SAS3560-2016

How to Find Your Perfect Match Using SAS® Data Management

Mary Kathryn Queen, SAS Institute Inc.

ABSTRACT

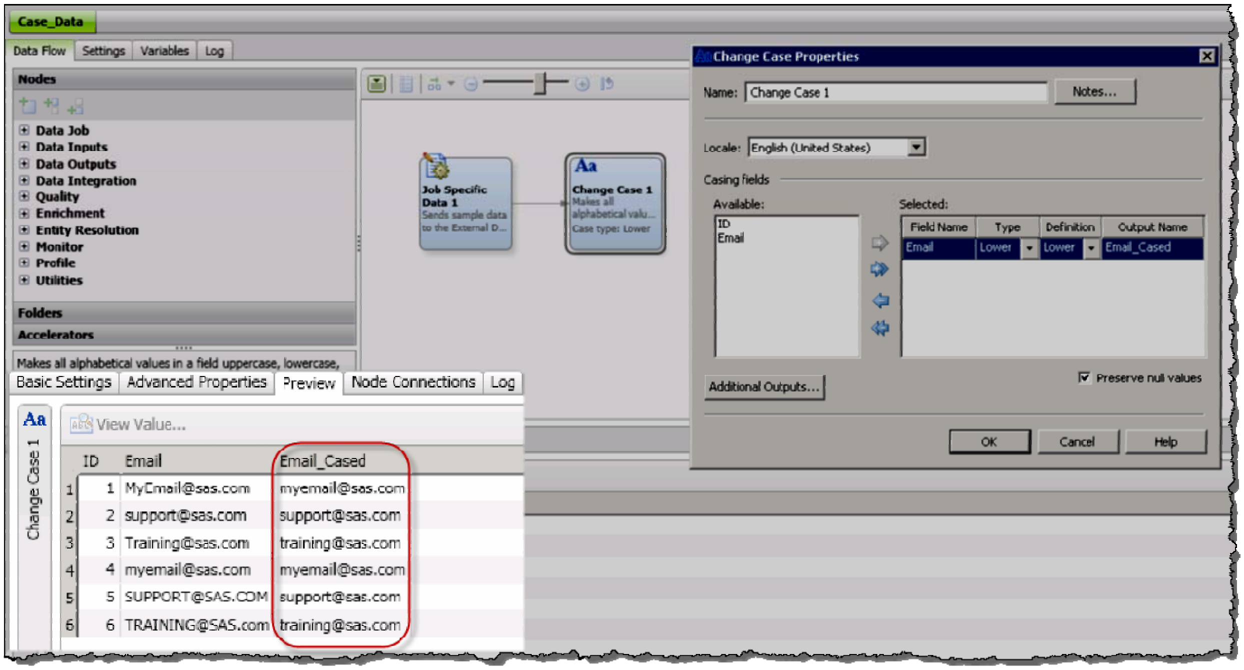
SAS® Data Management is not a dating application. However, as a data analyst, you do strive to find the best matches for your data. Similar to a dating application, when trying to find matches for your data, you need to specify the criteria that constitutes a suitable match. You want to strike a balance between being too stringent with your criteria and under-matching your data and being too loose with your criteria and over-matching your data. This paper highlights various SAS® Data Management matching techniques that you can use to strike the right balance and help your data find its perfect match. As a result, you can improve your data for reporting and analytics purposes.

INTRODUCTION

There are two main components to entity resolution (matching) in SAS® Data Management – *Match codes* and *Clustering*. Match codes use logic from a Match definition in the SAS® Quality Knowledge Base (QKB) to generate a code that can be used to fuzzy match data. Match definitions exist for many data types and locales (Language and Country combinations). You can also create your own custom Match definitions in the Quality Knowledge Base. Clustering provides the ability to match (group) records based on multiple conditions. Your match conditions can use a combination of match codes and exact data values to create the best match for your specific data set. This paper showcases tips and techniques for improving your match results using SAS® Data Management.

STANDARDIZE DATA

Standardizing your data helps improve your match results especially if you are matching exact data values versus fuzzy matching your data. For example, one record might list the email address as *myemail@sas.com* and another record might have *MyEmail@sas.com*. In order to successfully cluster (match) those values you need to standardize the case of the data.



