Paper 4795-2020

SAS® DataFlux™ Matchcodes: a Practical Use at the Canada Revenue Agency for Tax Filing Status

Jason A. Oliver, Senior Compliance Analyst, Canada Revenue Agency

ABSTRACT

To streamline the "data analytics supply chain," the Canada Revenue Agency (CRA) in partnership with SAS® has established an automated matchcodes program in SAS® DataFlux™ software, which takes collected web domain records and matches them on existing taxpayer databases in the data lake, using combinations of company or trade name, phone number, and/or postal code. Ultimately, this helps to reveal any non-filers as well as taxpayer risk/audit history, which can be used to derive predictive analytics algorithms, and is much more efficient than manual lookup of company contact info from web pages against the operational taxpayer database. This case study can conceivably be applied in the abstract for multiple types of government functions, such as investigations or law enforcement or national security—but in this presentation, it is from a taxation enforcement perspective.

INTRODUCTION

This paper is produced from the Canada Revenue Agency in the course of efforts to come up with a more accurate and efficient means of matching unknown commercial entities to the CRA taxpayer database to determine their status. This paper will serve as a blueprint of sorts for how anyone – not just in the taxation sector – may be able to perform entity reconciliation using fuzzy matching techniques with SAS® DataFlux™.[1] You will see what data cleansing and fine-tuning techniques are performed in order to realize optimal data quality that will mitigate false positive matches, and return a scored set of reliable corporate entity matches.

For anyone that is familiar with LEAN methodology, you will know about the need to remove non-value added steps in the supply chain or lifecycle. In this case, we are dealing with a "data supply chain", as web records are sourced from web crawling software, but it is very tedious to manually look them up in the taxpayer operational database. Thus SAS® becomes a formidable solution in accelerating our data supply chain, so that we may auto-match these sourced records to taxpayer identity view objects on our data lake. This minimizes the time between inception of the data and taking of action on anomalies.

ENTITY RESOLUTION & THE SAS-DF ADVANTAGE

At the CRA, we have a need to determine whether businesses operating online and transacting with Canadians are, in fact, registered with the CRA to pay taxes. We have an additional need to scrutinize on the audit and risk history of commercial taxpayers in a given industry sector, which becomes the scope of a "web crawl". The web crawl is an

[1] NOTE that we also refer to SAS DataFlux™ as Data Quality™.

upstream stage from SAS® DataFlux® matching; it entails using special software to send a complex programmed query out to the web and return a list of relevant web sites.

However, it is patently obvious that commercial entities are not going to transparently post their Taxpayer ID (TPID), so it is incumbent on us in the CRA to determine if they are

a) registered with us and for those who are,

b) determine if they have [recent] filing history, and if so,

c) what sort of risk and audit history do they have? (And how does it contrast with the general population?)

As it is incredibly tedious to manually match web domain owner records to our operational taxpayer database, we have realized an excellent catalyst for our data supply chain in the form of SAS® DataFlux™. However, in so doing, we had to rely on a data lake view object as opposed to the operational database, as clearly doing so would cause congestion and impair auditors' work.

Here is a basic overview of our "data supply chain" architecture:

