ONE Desktop Workshop

Lookups & Dictionaries

Prepared for:	v15.4.x
Prepared by:	Ataccama
Dated:	October 2024

Contents of the Document

1. Introduction

Lookups are lists of values captured in binary files of a specific format (***.lkp**) created for corrections, partial or exact matches, or providing replacements of data in a flow. In this workshop we will work with building lookups and using them for two main purposes:

- **Verification** - checking if the value appears in the lookup file.
- **Enrichment** - using the lookup file to obtain related attributes.

2. Tasks

For this exercise you will need the following data files:

-	**addresses.csv**	data with Canadian addresses
-	**ca_zip.csv**	certified list of valid ZIP codes
-	**ca_province.csv**	list of existing Canadian provinces
-	**ca_city.csv**	list of existing Canadian cities
-	**ca_street_apply_replacements.csv**	sample data for replacing values.

The **addresses.csv** file will be your data source – put it into **data\in**. The other 4 files will be your resources for working with lookup files – move them into **data\ext\src**.

2.1. Create a plan and add a data source.

Firstly, we need a new plan that will create our lookup file. This is a one-time action – once the lookup is created it can then be used in the upcoming plans we will create.

› Let's create a new plan called **05_address_lookup_builder.plan** in the **data\ext\build** folder.

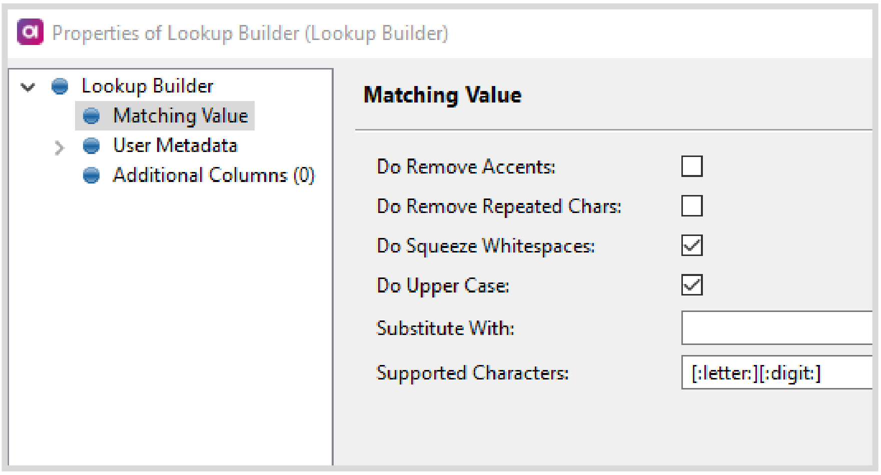
› Add the **ca_zip.csv** into the plan and make sure you have the metadata set correctly.

HINT *Set the **File type:** option to '**Delimited**' rather than '**Fixed**' in the top left section to get rid of unwanted characters in your values.*

2.2. Lookup builder for verification

Once the original CSV file with ZIP codes is properly recognized, it is time to convert its values into the lookup list in the ***.lkp** file format:

- Find the **Lookup builder** step, add it to the plan, and connect it to the **Text File Reader**.
- Open the **Lookup builder** step.
- Set the ***Duplicities*** option to **FIRST**. This ensures keys in the lookup will be unique.
- The ***File Name*** attribute is for where the lookup file is stored. Best practice is to put them in **data\ext\lkp**. Also, we need to name the lookup file - **ca_zip.lkp**.
- Set the ***key*** attribute. (There's only 1 option…!)
- Click on the composite element link for ***Matching Value***. Select ***Squeeze Whitespaces***, set up ***Upper Case***, and support only ***letters*** and ***digits***.

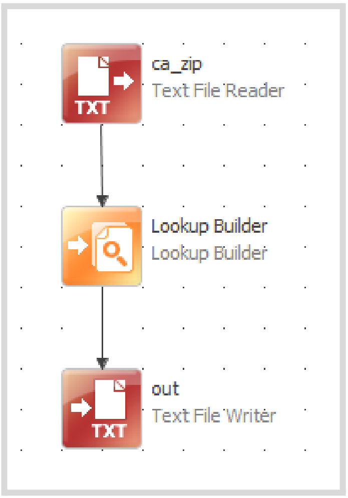

Properties of Lookup Builder (Lookup Builder)

- Lookup Builder
  - Matching Value
  - User Metadata
  - Additional Columns (0)

Matching Value

Do Remove Accents:	☐
Do Remove Repeated Chars:	☐
Do Squeeze Whitespaces:	✓
Do Upper Case:	✓
Substitute With:	
Supported Characters:	[:letter:][:digit:]

- Add a Text **File Writer** step to store rejected records. (duplicities will be stored in **data\out\ca_zip_reject.csv**)



- **Run** your plan to generate the lookup file.

