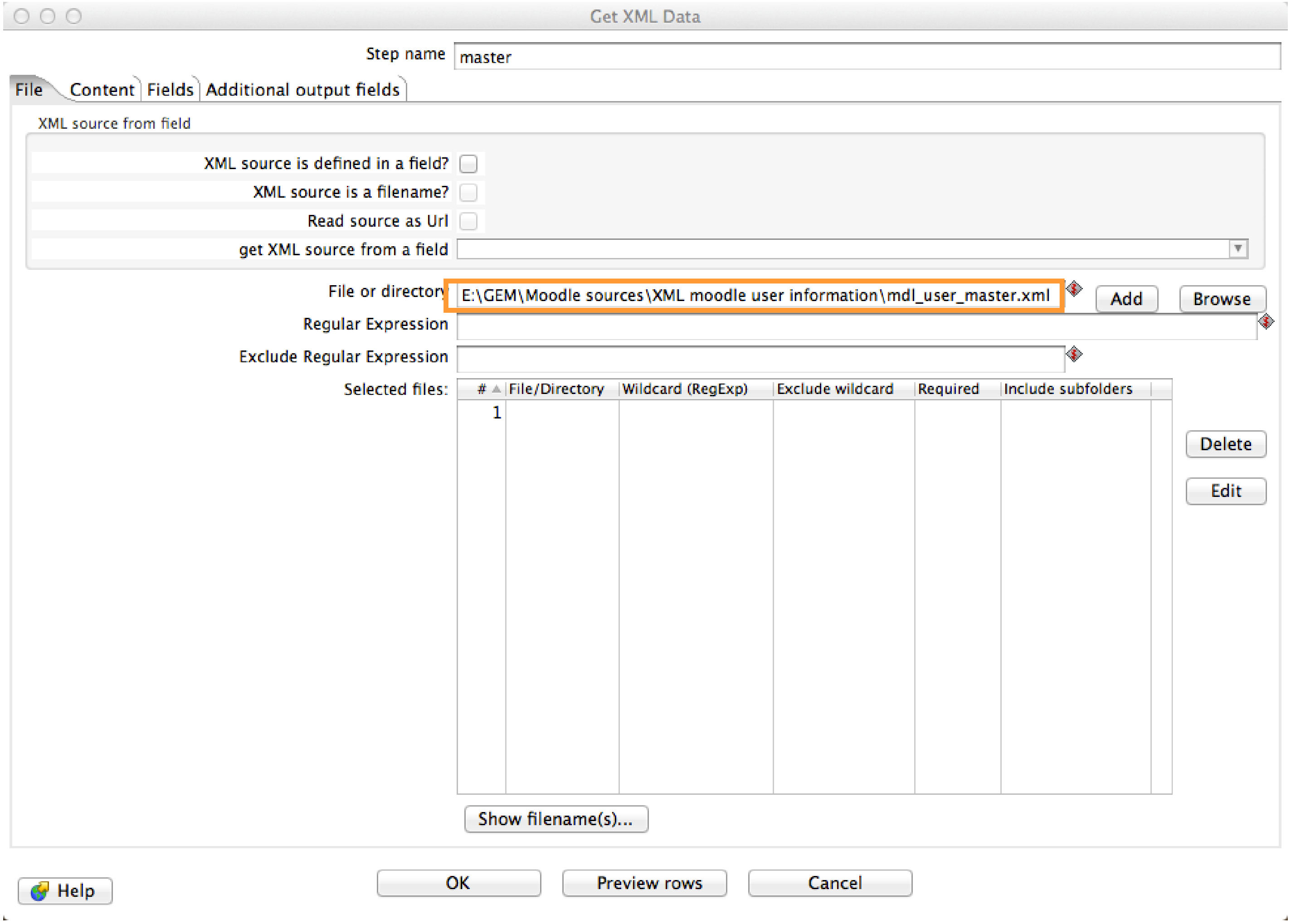
Fields :

#	Fieldname	Rename to	Length	Precision
1	department	candidate_candidate_departmentAT...		
2	cid	candidate__mdl_user_row_idATRIBUT		
3	country	candidate__candidate_countryATRIBUT		
4	repeats	candidate__candidate_repeatsATRIBUT		
5	address	candidate_candidate_addressATRIBUT		

Include unspecified fields, ordered by name □

(3) In a similar manner as we defined data source inputs of the ETL process in Part A, we need to input the xml source using the “Get data from XML” template from the “Input”. We drag the template to the canvas and parameterize as shown in the following figures.





Step name: master

XML source from field

XML source is defined in a field? □

XML source is a filename? □

Read source as Url □

get XML source from a field

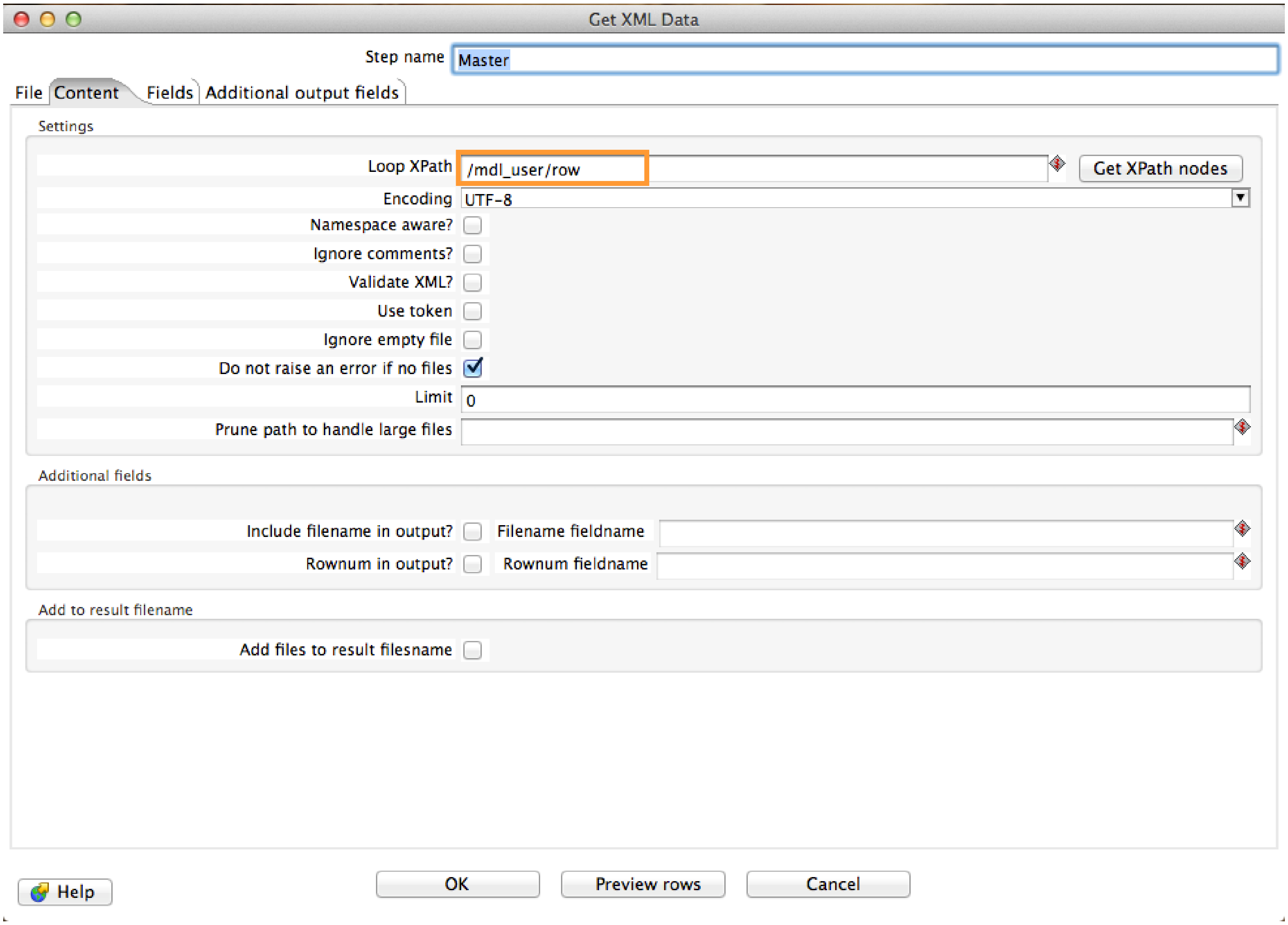
File or directory: E:\GEM\Moodle sources\XML moodle user information\mdl_user_master.xml

Regular Expression

Exclude Regular Expression

Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1					



Get XML Data

Step name: Master

File | Content | Fields | Additional output fields

Settings

Loop XPath: /mdl_user/row

Encoding: UTF-8

Namespace aware? □

Ignore comments? □

Validate XML? □

Use token □

Ignore empty file □

Do not raise an error if no files ☑

Limit: 0

Prune path to handle large files:

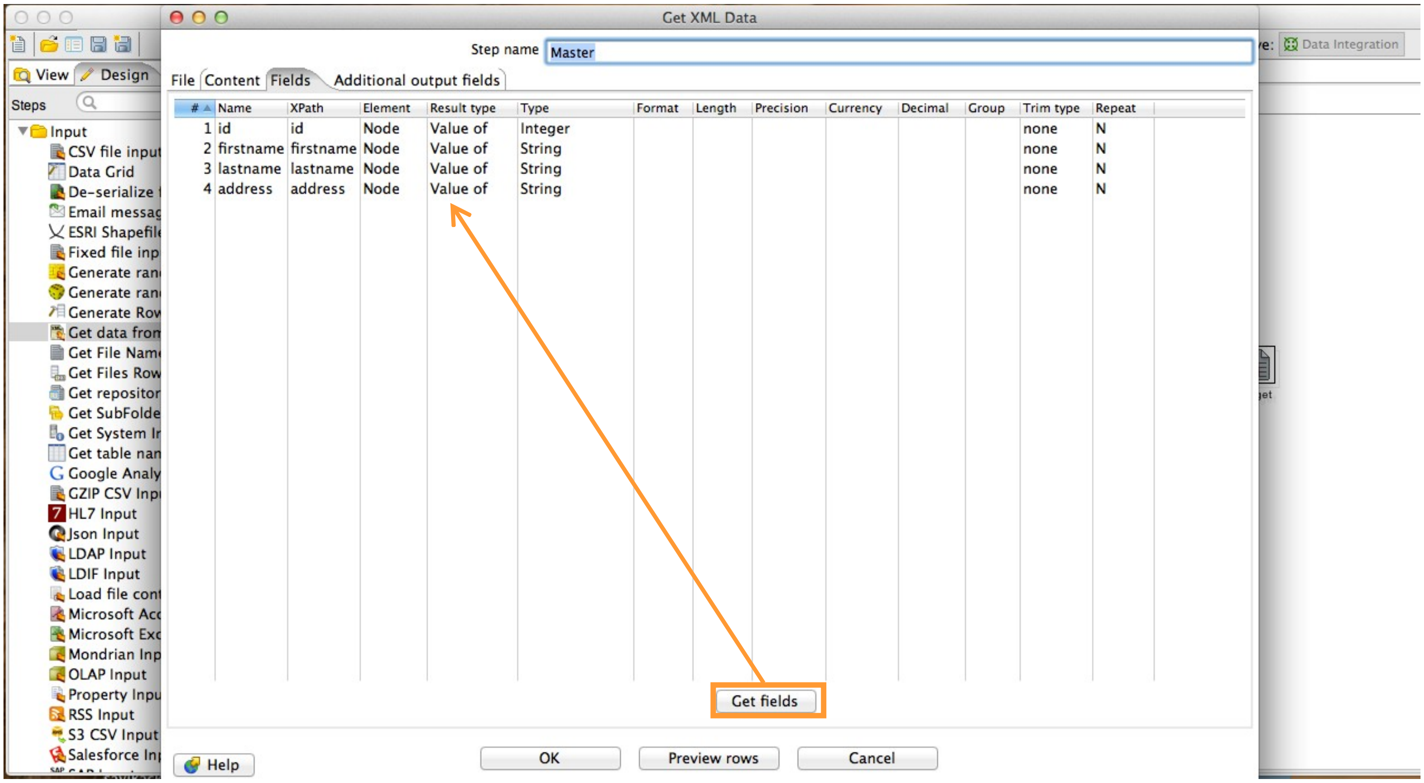
Additional fields

Include filename in output? □ Filename fieldname:

Rownum in output? □ Rownum fieldname:

Add to result filename

Add files to result filesname □



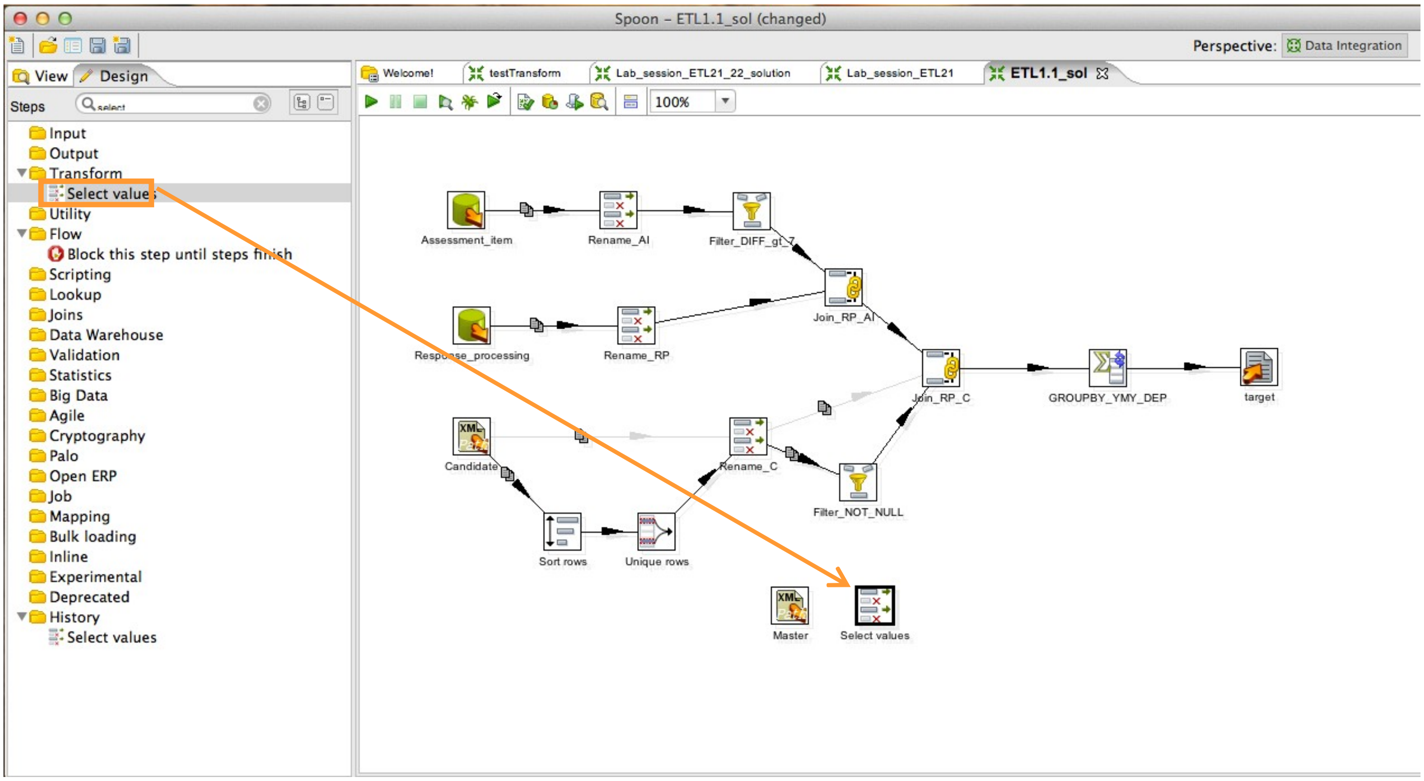
Get XML Data

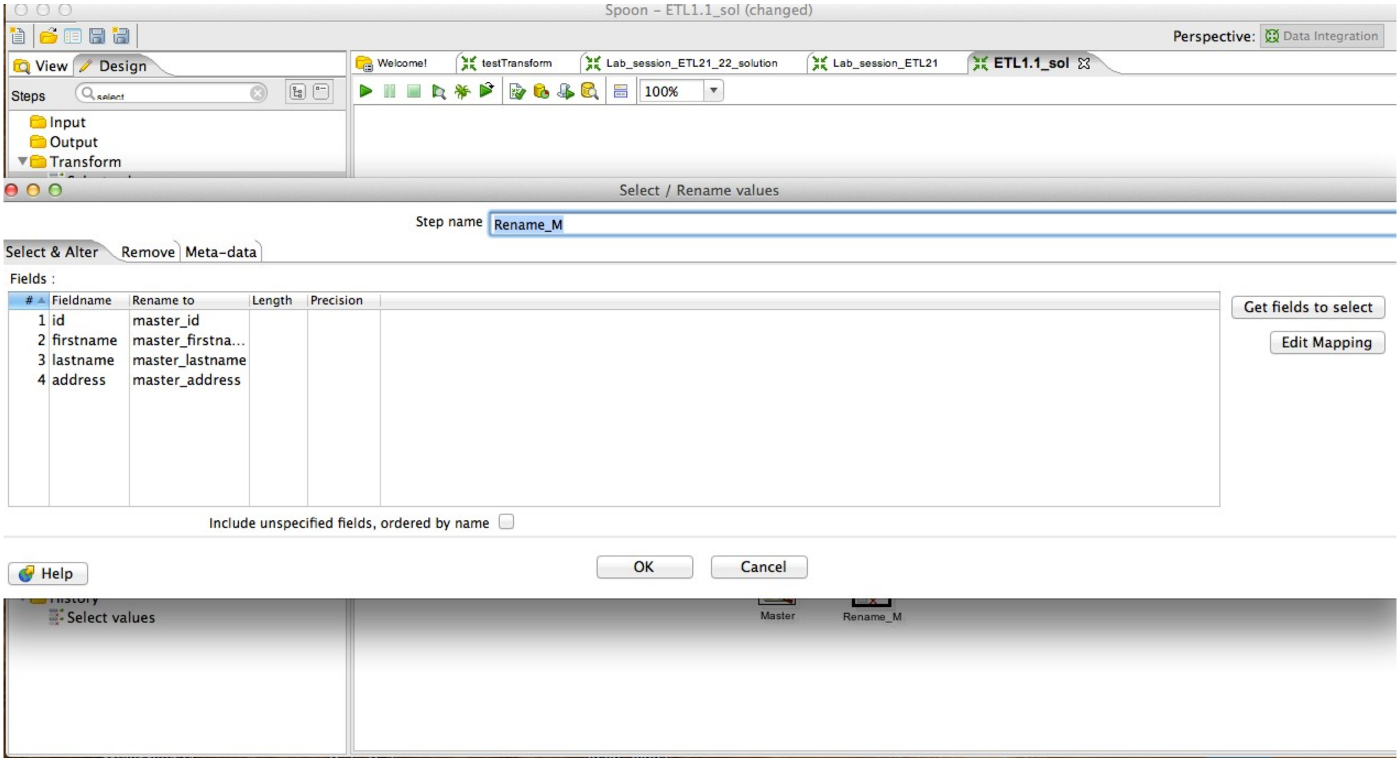
Step name: Master

File | Content | Fields | Additional output fields

#	Name	XPath	Element	Result type	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	id	id	Node	Value of	Integer							none	N
2	firstname	firstname	Node	Value of	String							none	N
3	lastname	lastname	Node	Value of	String							none	N
4	address	address	Node	Value of	String							none	N

(4) As we did with the other data source inputs, we need to add a rename step that will assure that all the field names in the process are unique. From the “Transform” category of the design palette we choose “Select Values” step and drag it to the Spoon canvas. Consequently, we parameterize the select values step as shown in the figures below.




Select / Rename values

Step name: Rename_M

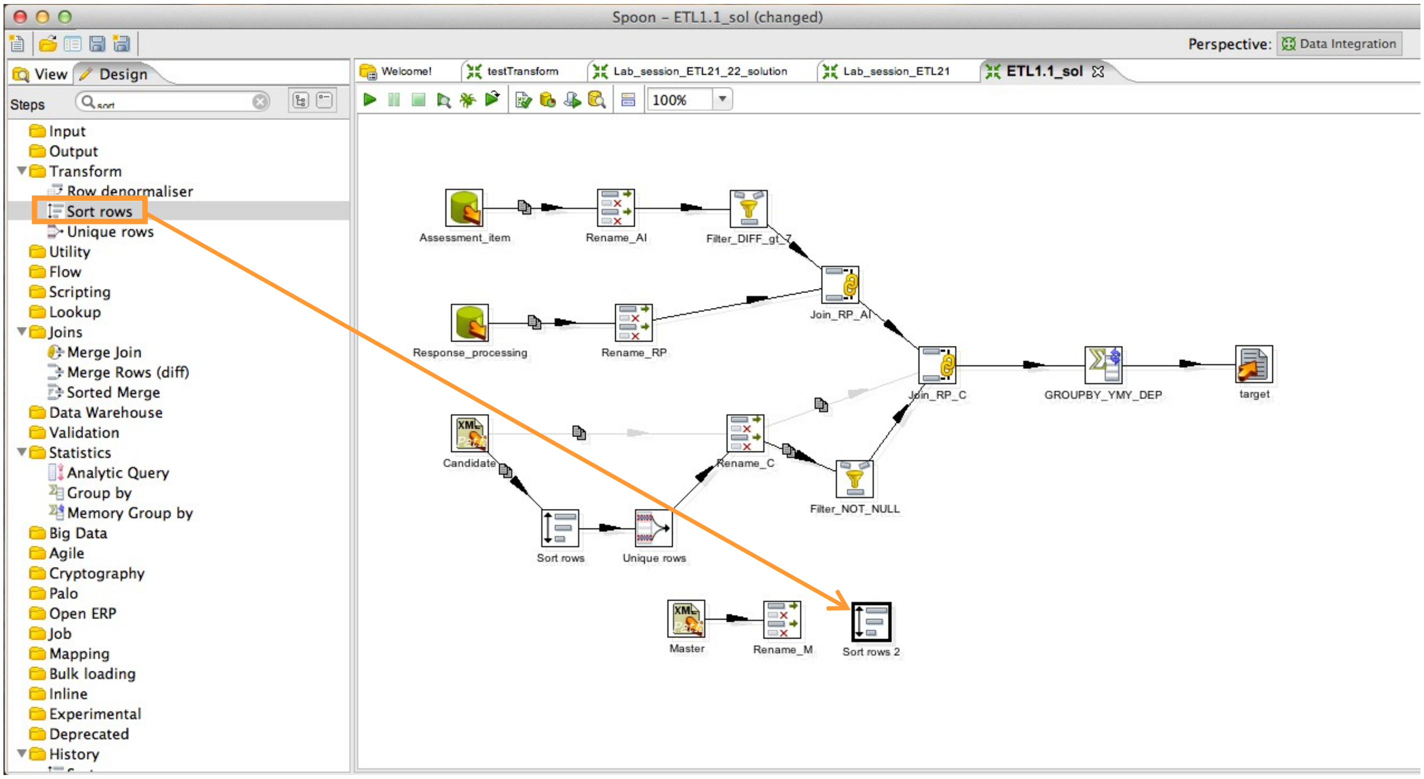
Select & Alter | Remove | Meta-data

Fields :

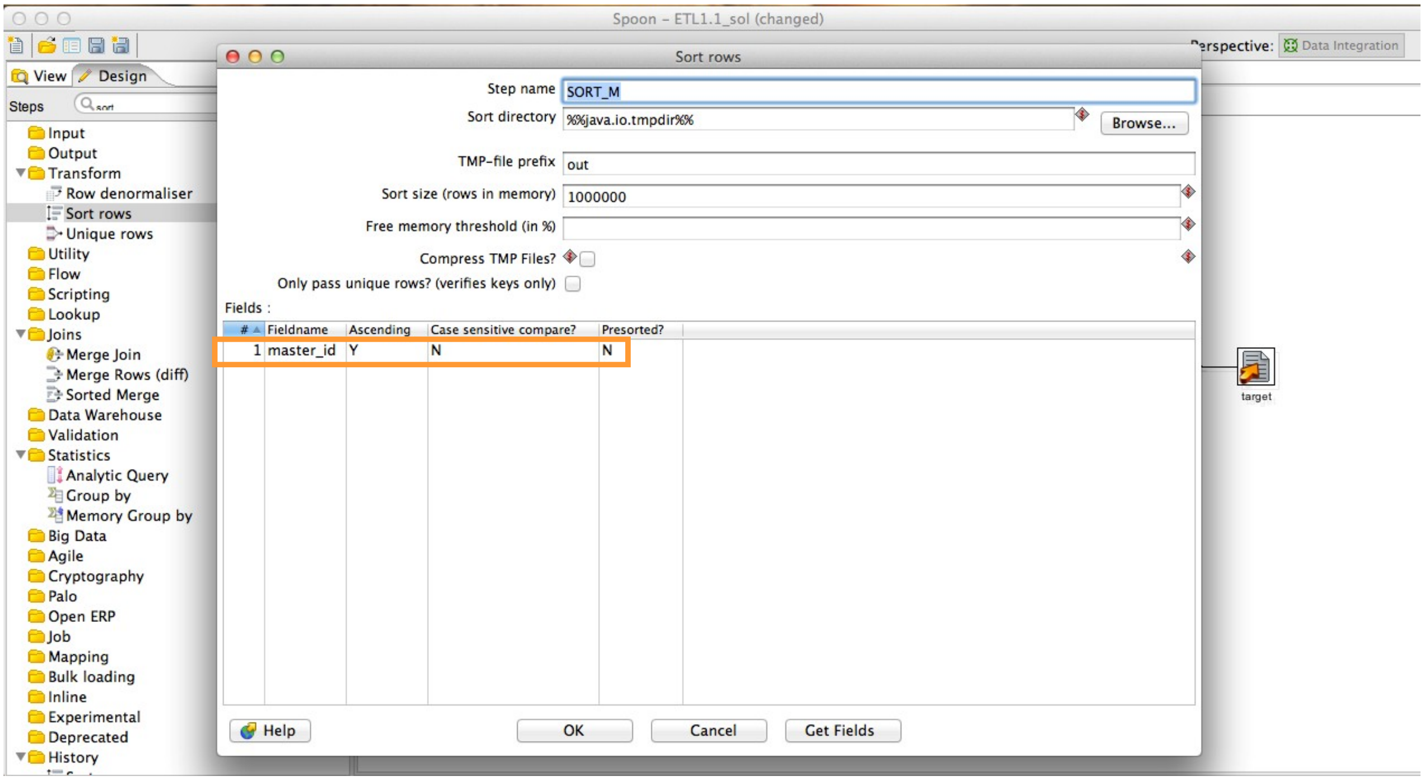
#	Fieldname	Rename to	Length	Precision
1	id	master_id		
2	firstname	master_firstna...		
3	lastname	master_lastname		
4	address	master_address		

Include unspecified fields, ordered by name ☐

(5) From the “Transform” category of the design palette choose “Sort rows” template and drag and drop it to the Spoon canvas as shown in the figure.