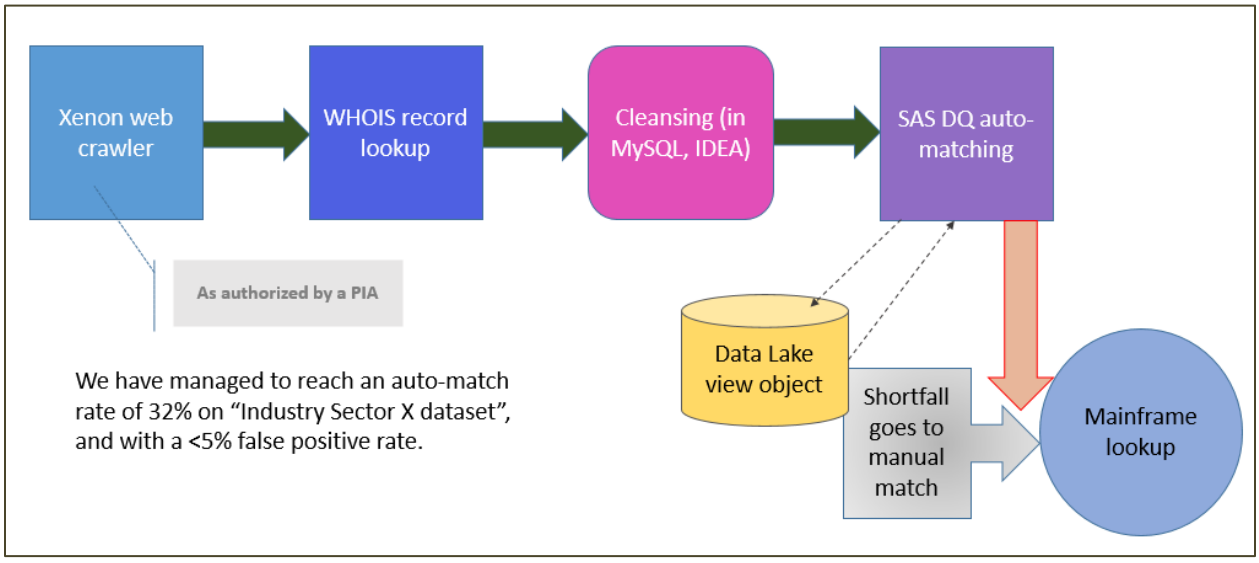


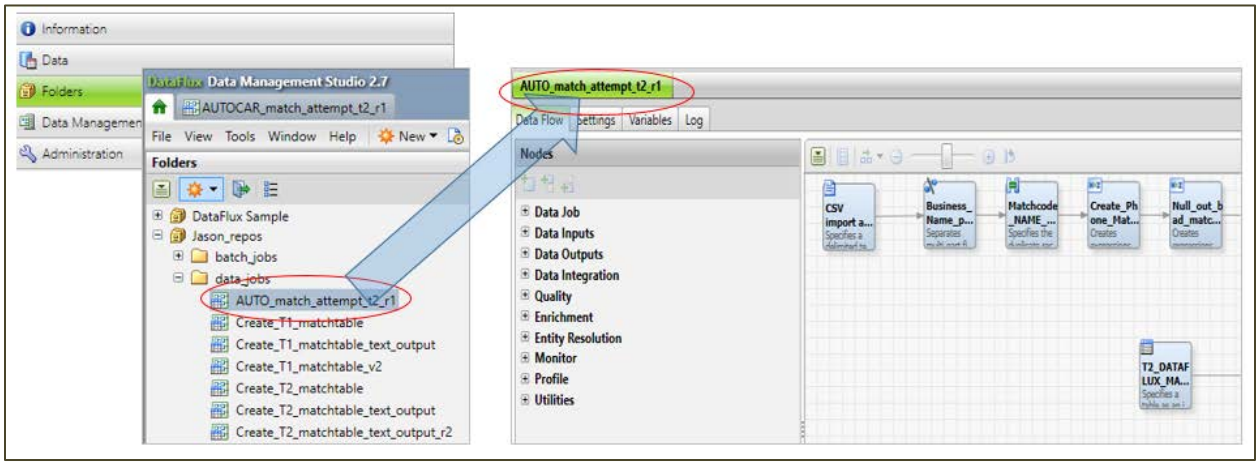
Figure 1. CRA Data Supply Chain for auto-matching

SAS® DATAFLUX™ NAVIGATION

When we first set up our process flow (or canvas) in DataFlux™, we need to go into the Folders → data_jobs section and initiate two branches:

- one, called the "left-hand table" or upper portion, for what I call the "candidate file", that is, the industry-specific web-crawled records of interest;
- two, called the "right-hand table" or lower portion, for ingesting the data lake view object which contains taxpayer profile records [including the TPID that we need].

From the menu you see under data_jobs, we refer to the first auto-match canvas. Note that we may always use "Save As" to create another instance of this, and all we need to do is replace the left-hand source table with the new one of interest.