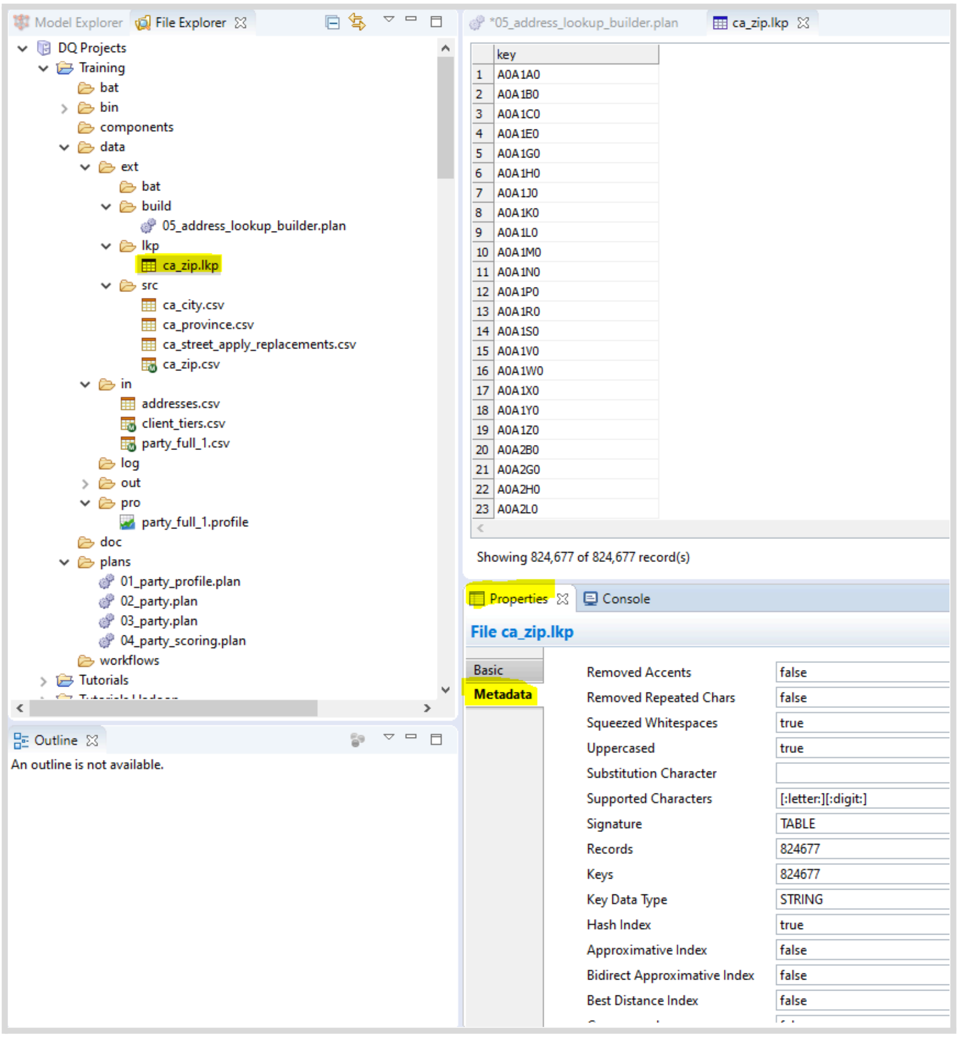
All done! Your lookup file should now be created in the folder specified. Let’s use it:

› Open the **ca_zip.lkp** lookup file in IDE by double-clicking it. Its preview opens in IDE just like a regular text file.

HINT

You can verify what options had been chosen when the lookup file was generated.

› Select the **ca_zip.lkp** file in ***File Explorer*** and check the ***Properties*** window at the bottom. Select the ***Metadata*** menu and scroll down - you can see here the Matching Values options that had been chosen for the lookup file. Useful for when someone directly gave you the lookup file and you didn’t build it yourself.



	key
1	A0A1A0
2	A0A1B0
3	A0A1C0
4	A0A1E0
5	A0A1G0
6	A0A1H0
7	A0A1J0
8	A0A1K0
9	A0A1L0
10	A0A1M0
11	A0A1N0
12	A0A1P0
13	A0A1R0
14	A0A1S0
15	A0A1V0
16	A0A1W0
17	A0A1X0
18	A0A1Y0
19	A0A1Z0
20	A0A2B0
21	A0A2G0
22	A0A2H0
23	A0A2L0

Showing 824,677 of 824,677 record(s)

File ca_zip.lkp

Removed Accents	false
Removed Repeated Chars	false
Squeezed Whitespaces	true
Uppercased	true
Substitution Character	
Supported Characters	[:letter:][:digit:]
Signature	TABLE
Records	824677
Keys	824677
Key Data Type	STRING
Hash Index	true
Approximative Index	false
Bidirect Approximative Index	false
Best Distance Index	false

2.3. Using lookup for verification

With the existing lookup file (**ca_zip.lkp**), we can now use it in a plan to verify our source data for correct values of the ZIP code:

- Create another plan called **05_address_verify.plan** in **plans** folder.
- Add **addresses.csv** as a data source. (Open it up and double-check the data!)
- Add an **Alter Format** step and use it to create 3 new columns:
  - **std_zip** (STRING)
  - **sco_zip** (INTEGER)
  - **exp_zip** (STRING)
- Add a **Lookup** step and open its properties by double-clicking.
- Make sure you are on the ***General*** tab and set the **Key Lookup Value**.

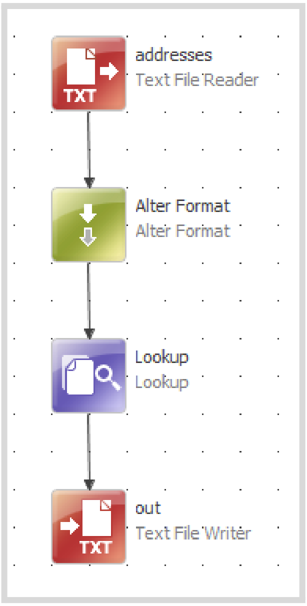
TASK

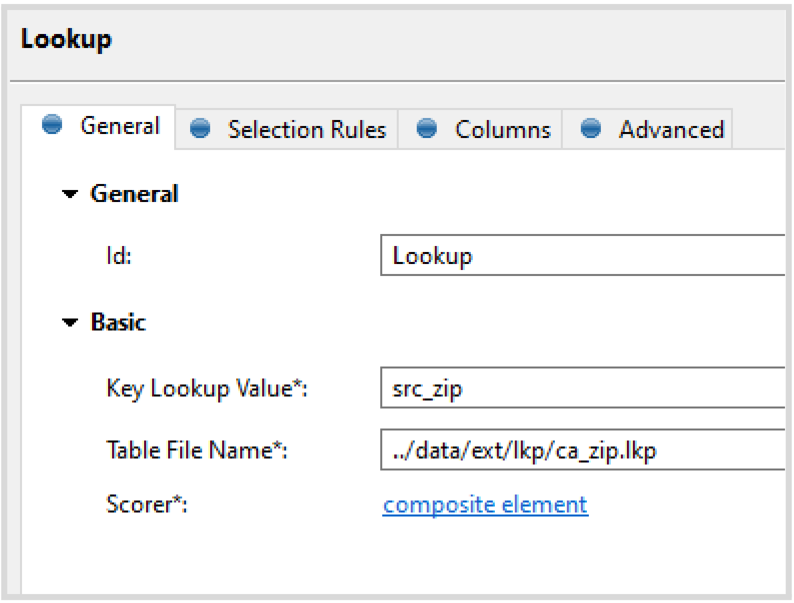
Knowing that you need to check your data against a list of ZIP codes, what would be the key to checking from this file?