(6) We need to parameterize the sort rows step considering the keys of the input. See figure below.



Step name: SORT_M

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

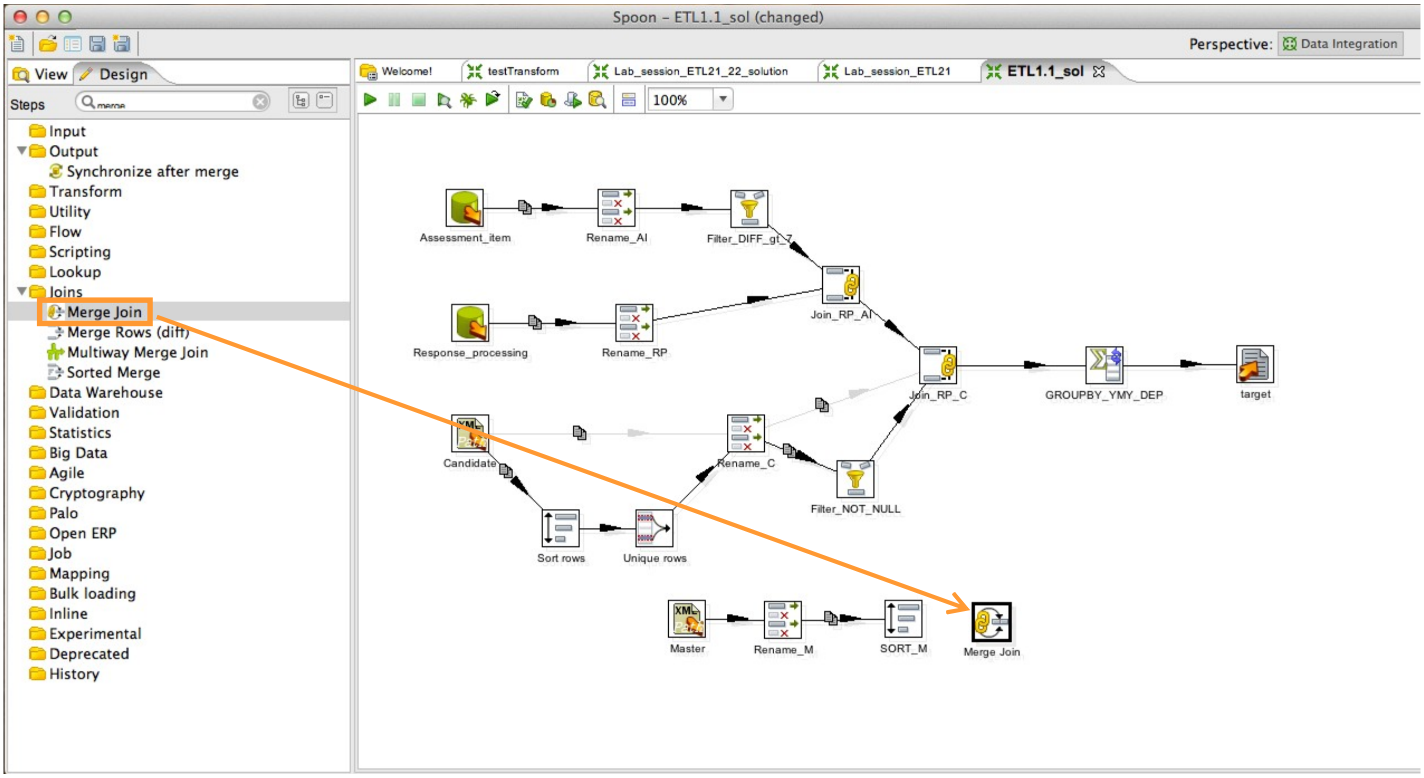
Compress TMP Files? □

Only pass unique rows? (verifies keys only) □

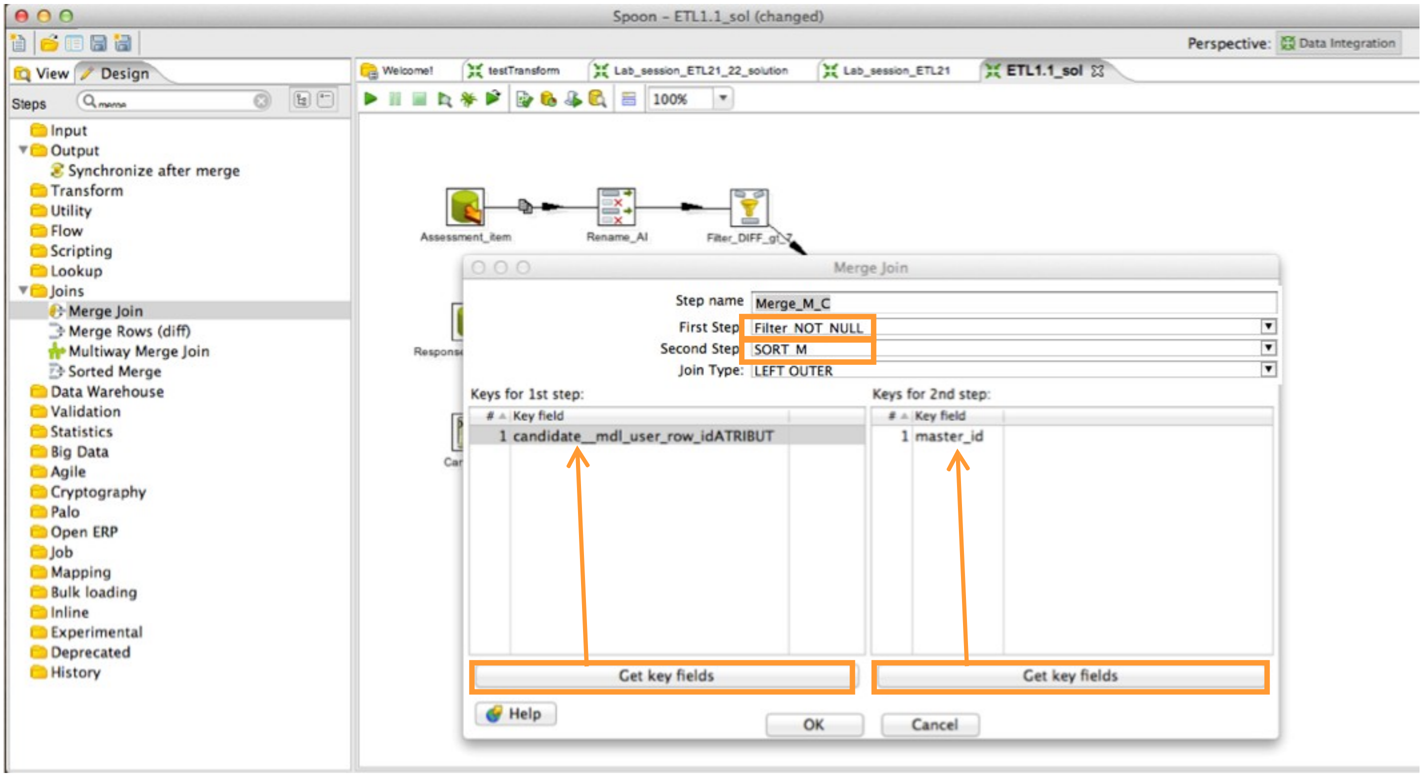
Fields :

#	Fieldname	Ascending	Case sensitive compare?	Presorted?
1	master_id	Y	N	N

(7) Now we need to join data from the two different sources. From the “Joins” category of the design palette choose “Merge Join” template and drag and drop it to the Spoon canvas as shown in the figure.



(8) Then after connecting "Sort_M" and "Filter_NOT_NULL" steps to the added join step and deleting the hop between "Filter_NOT_NULL" and "Join_RP_C", we should parameterize the Join Rows step to properly define the join attributes, in this case candidate mdl_user_row_idATRIBUT and master_id. We also select the "Left Outer" option, which defines a left outer join. See figure below.