Display 1. Auto-matching job selection & load

Below the AUTO_match_attempt_t2_r1 in the list at left, you will see a list of experimentation creation instances of matchcode tables, for the T1 sector (which is small business such as sole proprietors and partnerships) and the T2 sector (which is corporate).

When you look at the wider portion on the right, it is your process flow or canvas where you do your data mutations and "prettying up" for match suitability.

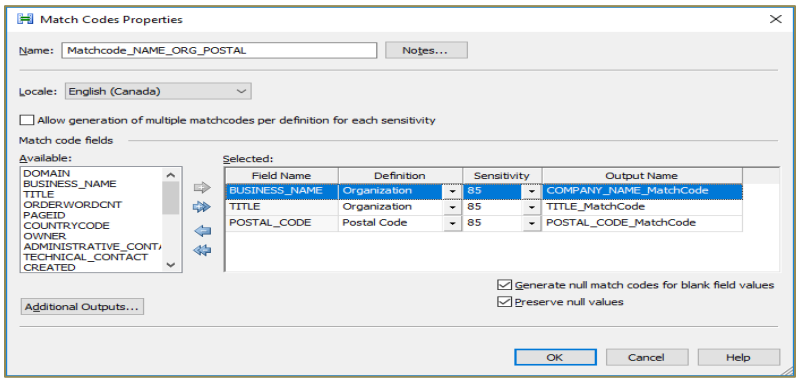
HITTING THE DATAFLUX™ **CANVAS**: THE MAIN EVENT

This is where the rubber hits the pavement.

Calibrate Wisely: Matchcode Sensitivity

When we were initially setting up our candidate dataset for ingestion into the DataFlux™ canvas, we had to make several attempts with the optimal sensitivity level when setting up a defined object for our Company Name (called "BUSINESS_NAME"). If we set the sensitivity too close to 100%, then fuzzy matching would be rendered impossible and we'd be looking at literal matching; if we put it at say 70-75%, we'd be looking at a slew of false positives.

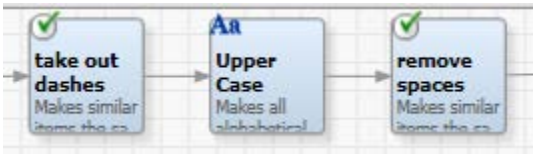
Thus we found our ideal sensitivity rate at 85%. We repeated this rate for the TITLE (which additionally represents the organization name), and the POSTAL_CODE fields.



Display 2. Setup of match code output objects with sensitivity percentage

Further to this, we need to consider "stripping down" the Company Name matchcode object to ignore extraneous attributes such as dashes, spaces, and ignoring capitalization (astute data scientists who have studied text analytics will be familiar with the term *tokenization*, which is exactly what this is).

This is covered in these three nodes in the upper (left-side) stream:



Display 3. Nodes to remove extraneous attributes from Company Name

Matching by Length, and the Curse of "Commonyms"

When we set up our match conditions, we also had to specify what would be a suitable length. Clearly, we have to balance between a "lightweight" and efficient model, and one that is too onerous. Thus we compromised at a match-string condition of the first 14 characters, and another at the first 5 characters. Downstream, I have created a scoring formula that gives a score to the observation based on matchcode combination strength, and so any matches on a 14-character company would score higher than those on a 5-charcter company match (more on that later).

```
string(14) Company_Name_14Char

Company_Name_14Char = left(`Company_Name_Upper_Stnd`,14)
if isblank(Company_Name_14Char) then Company_Name_14Char = null

string(5) Company_Name_5Char

Company_Name_5Char = left(`Company_Name_Upper_Stnd`,5)
if isblank(Company_Name_5Char) then Company_Name_5Char = null
```

Display 4. Input of code to narrow Company Name match length

One of the big stumbling blocks of our automated match codes is dealing with what I call "commonyms". These impede the ability to match by length of a Company Name value where the Company Name may start with something like "Alberta Association of…" or "Mississauga Academy | Centre | Institute of...etc." It gets even more challenging when processing businesses from the province of Quebec where French is the first language and so the words are inverted, e.g. the general business *type* comes first within the general company name, and *then* the business name proper. But this is just something we have to "live with", as we have not found a way in which we can selectively scrutinize right-to-left.