- Fill the **Table File Name** - **data\ext\lkp\ca_zip.lkp** file that you built in the previous task.
- In the ***Scorer*** composite element:
  - set the **sco_zip** to be the **Score column.**
  - set the **exp_zip** to be the **Explanation column.**
- You can adjust the scoring here if you want, for example:
  - give a high score of 10,000,000 for LKP_NULL_ID and LKP_NOT_FOUND
  - give a lower score of 100 for LKP_APPROX_USED
- In the ***Columns*** tab, assign the lookup.key to column **std_zip**.
- Add a **Text File Writer** step to store the output at **data\out\addresses_out.csv**.
- **Run** the plan and check your output file.

See how the **std_zip** is null where it could not find a match and how the scoring and explanation for records differ when the **src_zip** itself was null vs when **src_zip** could not be matched against the lookup.

The following pictures will help you with your plan and **Lookup** step setup:




Lookup

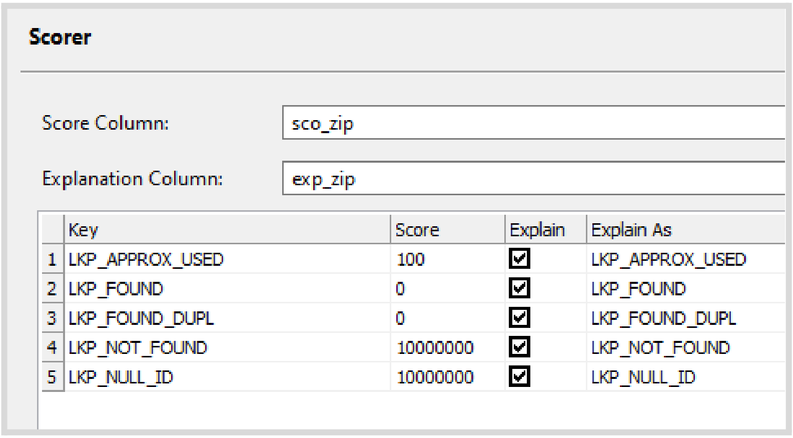
General | Selection Rules | Columns | Advanced

General

Id:	Lookup

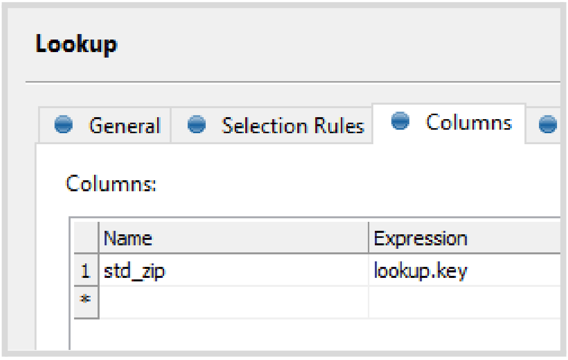
Basic

Key Lookup Value*:	src_zip
Table File Name*:	../data/ext/lkp/ca_zip.lkp
Scorer*:	composite element


Scorer

Score Column:	sco_zip
Explanation Column:	exp_zip

	Key	Score	Explain	Explain As
1	LKP_APPROX_USED	100	✓	LKP_APPROX_USED
2	LKP_FOUND	0	✓	LKP_FOUND
3	LKP_FOUND_DUPL	0	✓	LKP_FOUND_DUPL
4	LKP_NOT_FOUND	10000000	✓	LKP_NOT_FOUND
5	LKP_NULL_ID	10000000	✓	LKP_NULL_ID


Lookup

General | Selection Rules | Columns

Columns:

	Name	Expression
1	std_zip	lookup.key
*		

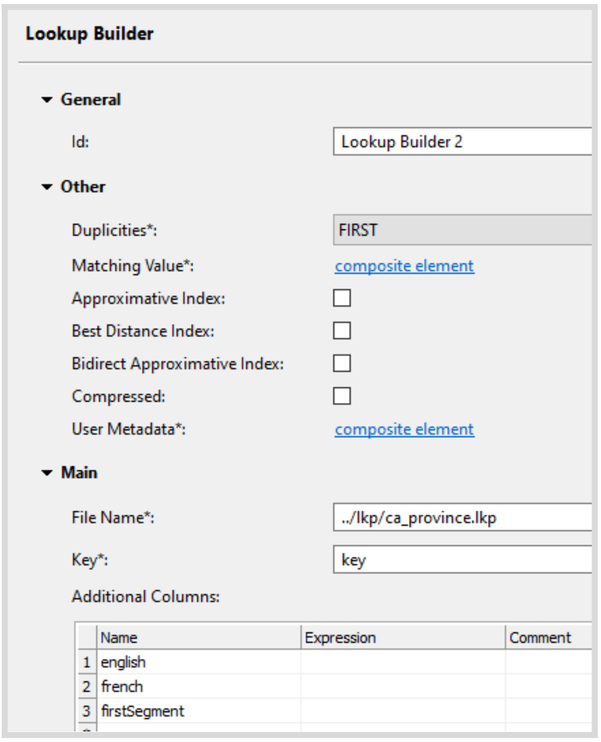
HINT

Do the explanation codes look good? Can you think of better prefixes to distinguish which lookup generated which explanation code?

2.4. Lookup builder for translation

Time to make additional use of the lookup files. We will be translating the value from the province code (**src_province**) into the full name of the province in two languages - English and French.

- Go back to your **address_lookup_builder.plan**.
- Add another file - **ca_province.csv** as a data source. (Yes, ONE Desktop plans can have multiple data streams which will execute in parallel!)
- Fill in the configuration of the **Lookup Builder** just like you did in section 2.2:
  - Set the ***File Name*** for the output as **data\ext\lkp\ca_province.lkp**
  - This time add the rest of the flow's columns to the ***Additional Columns*** list.
  - For this case there is no need to configure anything in ***Matching Values***.