Step name: Merge_M_C
First Step: Filter_NOT_NULL
Second Step: SORT_M
Join Type: LEFT OUTER

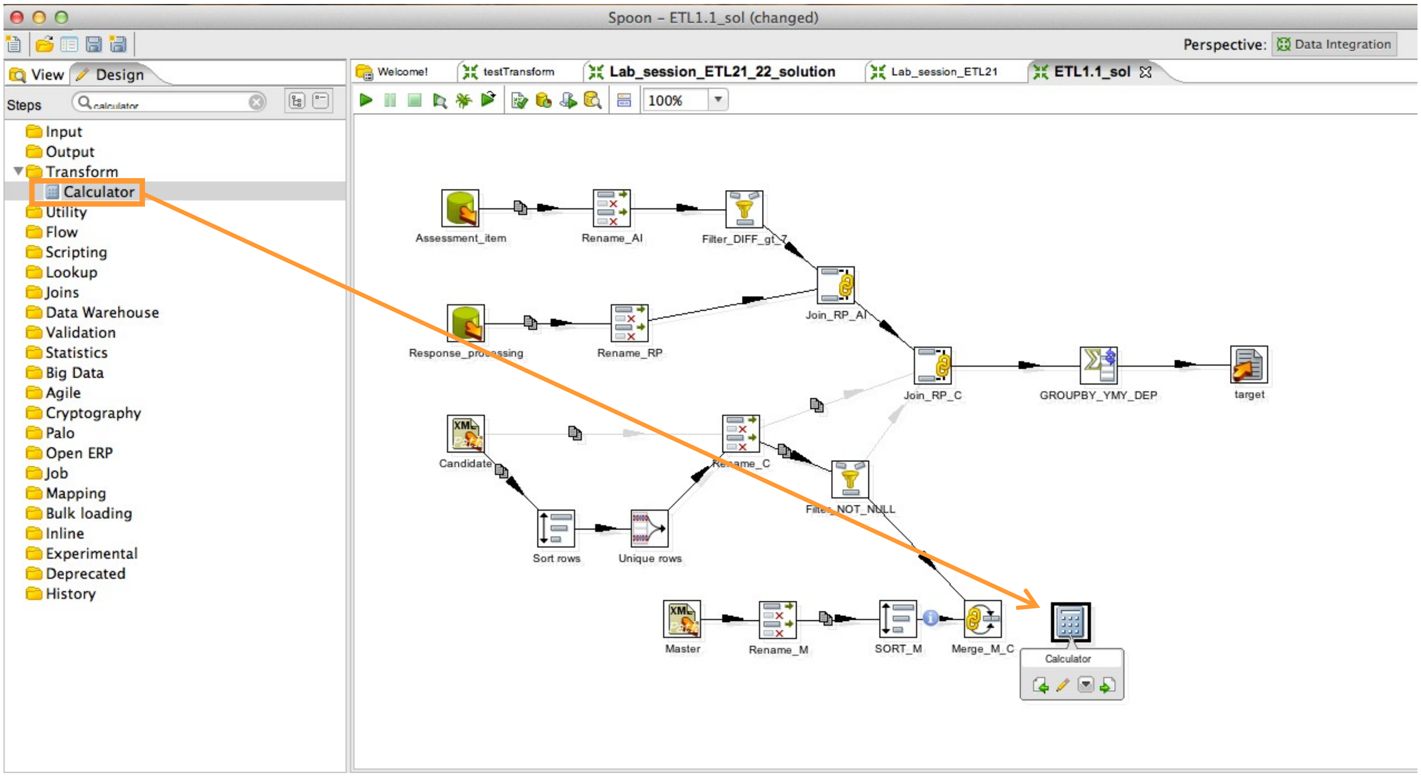
Keys for 1st step:

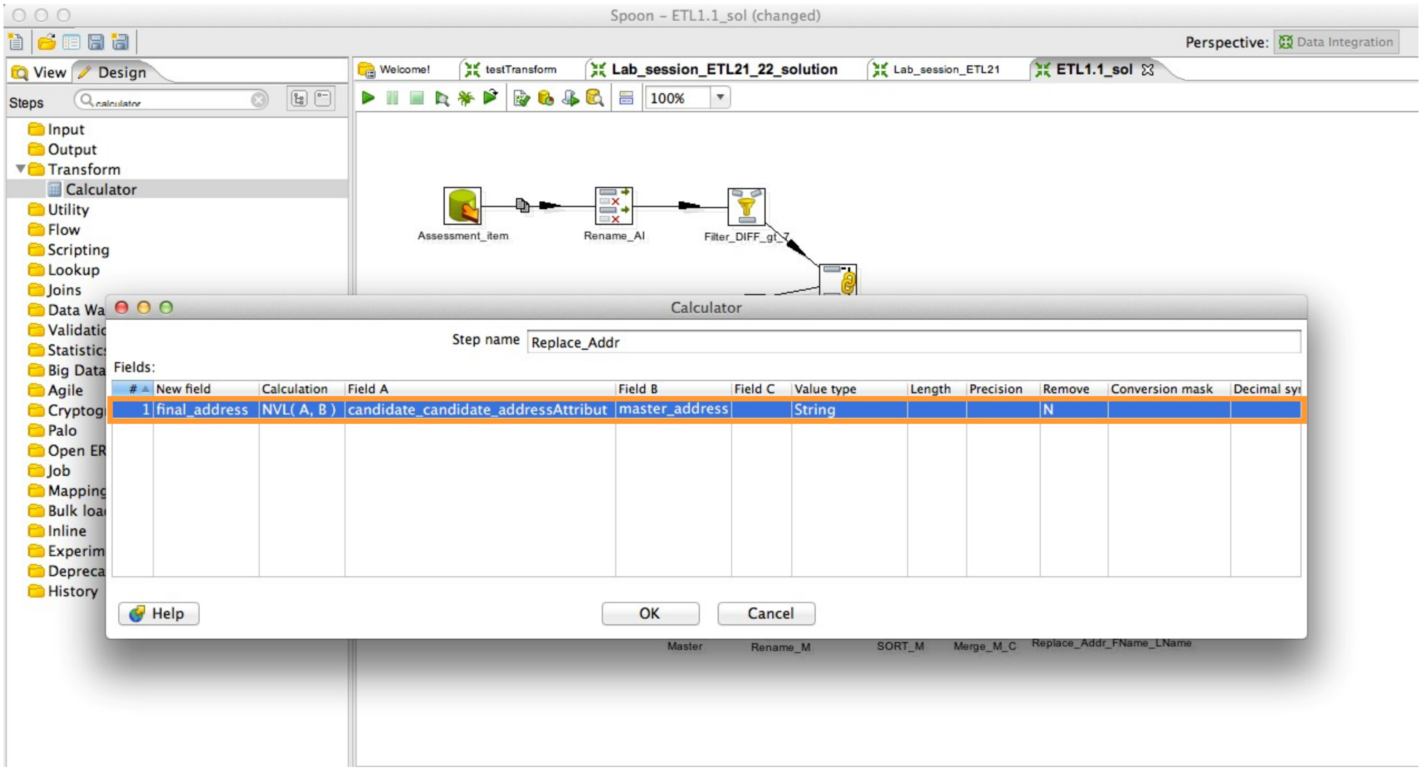
#	Key field
1	candidate__mdl_user_row_idATRIBUT

Keys for 2nd step:

#	Key field
1	master_id

(9) Next we need to provide a function that will check for null values at the address attribute that has derived from the Candidate input source (candidate_candidate_addressATRIBUT) and if there are any null values, replace them with the address values coming from the Master input source (master_address). For this reason, from the "Transform" category of the design palette choose "Calculator" template and drag and drop it to the Spoon canvas and configure it using the NVL() function, as shown in the following figures.