Supporting Parameters and Match Flexibility

In addition to Business Name, we use the phone number (both the 10-digit and the 7-digit instance) and the Postal Code. We don't use the Address field, as it can be very onerous and

inconsistent when dealing with corporate entities. In other words, it would be a great deal of effort and refinement contrasted with little gain in match rate.

We injected a node into our left-hand (upper) stream to separate out the phone number, so that we can match it both on the 10-digit instance as a "stand-alone" condition, or as a 7-digit instance in conjunct with the Company Name (or Trade Name), as we will see later on in our join node when we bring it all together.

Display 5. Emphasizing the PHONE field processing in the stream

The code from the first node [Create_Phone_Matchcode] is a pre-processing expression, as follows:

```
integer phone7_matchcode
if phone_raw <> "NA" then
phone7_matchcode = right(trim(`PHONE_RAW`), 7)

integer phone10_matchcode
if phone_raw <> "NA" then
phone10_matchcode = right(trim(`PHONE_RAW`), 10)
```

Notice how we inserted a "safety switch" to filter out NAs. But we go one further than this in the next node [called Null_out_bad_matchcodes], which also covers business name and titles:

```
if inlist(`phone7_matchcode`, "0000000", "1111111", "5555555", "9999999")
or isblank(`phone7_matchcode`) then `phone7_matchcode`=null

if mid(`TITLE_MatchCode`,8,1) == '$' then `TITLE_MatchCode` = null

if mid(`COMPANY_NAME_MatchCode`,8,1) == '$' then `COMPANY_NAME_MatchCode` =
null
```

This is truly cleansing for consistency! We are not going to do so well with "Hollywood-ized numbers" like your generic 555-5555.

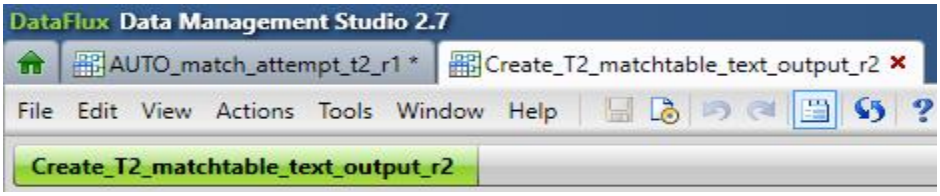
As for the POSTAL CODE, we do not need any special formatting nodes for that. Our next area of focus is for the right-side table, that of the matchcodes object from the data lake.

Importing Matchcodes Lookup View to the Canvas

We have a view object (not a table per se) in our CRA Data Lake, for the lookup of corporate (T2) entities for identification purposes only, i.e. this doesn't contain any financial variables or classification variables, just identity and coordinates.

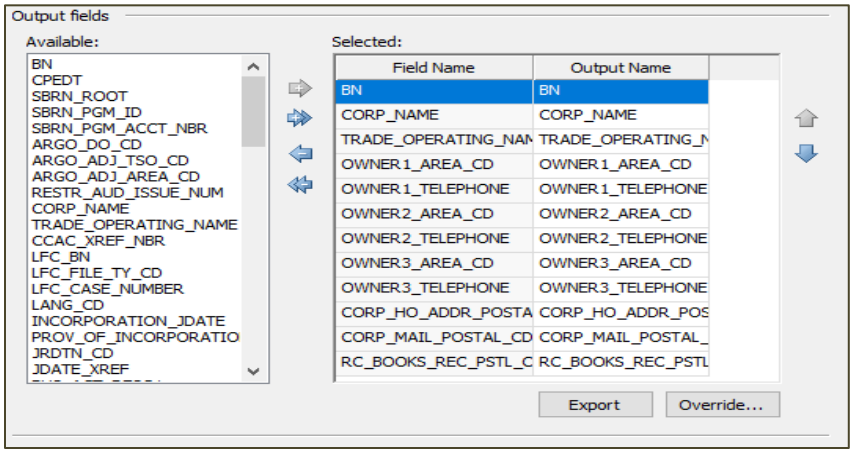
We used an ODBC connection to extract this data lake view object and bring it into DataFlux™, so that we can do the necessary refinements. However, before bringing it directly into the candidate stream at hand (i.e. the one with the left-side stream), we had to bring it into its own canvas to perform preliminary adjustments.