Lookup Builder

General

Id:	Lookup Builder 2

Other

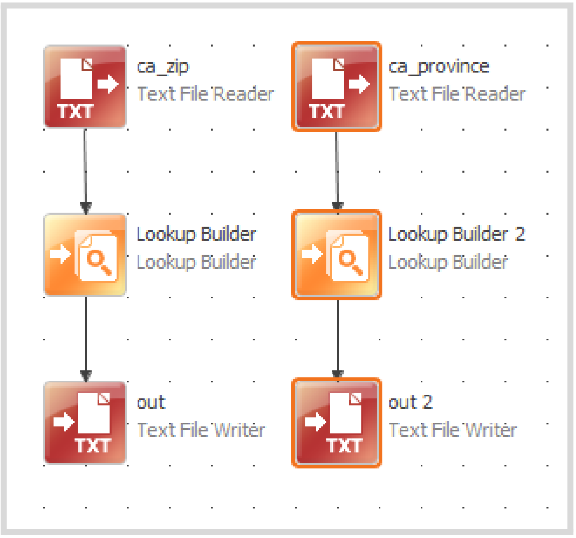
Duplicities*:	FIRST
Matching Value*:	composite element
Approximative Index:	☐
Best Distance Index:	☐
Bidirect Approximative Index:	☐
Compressed:	☐
User Metadata*:	composite element

Main

File Name*:	../lkp/ca_province.lkp
Key*:	key

Additional Columns:

	Name	Expression	Comment
1	english		
2	french		
3	firstSegment		



- **Run** the plan to generate the second lookup file **ca_province.lkp**.

2.5. Using lookup for translation

Can you enhance the address_verify.plan to provide a French translation of the provinces?

TASK

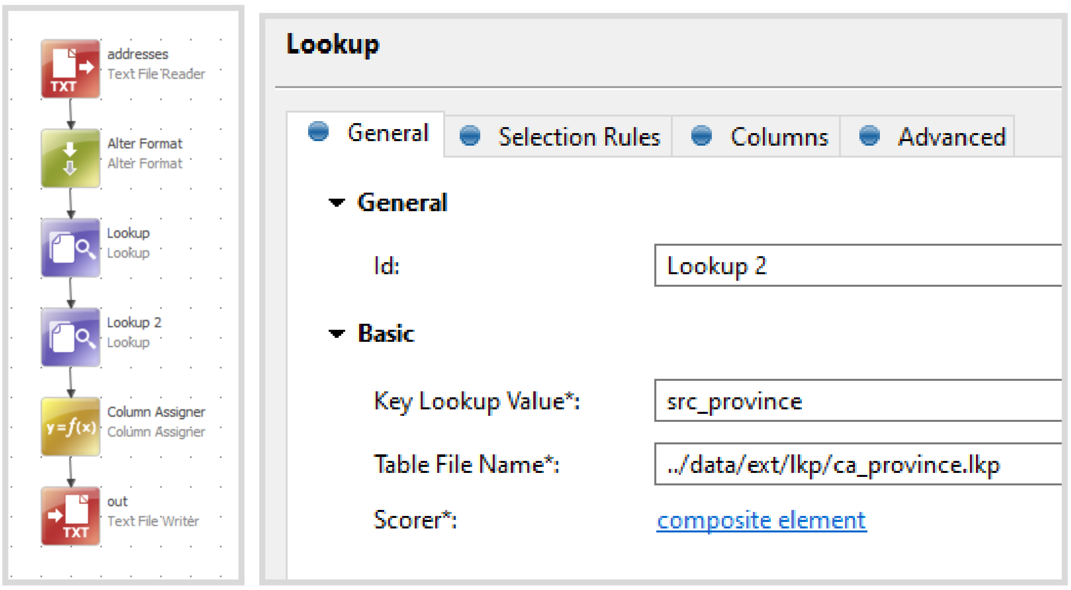
*Create new columns **std_province_en**, **std_province_fr**, **sco_province**, **exp_province**, **sco_total** and **exp_total**.*
*Store the lookup values into **std_province_en** and **std_province_fr**.*
*Store the scoring and explanation into **sco_province** and **exp_province**.*
*Add the scoring up in **sco_total** and concatenate the explanations into **exp_total**.*

HINT

You can save time by copying and pasting steps and changing the configuration for the new step!

HINT

The + sign adds numbers for INTEGER data types. It will concatenate when used for STRING data types!