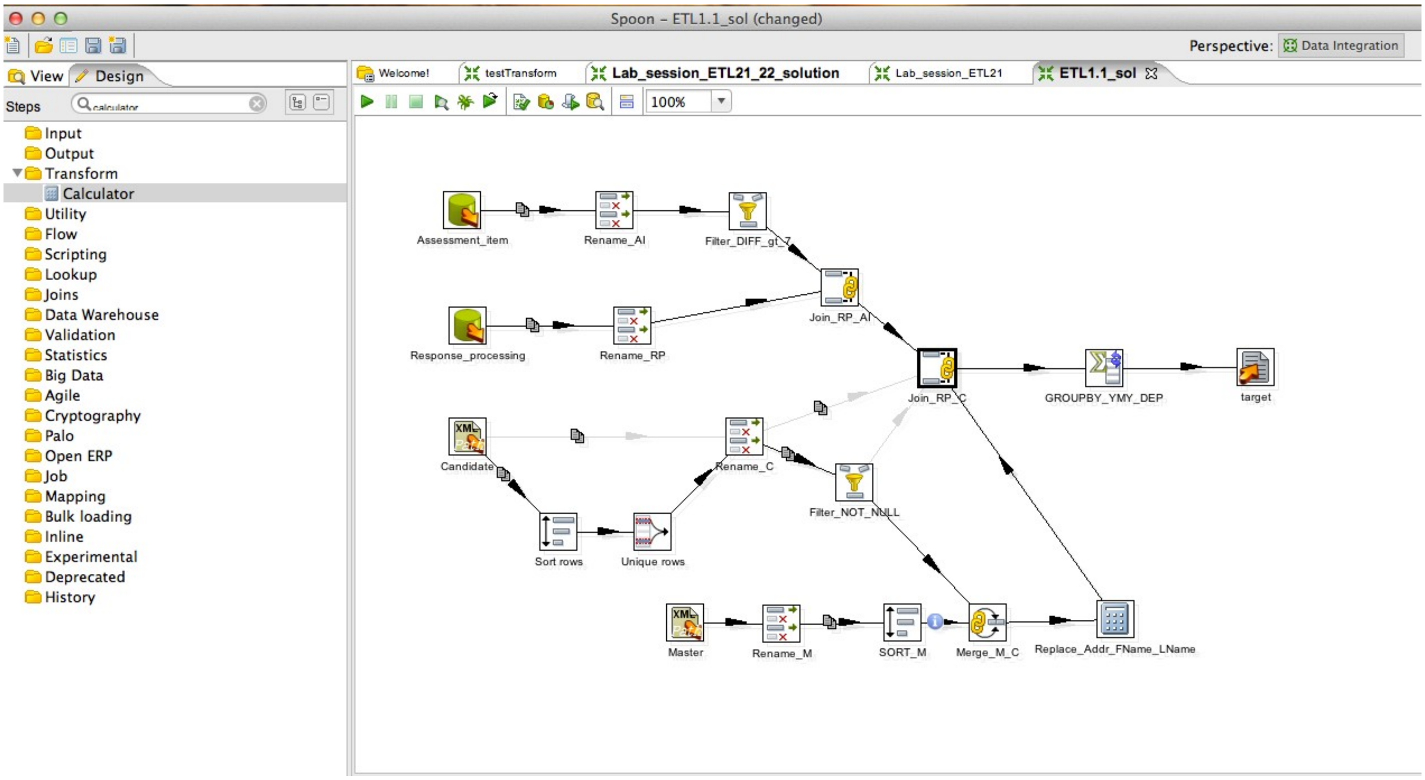


Step name Replace_Addr

Fields:

#	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision	Remove	Conversion mask	Decimal sy
1	final_address	NVL(A, B)	candidate_candidate_addressAttribut	master_address		String			N		

(10) Finally, we add a hop between the “Replace_Addr_FName_LName” and the “Join_RP_C” steps, as cn be seen in figure below. Now the information about candidates that passes through the join, is more complete with regards to candidates’ addresses.



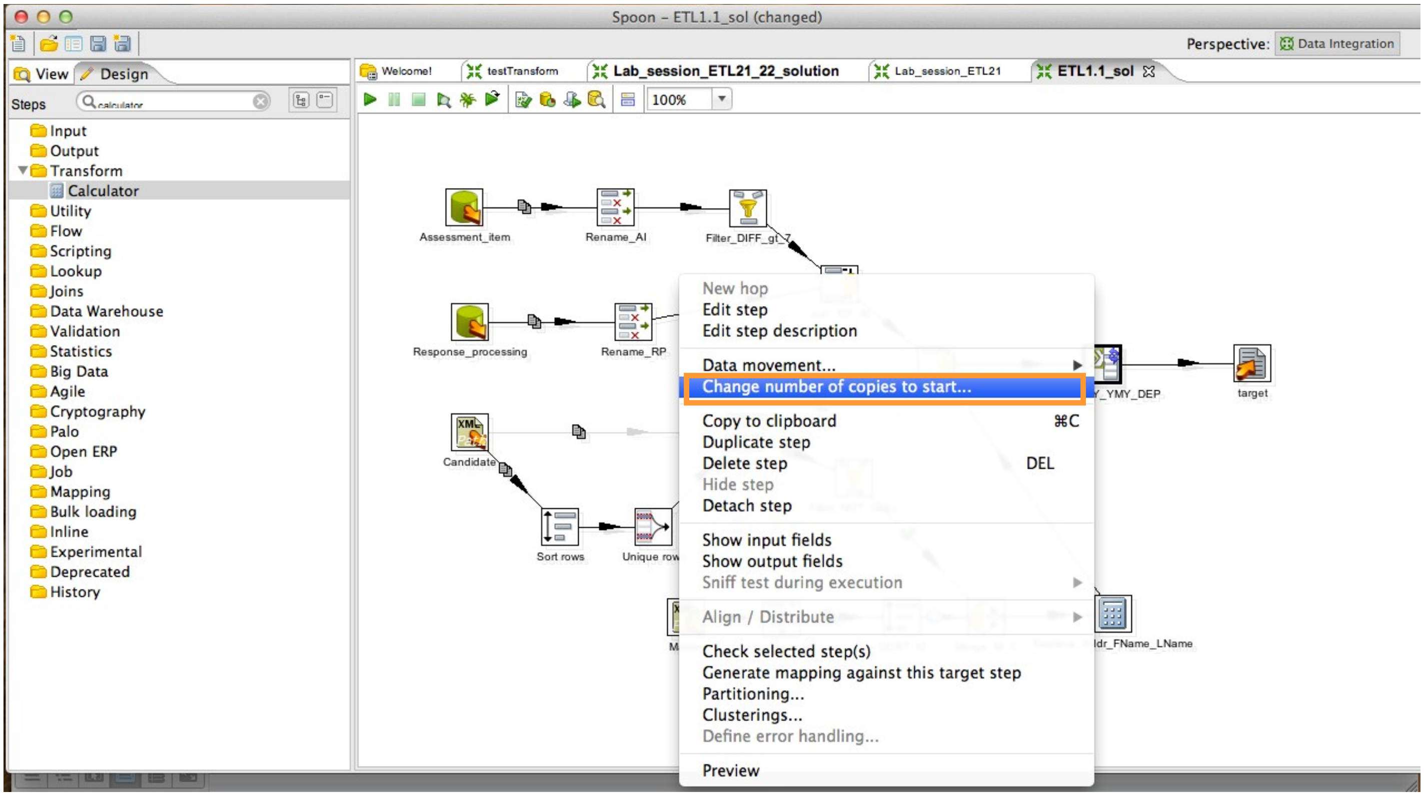
2. Improving Performance:

The performance of the ETL process refers to the execution efficiency of the process, in terms of time efficiency, throughput, resource utilization etc. In this example we will consider the time efficiency of the process, which is the degree of low response times, low processing times and high throughput rates. To achieve that using Pentaho Data Integration tool, we have various options, some of which are described below.

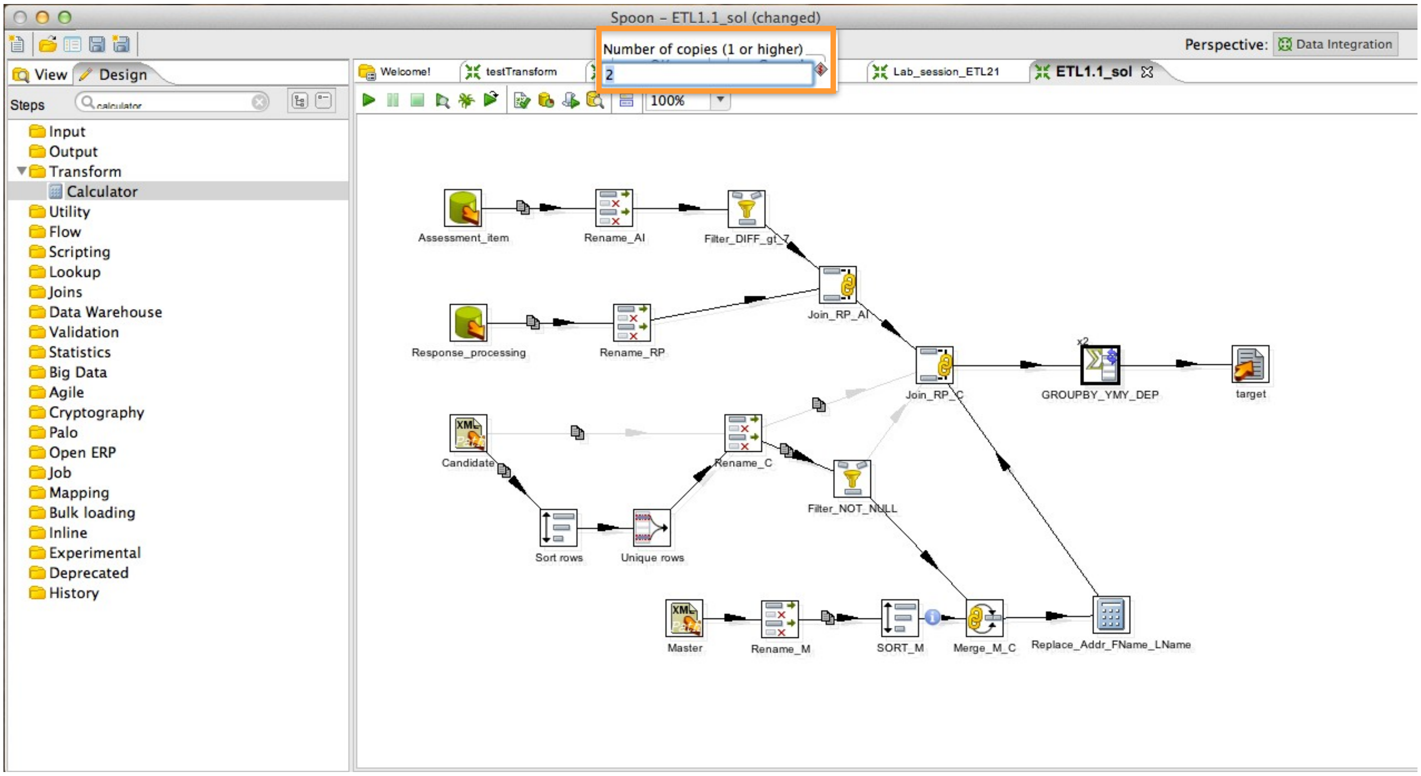
a. Parallelizing the flow:

Parallelization refers to the concurrent execution of activities belonging to the same process, thus resulting in faster execution time and higher throughput of the process. In Pentaho Data Integration tool, this can be achieved by increasing the number of copies (threads) of a specific task. Consequently, execution of this task is distributed among more than one threads that run at the same time. However, resources should not be spent with wastefulness and therefore it would be a good idea to increase (first) the number of copies for tasks that are time-consuming\computationally intensive, such as calculations, aggregations in an ETL process.

(1) Right click on the “GROUPBY_YMY_DEP” step and select the option “Change number of copies to start”, as shown in the figure below.



(2) Select the number of copies and you will notice a "x<number_of_copies>" label on the top left corner of the task



Number of copies (1 or higher)

2

b. Assigning more memory:

A process with more available memory can faster execute operations such as sorting and joining, since more entries to be processed can be at the same time "loaded" on memory, where algorithms are applied. One way to achieve more memory allocation to specific tasks in Pentaho Data Integration tool, is by increasing the cache size of operations.

(1) Right click on any join step and select the option "Change number of copies to start", as shown in the figure below.

(2) Select the number of rows to cache before the system reads data from temporary files, by setting the "Max. cache size" field.

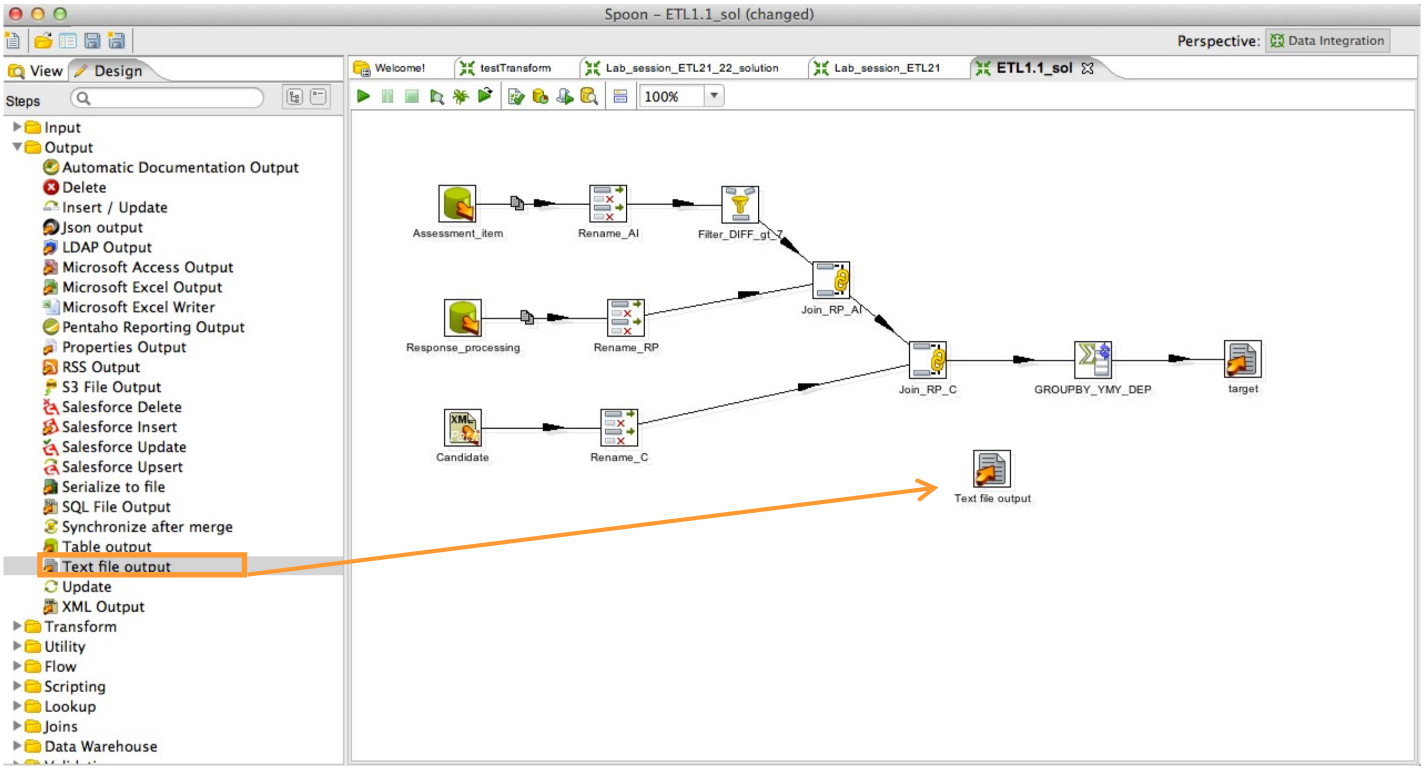
3. Improving Reliability:

Reliability of an ETL process is the degree to which the ETL process can maintain a specified level of performance for a specified period of time. It includes *robustness*, the degree to which the process operates as intended despite unpredictable or malicious input and *recoverability*: the degree to which the process can recover the data directly affected in case of interruption or failure. To improve this quality dimension using Pentaho Data Integration tool, we can add recovery points and handle errors, as described below.

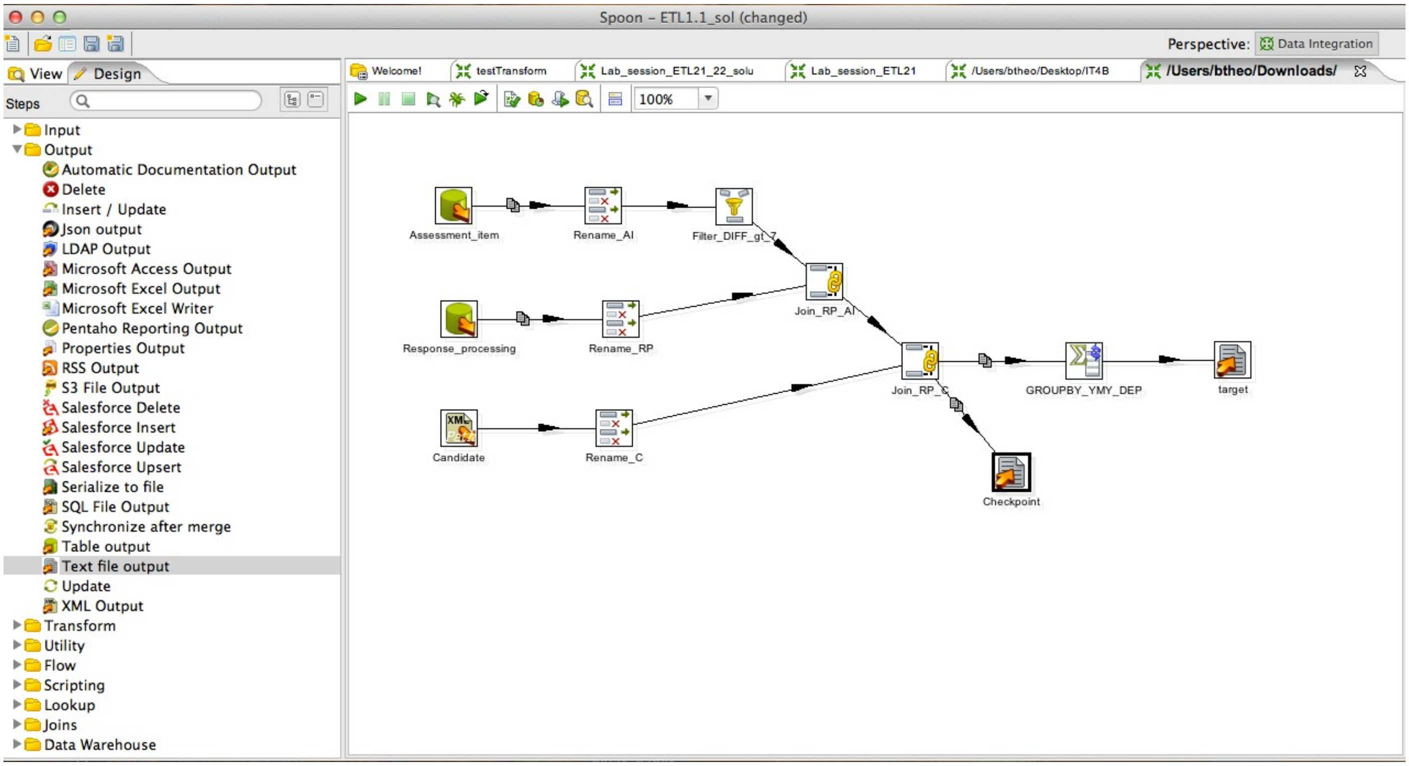
a. Adding recovery points:

Recovery points are execution "checkpoints" that store intermediary execution information so that a process can resume in case of interruption or failure. In Pentaho Data Integration tool this can be performed by adding output files after specific tasks. It would be a good idea to add checkpoints after tasks that are time-consuming\computationally intensive, such as group-by operations, calculations, aggregations etc., in the ETL process.