This workspace is called "Create_T2_matchtable_text_output_r2".



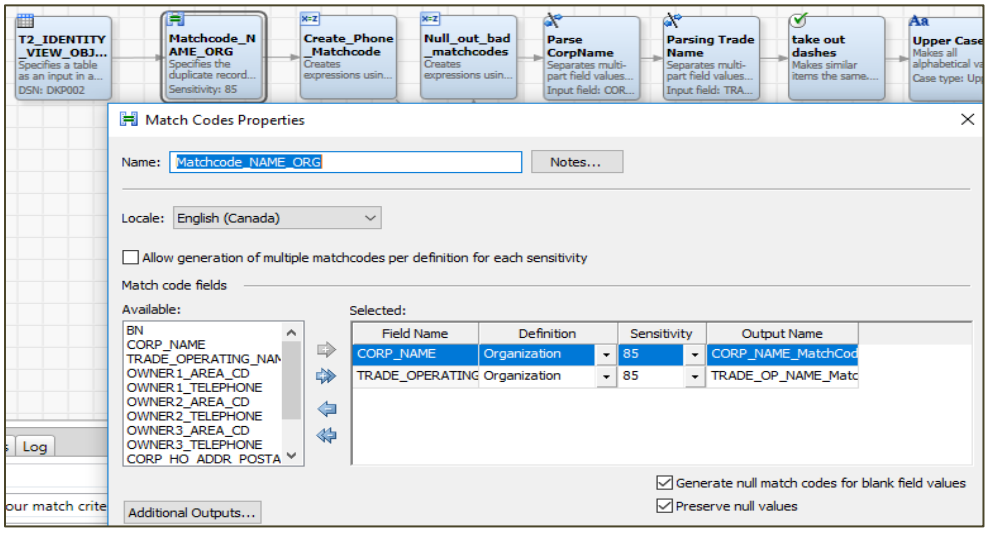
Display 6. Showing the open file tab for the T2 matchcodes object

We didn't use the full range of fields from this view object, but only those that were absolutely necessary for our matchcodes algorithm. These totaled 13. Note that there were three telephone field types (with corresponding area codes), as well as three postal code field types (depending on HQ/HO, mailing address, and bookkeeping – see bottom three fields of image below).



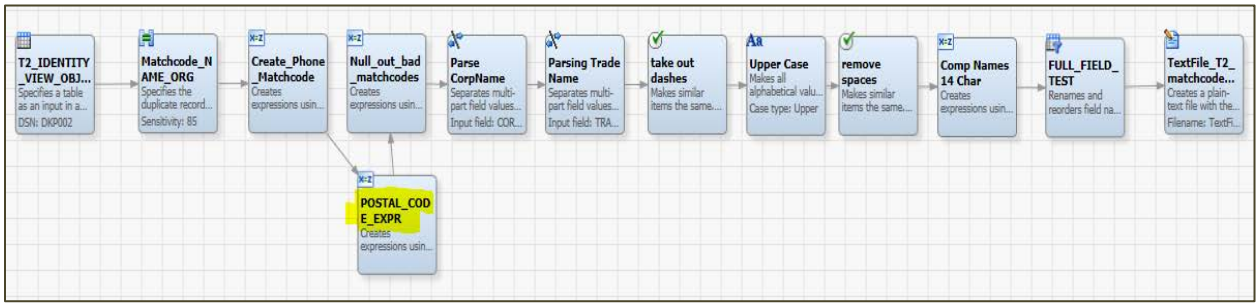
Display 7. The selection of output fields for the T2 matchcodes object

For the first post-import node in the sub-stream of the view object, we need to establish the sensitivity rating of the tokenized fields for CORP_NAME and TRADE_OPERATING_NAME. Once again, these are at 85% to be consistent with the left-side candidate data stream.