General

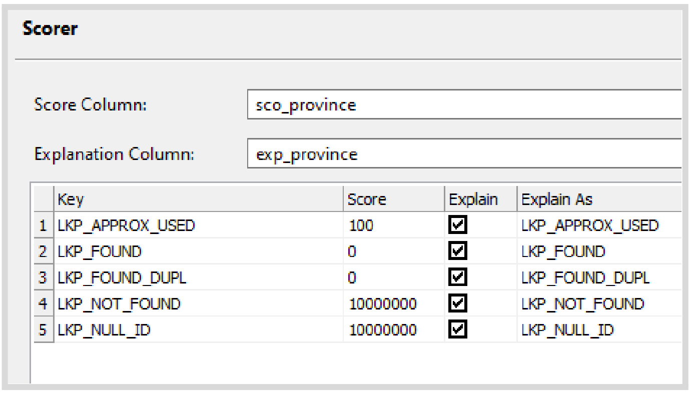
Id: Lookup 2

Basic

Key Lookup Value*: src_province

Table File Name*: ../data/ext/lkp/ca_province.lkp

Scorer*: composite element

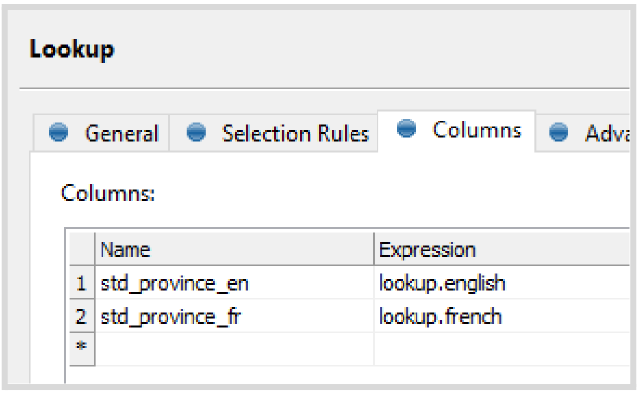


Scorer

Score Column: sco_province

Explanation Column: exp_province

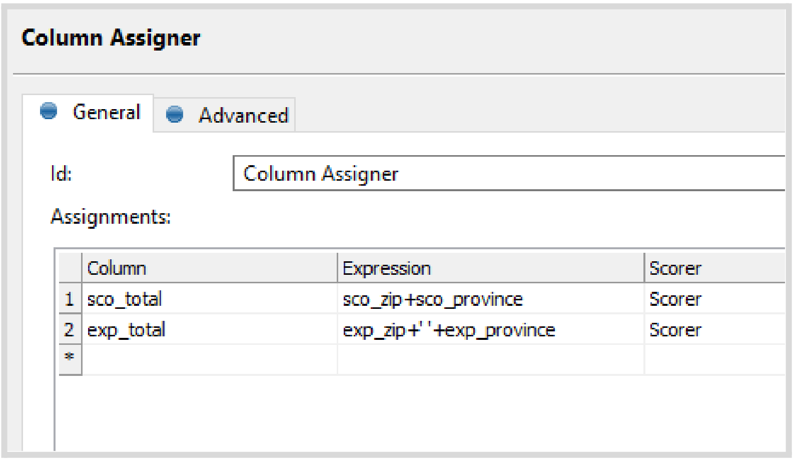
	Key	Score	Explain	Explain As
1	LKP_APPROX_USED	100	☑	LKP_APPROX_USED
2	LKP_FOUND	0	☑	LKP_FOUND
3	LKP_FOUND_DUPL	0	☑	LKP_FOUND_DUPL
4	LKP_NOT_FOUND	10000000	☑	LKP_NOT_FOUND
5	LKP_NULL_ID	10000000	☑	LKP_NULL_ID



Lookup

Columns:

	Name	Expression
1	std_province_en	lookup.english
2	std_province_fr	lookup.french
*		



Column Assigner

Id: Column Assigner

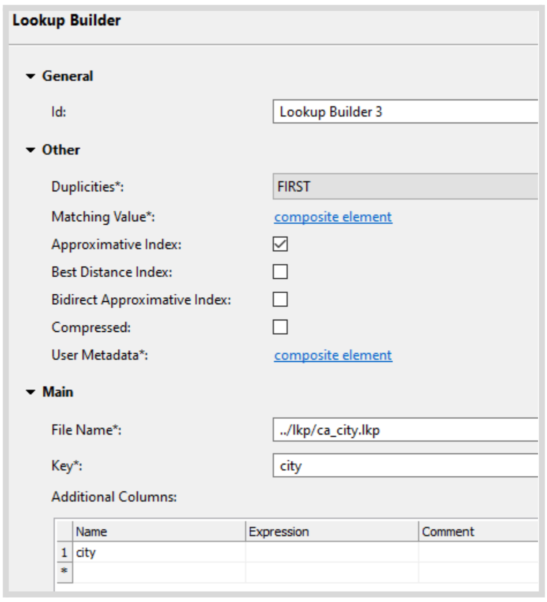
Assignments:

	Column	Expression	Scorer
1	sco_total	sco_zip+sco_province	Scorer
2	exp_total	exp_zip+' '+exp_province	Scorer
*			

2.6. Approximative lookup

Until now, we have been comparing our values for an exact match. This time we will allow minor differences between the lookup key and a source value and still consider them a successful match.

- Go back to **05_address_lookup_builder.plan**.
- Add another file **ca_city.csv** as a data source.
- Use a new **Lookup Builder** to build a **ca_city.lkp** file.
- Choose the **Approximative Index** option when configuring its properties.
- In the **Matching Value** composite element: enable **Remove Accents**, **Squeeze whitespaces**, **Do Upper Case,** and support **letters** only.



Lookup Builder

▾ General

Id:	Lookup Builder 3

▾ Other

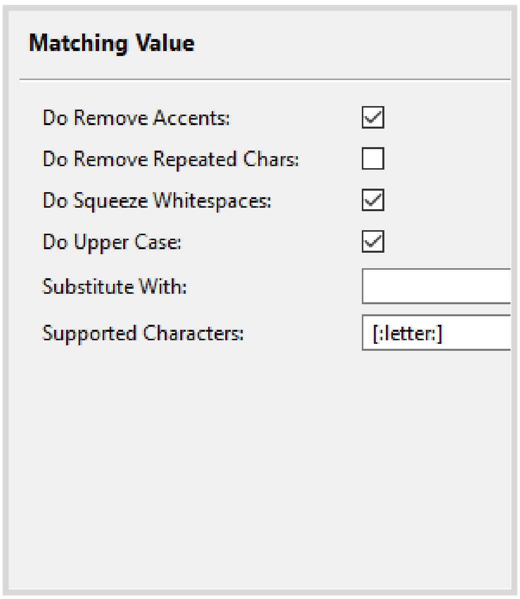
Duplicities*:	FIRST
Matching Value*:	composite element
Approximative Index:	[✓]
Best Distance Index:	[ ]
Bidirect Approximative Index:	[ ]
Compressed:	[ ]
User Metadata*:	composite element

▾ Main

File Name*:	../lkp/ca_city.lkp
Key*:	city

Additional Columns:

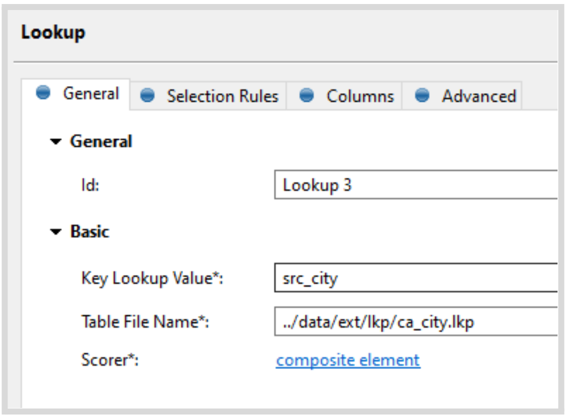
	Name	Expression	Comment
1	city		
*			



Matching Value

Do Remove Accents:	[✓]
Do Remove Repeated Chars:	[ ]
Do Squeeze Whitespaces:	[✓]
Do Upper Case:	[✓]
Substitute With:	
Supported Characters:	[:letter:]

- **Run** the plan to generate your new lookup file.
- Go back to **05_address_verify.plan**.
- Enhance it like the last step:
  - Add columns **std_city**, **sco_city** and **exp_city**
  - Update **sco_total** and **exp_total** calculations
- Add a **Lookup** step to use the **ca_city.lkp** file. In the ***Selection Rules*** tab, enter the **Max Difference** value to be '**2**'. This allows a matching based on no more than 2 characters difference.