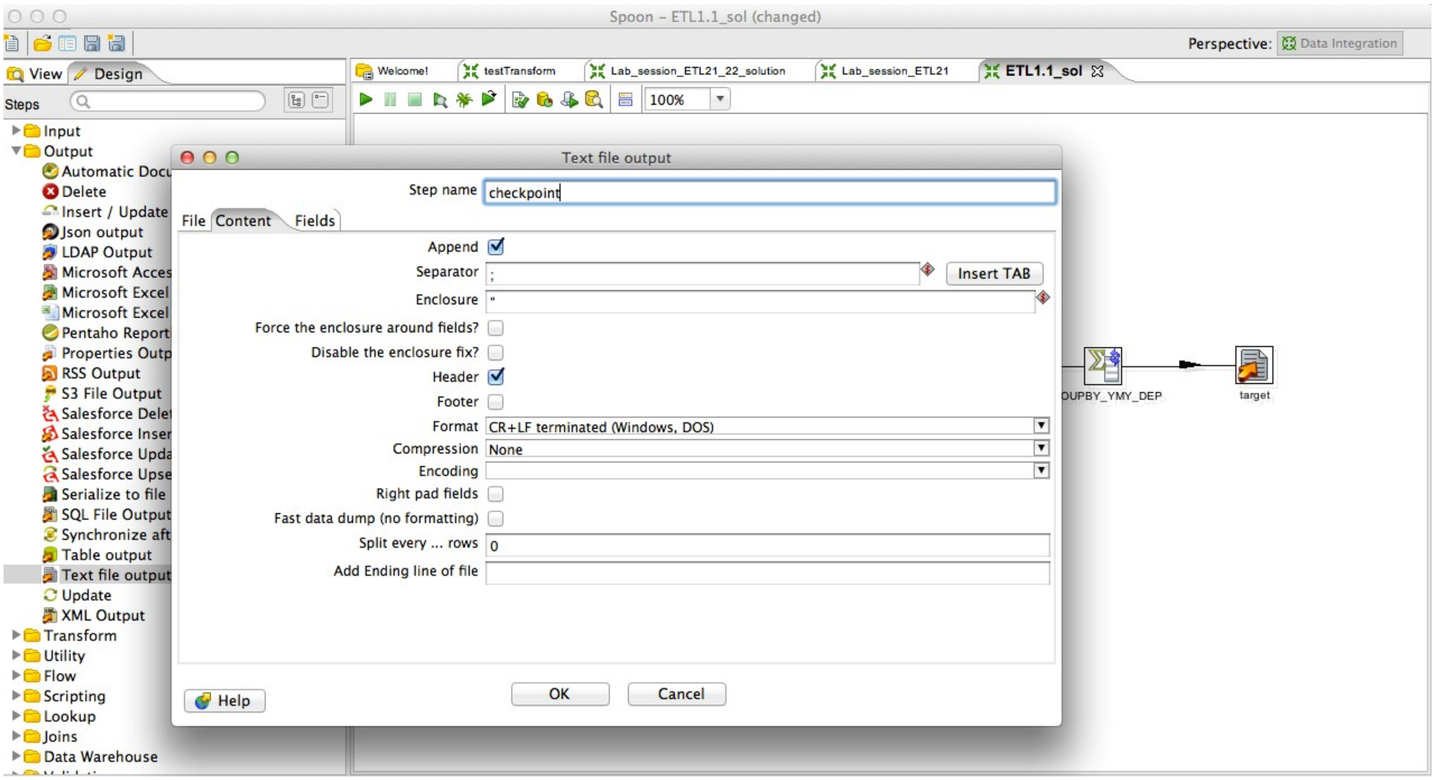
(1) From the "Output" category of the design palette choose "Text file output" template and drag and drop it to the Spoon canvas as shown in the figure.



(2) Connect the created recovery point (checkpoint) to the last join operation (Join_RP_C).



(3) Configure the Content and Fields parameters as shown in the following pictures.



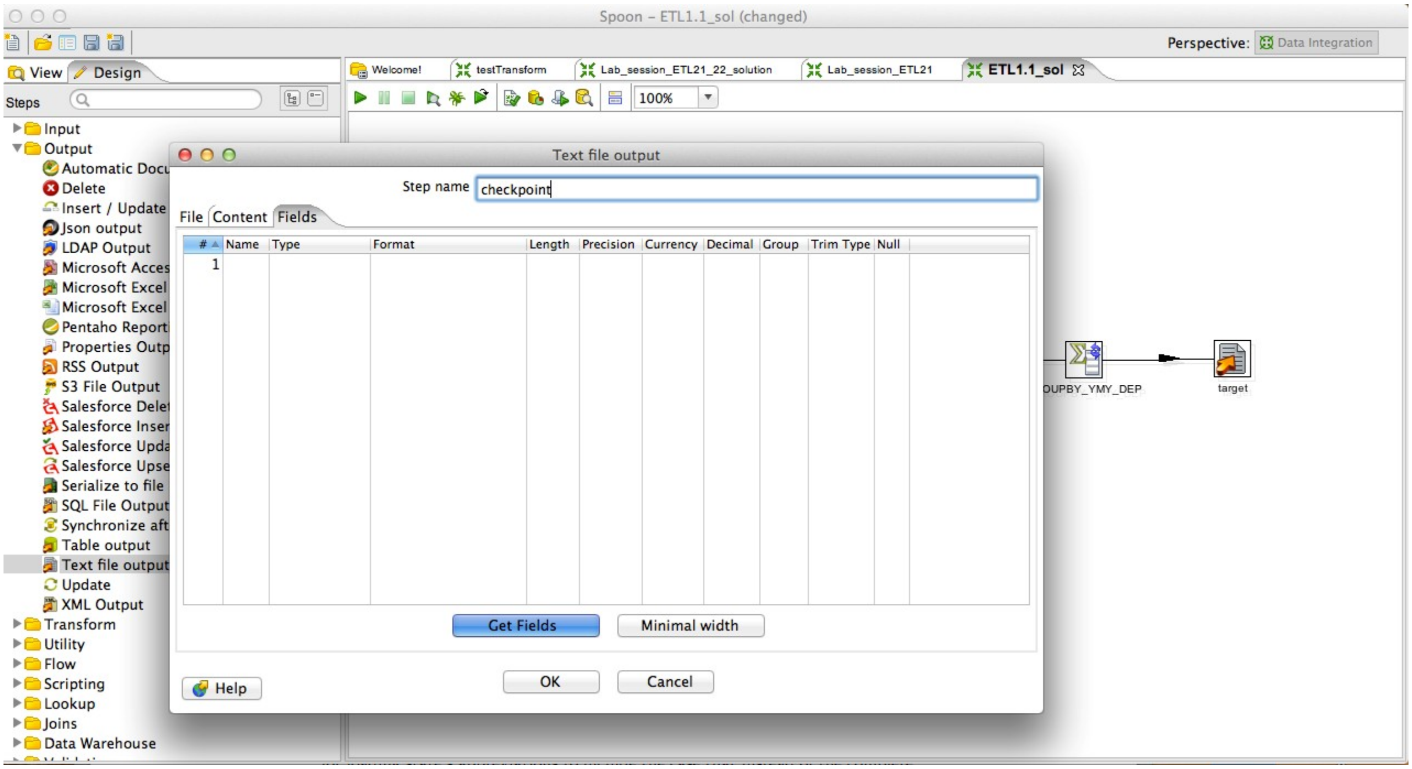
Spoon – ETL1.1_sol (changed)

Text file output

Step name: checkpoint

File | Content | Fields

- Append ☑
- Separator: ;
- Enclosure: "
- Force the enclosure around fields? □
- Disable the enclosure fix? □
- Header ☑
- Footer □
- Format: CR+LF terminated (Windows, DOS)
- Compression: None
- Encoding:
- Right pad fields □
- Fast data dump (no formatting) □
- Split every ... rows: 0
- Add Ending line of file:



Spoon – ETL1.1_sol (changed)

Text file output

Step name: checkpoint

File | Content | Fields

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim Type	Null
1										