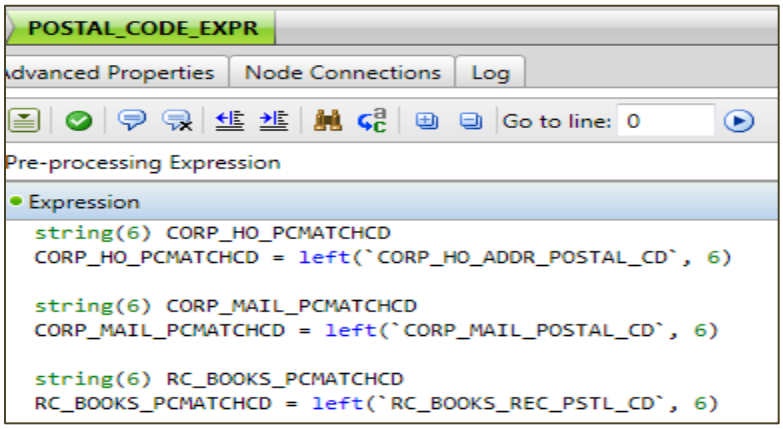


Display 8. Establishing sensitivity level for the matchcodes object fields

For the remaining fields in the sub-stream for our view object, they are very similar to the parsing work we'd done for our candidate dataset earlier. We ought to focus, however, on the POSTAL_CODE_EXPR node in particular, as this is expected to make a great difference downstream.



Display 9. Full canvas, T2 matchcodes view object parsing



Display 10. Pre-processing Expression for POSTAL_CODE_EXPR

In all three Postal Code instances, we need to ensure that it's only extracting 6 characters. We found through trial and error that it was mistakenly pulling leading spaces or characters so we injected this corrective code.

Later on, as we'll see when applying score to the strength of matchcodes, some of these postal code instances give more points when they're "hit on" [in conjunct with Corporate Name].

Now we exit the sub-stream for the T2 matchcodes object, and come back into the main canvas, which is where we still have four outstanding nodes to run on our T2 object that we obtained from the output of our sub-stream.

Display 11. Return to main canvas, lower branch for matchcodes object nodes

In the upper-right of the above image, you can see the corner of the JOIN node, which is where we next turn our attention.

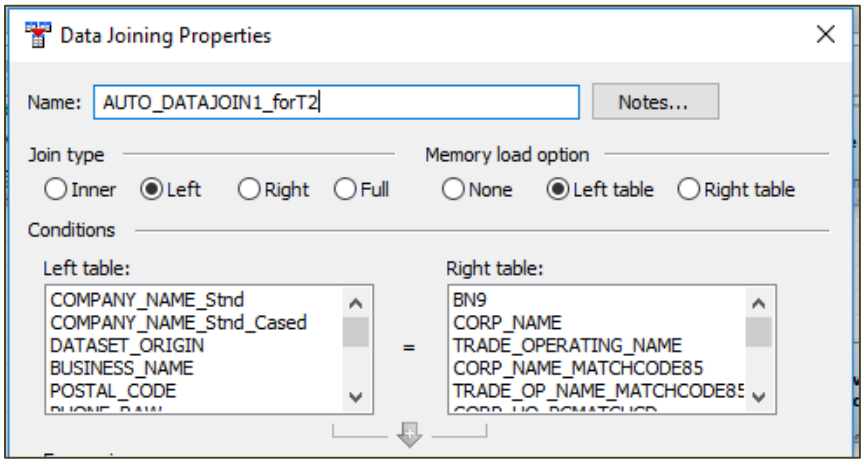
Otherwise, the nodes in this branch cover:

- the tokenization of the CORP. NAME entity
- the tokenization of the TRADE OPERATING NAME entity
- the derivation of a 7-digit phone number from the 10-digit instance
- the derivation of a 5-character "quick match" instance of the Corporation Name and Trade Operating Name.