Lookup

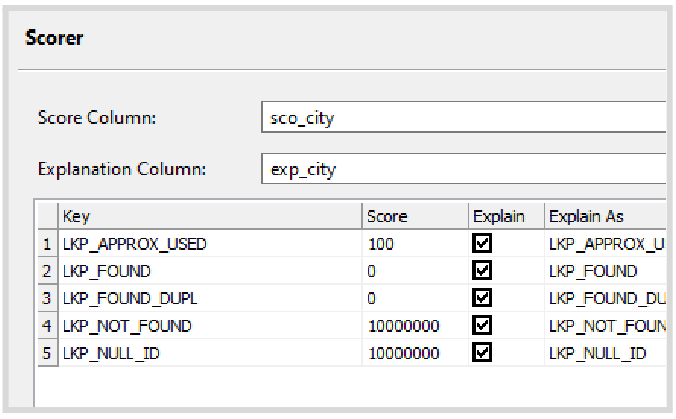
General | Selection Rules | Columns | Advanced

General

Id:	Lookup 3

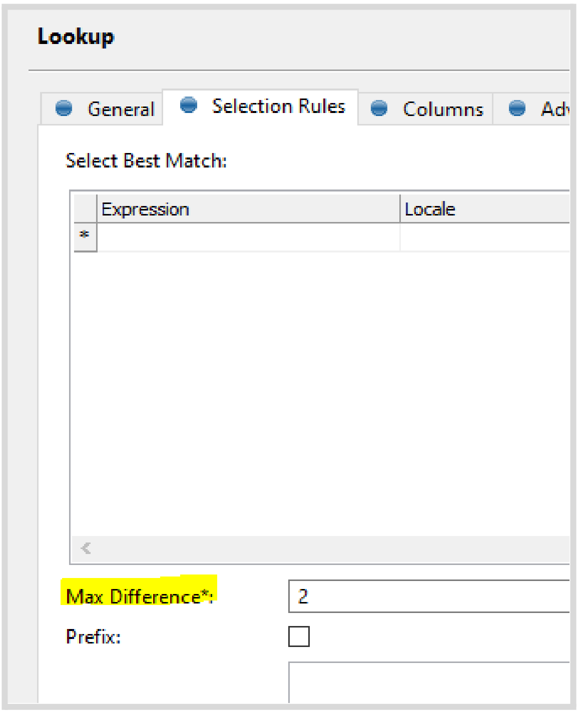
Basic

Key Lookup Value*:	src_city
Table File Name*:	../data/ext/lkp/ca_city.lkp
Scorer*:	composite element


Scorer

Score Column:	sco_city
Explanation Column:	exp_city

	Key	Score	Explain	Explain As
1	LKP_APPROX_USED	100	✓	LKP_APPROX_U
2	LKP_FOUND	0	✓	LKP_FOUND
3	LKP_FOUND_DUPL	0	✓	LKP_FOUND_DU
4	LKP_NOT_FOUND	10000000	✓	LKP_NOT_FOUN
5	LKP_NULL_ID	10000000	✓	LKP_NULL_ID

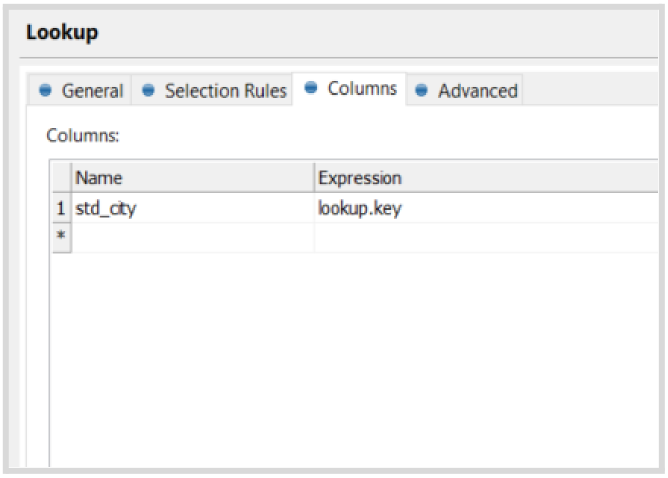

Lookup

General | Selection Rules | Columns | Adv

Select Best Match:

	Expression	Locale
*		

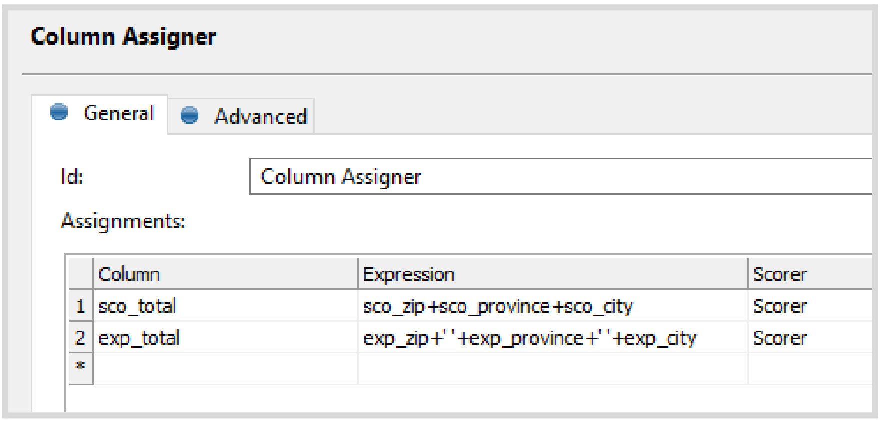
Max Difference*:	2
Prefix:	☐


Lookup

General | Selection Rules | Columns | Advanced

Columns:

	Name	Expression
1	std_city	lookup.key
*		


Column Assigner

General | Advanced

Id:	Column Assigner

Assignments:

	Column	Expression	Scorer
1	sco_total	sco_zip+sco_province+sco_city	Scorer
2	exp_total	exp_zip+' '+exp_province+' '+exp_city	Scorer
*			

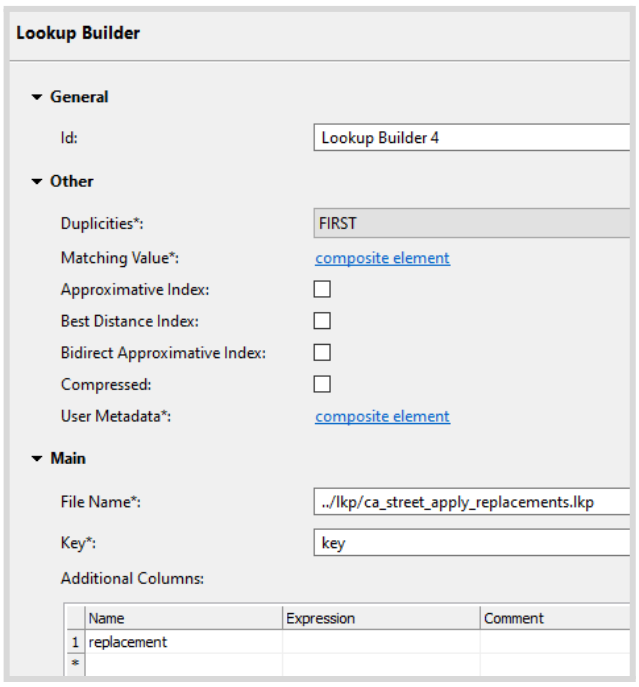
› **Run** the plan

See how some misspelled instances of Toronto have been corrected. (e.g. '***TORONNTO'*** and '***Tronto'***)

2.7. Using lookups to apply replacements.

The **Apply Replacements** step uses lookups to effectively replace values. Its functionality is very similar to the **Transliterate** step we had used before.

- Go back to your **05_address_lookup_builder.plan**.
- Add another file, **ca_street_apply_replacements.csv** as a data source.
- Use a new **Lookup Builder** step to build a **ca_street_apply_replacements.lkp** file. Make sure that you **Squeeze whitespaces**, **Do Upper Case** and accept ***letters*** and ***whitespace***.



Lookup Builder

General

Id: Lookup Builder 4

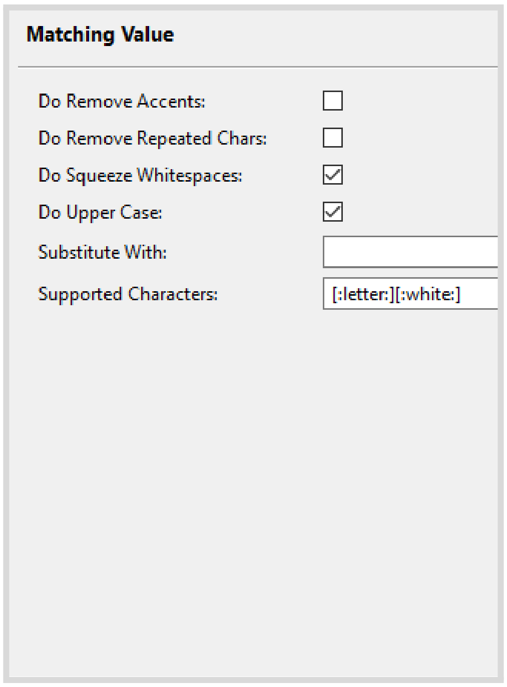
Other

Duplicities*: FIRST
Matching Value*: composite element
Approximative Index: ☐
Best Distance Index: ☐
Bidirect Approximative Index: ☐
Compressed: ☐
User Metadata*: composite element

Main

File Name*: ../lkp/ca_street_apply_replacements.lkp
Key*: key
Additional Columns:

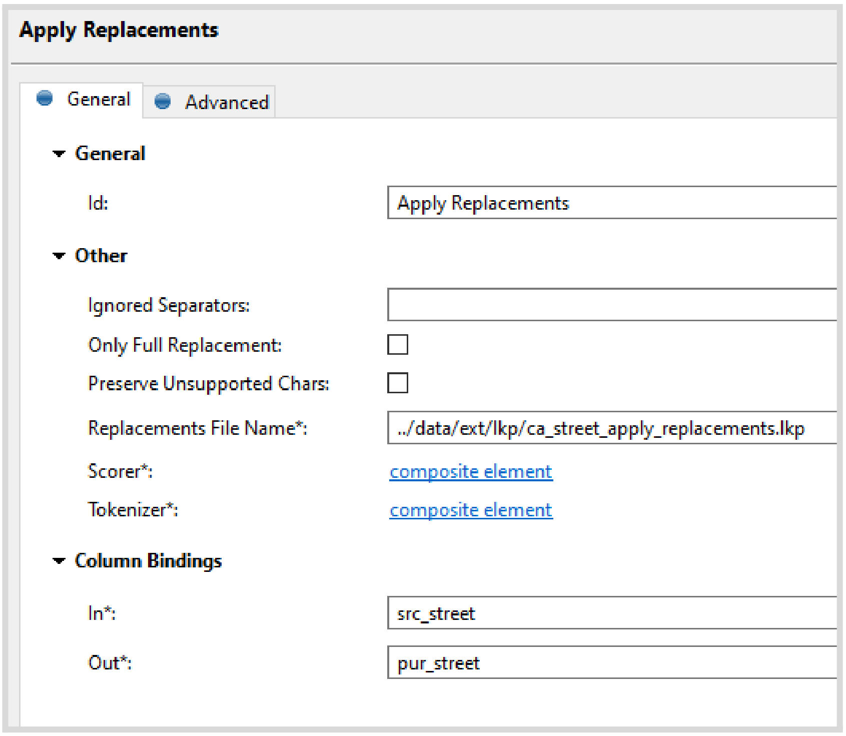
	Name	Expression	Comment
1	replacement		
*			



Matching Value

Do Remove Accents: ☐
Do Remove Repeated Chars: ☐
Do Squeeze Whitespaces: ✓
Do Upper Case: ✓
Substitute With:
Supported Characters: [:letter:][:white:]

- **Run** the plan to generate your lookup file.
- Go back to **05_address_verify.plan** and add new **pur_street** column (STRING).
- Add an **Apply Replacements** step and use your new **ca_street_apply_replacements.lkp** file with input as **src_street** and output as **pur_street**.