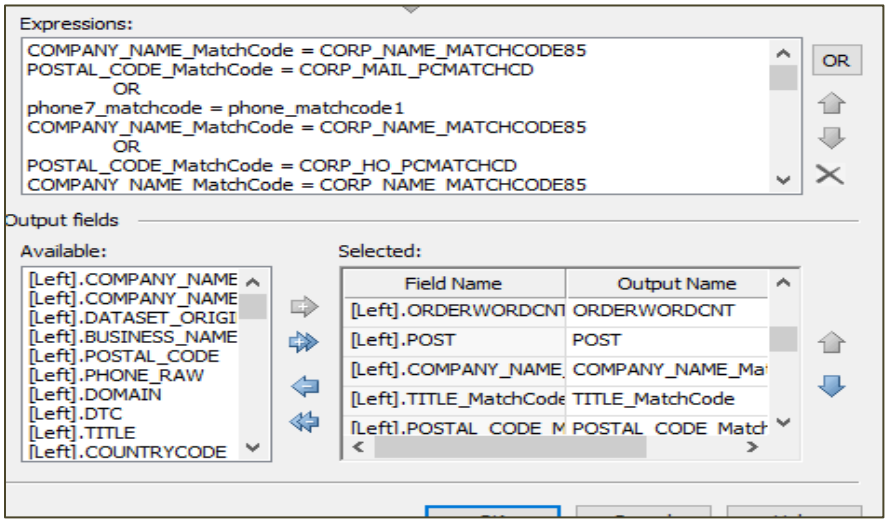
As a testing and troubleshooting aside, you can press 'F6' on any given node to preview its contents, and it will show you the contents of the first 'n' rows (10 by default), just to give that extra assurance of full functionality. *It is a very wise idea to do this before your JOIN.*

The JOIN: Bringing it all Together

In the JOIN node, you are specifying not only the fields from the left-side candidate table and the right-side view object [for lookup], but you are specifying which of these fields will be part of the matchcode combos. The combos may seem somewhat repetitive, but this is because there are three phone fields, three postal code fields, and two business name fields (i.e. Company Name, and Trade Operating Name). We want to leave no stone unturned!



Display 12a. Data Joining Properties, upper portion, join type & conditions



Display 12b. Data Joining Properties, lower portion, matchcode combos and output fields

As you can see from the above portion, we use matchcode combos (or mcc#, where # is from 1 to 17), containing combinations of Company Name, Postal Code type, and Phone Number type. The algorithm will keep running even if it encounters a "hit" on a record, which could mean multiple hits may occur. This brings us to our next section, on scoring.

SCORING FOR SUCCESS

After we have entered our 17 matchcode combos, we need to assign them suitable scores, based on how reliable they are. As you might expect, those matchcode combo occurrences that are anticipated to be less common, will be accorded a higher score.

Here is a snapshot excerpt of the matchodes scoring configuration.

```
integer mcc4
mcc4 = 0
if phone7_matchcode == phone_matchcode3 AND COMPANY_NAME_MatchCode ==
CORP_NAME_MATCHCODE85
        then mcc4 = 4

integer mcc5
mcc5 = 0
if POSTAL_CODE_MatchCode == CORP_HO_PCMATCHCD AND
COMPANY_NAME_MatchCode == TRADE_OP_NAME_MATCHCODE85
        then mcc5 = 7

integer mcc6
mcc6 = 0
```

```
if phone7_matchcode == phone_matchcode1 AND COMPANY_NAME_MatchCode ==
TRADE_OP_NAME_MATCHCODE85
        then mcc6 = 7

integer mcc7
mcc7 = 0
if Company_Name_PARSED == Corp_Name_PARSED
        then mcc7 = 3
```

You can tell that that certain conditions legitimately yield more points. The more points accorded to a given entity, the more confidence we have in the match integrity.

Maximum vs. Summary scores

For each record, the scoring algorithm (as per the post-join nodes) will establish a "max score", which is the highest-occurring value for each of the 17 matchcode combos in that row, and a "summary score", which is the total of all "score hits" in that row. Needless to say, if only one mcc gets a hit, then "summary score" = "max score".