Apply Replacements

General | Advanced

General

Id: Apply Replacements

Other

Ignored Separators:

Only Full Replacement: ☐

Preserve Unsupported Chars: ☐

Replacements File Name*: ../data/ext/lkp/ca_street_apply_replacements.lkp

Scorer*: composite element

Tokenizer*: composite element

Column Bindings

In*: src_street

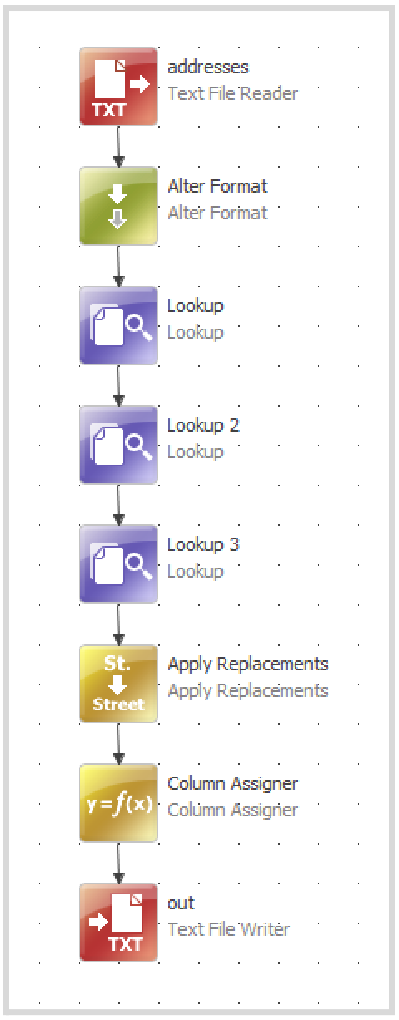
Out*: pur_street

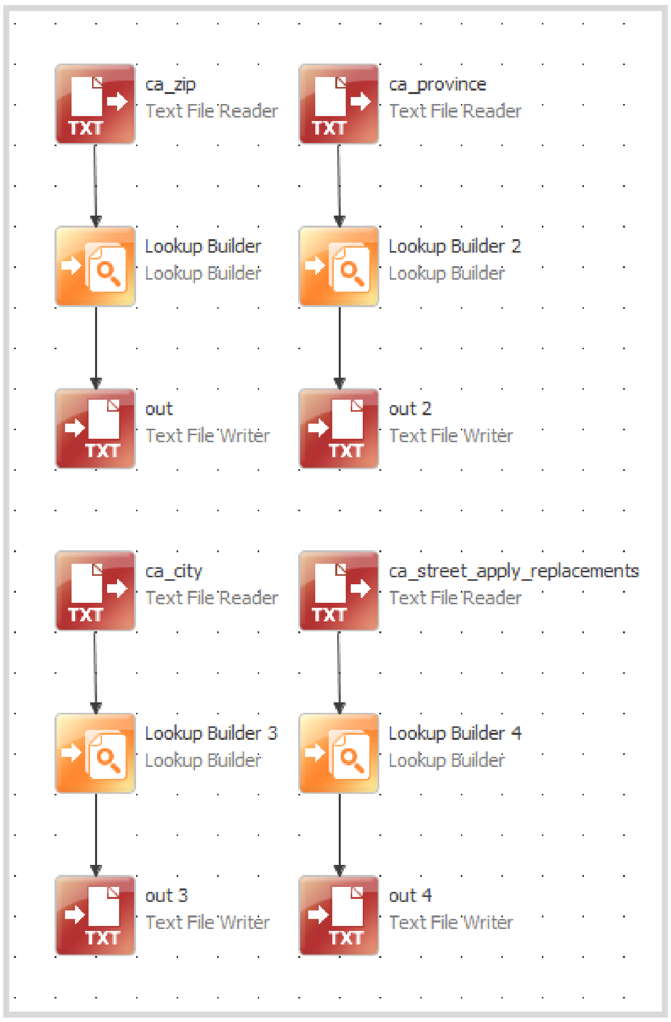
› **Run** the plan.

See that where there are misspellings, “***Leslei***” has been corrected to “***Leslie***” according to the lookup. Because we have not used the **Approximative** search function, see that it is requiring an exact match before it will replace.

CAUTION *Be very careful. You do not want to by mistake replace something that was correct in the first place!*

Your finished **05_address_verify.plan** and **05_address_lookup_builder.plan** should look something like this:


addresses
Text File Reader
Alter Format
Alter Format
Lookup
Lookup
Lookup 2
Lookup
Lookup 3
Lookup
Apply Replacements
Apply Replacements
Column Assigner
Column Assigner
out
Text File Writer


ca_zip
Text File Reader
ca_province
Text File Reader
Lookup Builder
Lookup Builder
Lookup Builder 2
Lookup Builder
out
Text File Writer
out 2
Text File Writer
ca_city
Text File Reader
ca_street_apply_replacements
Text File Reader
Lookup Builder 3
Lookup Builder
Lookup Builder 4
Lookup Builder
out 3
Text File Writer
out 4
Text File Writer

3. Conclusion

We have come to the end of this workshop. We have used lookups for checking inputs against a list of valid values and to translate a code into related attributes. In addition to that, we have used a lookup to cleanse spelling and use approximative matching.

As you have just experienced, using a lookup step is more powerful than a join in terms of the scoring and explanation functionality. It is more streamlined than configuring individual join steps or replace functions.