In the sample output file that follows (as an image snapshot from Excel), the summary score is referred to as "score_combiner", and the max. score is referred to as "score_selector".

Z	AA	AG	AH	AI	AJ
score_combiner	score_selector	mcc6	mcc7	mcc8	mcc9
2	1	0	0	0	1
3	3	0	0	3	0
0	0	0	0	0	0
3	3	0	0	3	0
0	0	0	0	0	0
4	3	0	0	3	0
0	0	0	0	0	0
0	0	0	0	0	0
10	7	7	0	0	1
2	1	0	0	0	1
1	1	0	0	0	0
1	1	0	0	0	1
1	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
3	3	0	0	3	0
0	0	0	0	0	0
4	3	0	0	3	0
9	7	7	0	0	1

Display 13. Snapshot cross-reference of scoring output per entity record

Scrutinizing Output

While we have good faith in our automated matchcodes algorithm, it does not give us a free pass for visual quality assurance checks! We still scrutinize the output, in a CSV file (presented as Excel) as far as:

- does the business name returned make sense?
- if not, could it be a subsidiary or affiliate relationship, perhaps?
- are there multiple instances (duplicates) of the same business name [and TPID]?
- which scores are the highest?
- what is the average score [filtering out scores of zero first]?
- given a cursory glance over where the score was rated '0', is there a discernible pattern, i.e. did the left-side (candidate data) Company Name appear too obscure or contain "commonyms"?

These will provide guidance towards next steps, which will either mean refining aspects of the cleansing and matching conditions, or resorting to manual matching in the operational taxpayer database.

One recurring issue we see in particular is where a solitary matchcode of value '1' is picked up. This happens whenever there is a singular match on the 10-digit phone number field of the candidate dataset to one of the three 10-digit phone number fields residing in the matchcodes object. Often, this match hit is questionable, because the phone number may simply represent the administrative office of a mall or a business park (and thus does not belong to the corporate entity per se). So we have to do an extra layer of screening on these solitary '1' valued matches.

CONCLUSION

While it has taken a great deal of painstaking effort, fine-tuning and multiple iterations, I can honestly say that without SAS® DataFlux™ I would have had a much more difficult time attempting to reliably match corporate business web records to a taxpayer identity record. This solution has proven invaluable in performing the "heavy lifting", thus freeing up time to focus on more meaningful pursuits in tax-risk analysis.

As far as improving our match rate beyond the current configuration, I am skeptical that any further refinement of the code or nodes in the DataFlux™ canvas will yield a better combination of true positives vs. false positives (many of you will know about the area under the ROC curve or *Receiver Operating Characteristic* curve, which gauges this ever-present trade-off). Adding and decomposing address parameters, in my opinion, is more likely than not to confound the algorithm.

Where I see opportunities to improve match rate occurs at the inception point: we are in the process of upgrading our WHOIS server to realize a greater variety, volume and velocity of company names, and so this could lead to more match candidates on the left-side table. We have to realize that companies sometimes have aliases, subsidiaries, or affiliates that may go unrecognized by the algorithm today. We are also slated to deploy a change later this year to our PDA object to include the corporate identifying information of businesses based *outside of Canada, yet operating within Canada*. As currently this information is just available in the operational taxpayer database, inaccessible to DataFlux™.

REFERENCES

No references were used in the composition of this material, as it is the first of its kind known to me.

ACKNOWLEDGMENTS

I am grateful for the assistance of Arnold Toporowski, our designated SAS® guru and advisor for the CRA. He spent tireless hours assisting me with tweaking and fine-tuning the code and structure of the nodes in my SAS® DataFlux™ stream.

RECOMMENDED READING

- *Cody's Data Cleaning Techniques Using SAS®,* 3rd ed., by Ron Cody. Copyright © 2017, SAS Institute Inc., Cary, NC, USA.
- *Data Management with SAS®: Special Collection.* Copyright © 2019, SAS Institute Inc., Cary, NC, USA.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jason A. Oliver, Senior Compliance Analyst & Data Scientist
Canada Revenue Agency
Jason.oliver@cra-arc.gc.ca