Makes all alphabetical values in a field uppercase, lowercase,

Change Case Properties

Name: Change Case 1

Locale: English (United States)

Casing fields

Available: ID, Email

Selected:

Field Name	Type	Definition	Output Name
Email	Lower	Lower	Email_Cased

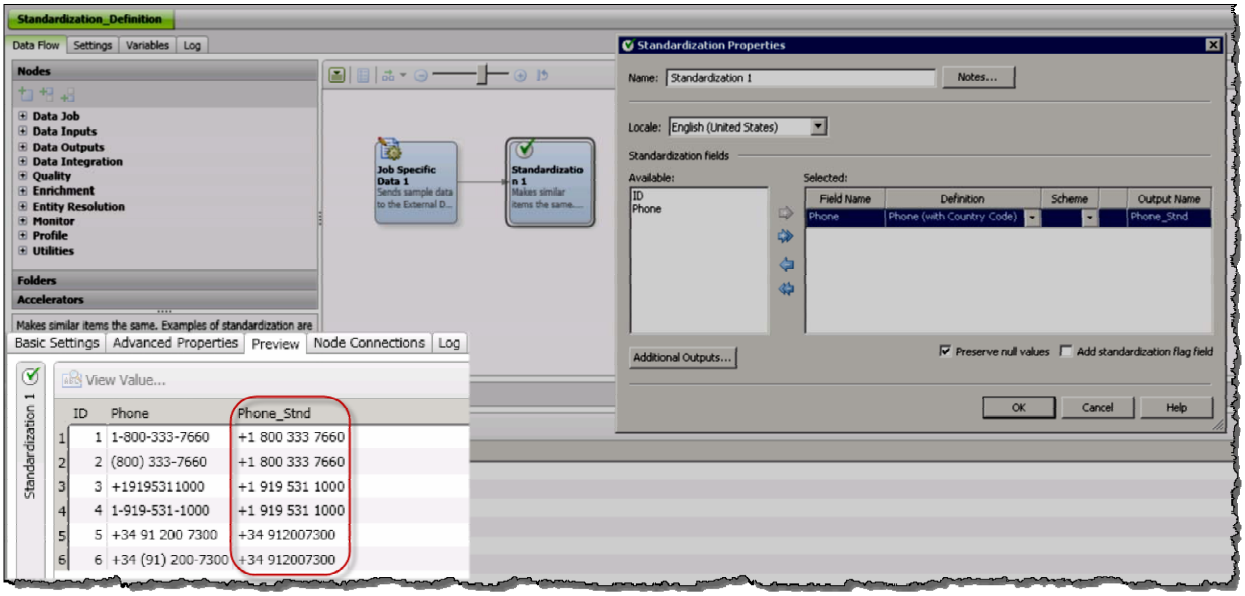
☑ Preserve null values

	ID	Email	Email_Cased
1	1	MyEmail@sas.com	myemail@sas.com
2	2	support@sas.com	support@sas.com
3	3	Training@sas.com	training@sas.com
4	4	myemail@sas.com	myemail@sas.com
5	5	SUPPORT@SAS.COM	support@sas.com
6	6	TRAINING@SAS.com	training@sas.com

Figure 1. Standardize Data – Change Case Results

Note: You can also use the DQCASE function to case your data. Refer to the *SAS® 9.4: Data Quality Server Reference* guide (available at http://support.sas.com/documentation/cdl/en/dqclref/68376/PDF/default/dqclref.pdf) for more information about this function.

There are also out-of-the-box Standardization definitions and schemes delivered as part of the Quality Knowledge Base that you can apply to many different data types. In the figure below the provided *Phone (with Country Code)* Standardization definition is used to standardize the input Phone number data. If you are performing an exact match before standardization, records that have the same numbers as other records, might not match with each other. This is because they used different punctuation. However, after standardization, they would match since the records are now in a common format.



Name: Standardization 1

Locale: English (United States)

Standardization fields

Available: ID, Phone

Selected:

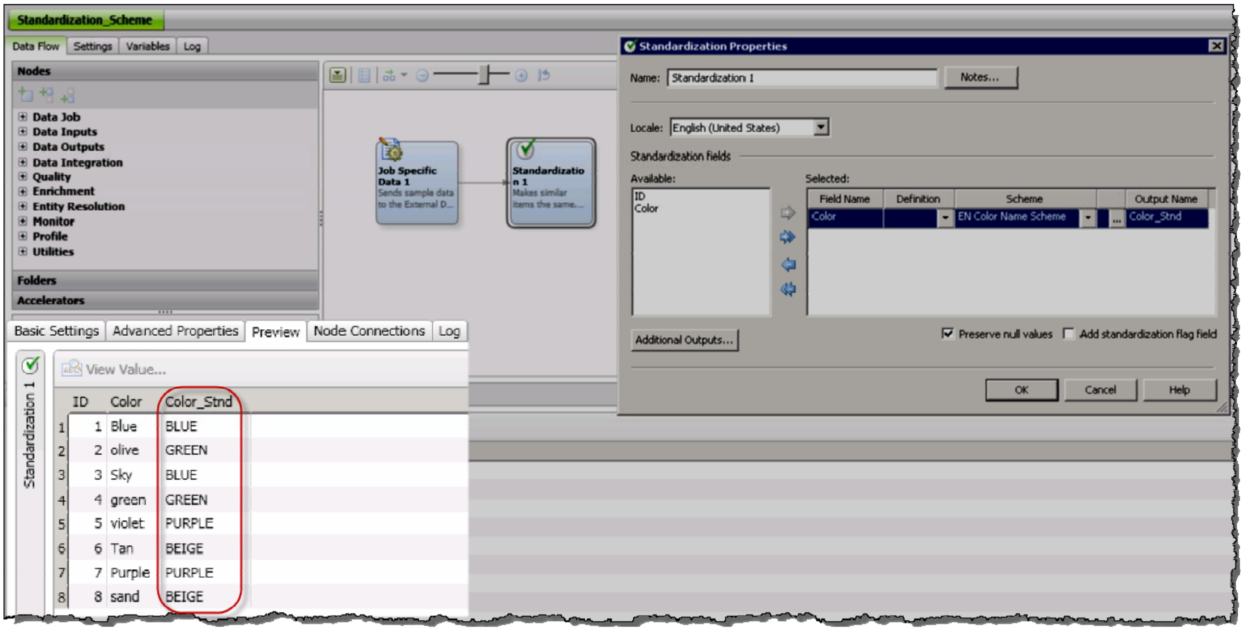
Field Name	Definition	Scheme		Output Name
Phone	Phone (with Country Code)			Phone_Stnd

☑ Preserve null values ☐ Add standardization flag field

	ID	Phone	Phone_Stnd
1	1	1-800-333-7660	+1 800 333 7660
2	2	(800) 333-7660	+1 800 333 7660
3	3	+19195311000	+1 919 531 1000
4	4	1-919-531-1000	+1 919 531 1000
5	5	+34 91 200 7300	+34 912007300
6	6	+34 (91) 200-7300	+34 912007300

Figure 2. Standardize Data – Standardization Definition Results

The Quality Knowledge Base might not provide a Standardization definition or scheme for a particular data type that you want to standardize. You can create your own custom Standardization definitions and/or schemes in the Quality Knowledge Base as needed. For example, in the figure below a custom Standardization scheme was created to standardize the names of colors and then was applied to the data. Now, the standardized color names can be used to produce better matching results.



Name: Standardization 1

Locale: English (United States)

Available:
ID
Color

Selected:

Field Name	Definition	Scheme	Output Name
Color		EN Color Name Scheme	Color_Stnd

☑ Preserve null values ☐ Add standardization flag field

	ID	Color	Color_Stnd
1	1	Blue	BLUE
2	2	olive	GREEN
3	3	Sky	BLUE
4	4	green	GREEN
5	5	violet	PURPLE
6	6	Tan	BEIGE
7	7	Purple	PURPLE
8	8	sand	BEIGE

Figure 3. Standardize Data – Standardization Scheme Results

Note: You can also use the DQSTANDARDIZE function and/or DQSCHEMEAPPLY function to standardize your data. Refer to the *SAS® 9.4: Data Quality Server Reference* guide (available at http://support.sas.com/documentation/cdl/en/dqclref/68376/PDF/default/dqclref.pdf) for more information about these functions.

MATCHING NAMES

Sometimes when trying to fuzzy match names, you want to fuzzy match just a portion of the name. For example, you might want to generate separate match codes for the Given Name and/or Family Name columns. A common mistake that people make is to map the Given Name and Family Name columns separately into the *Match Codes* node instead of using the *Match Codes (Parsed)* node.

So, why is this a mistake? The *Name* Match definition is designed to accept a full name and then parse the supplied name information into its tokens. This includes Name Prefix, Given Name, Middle Name, Family Name, Name Suffix, and Title/Additional Info.

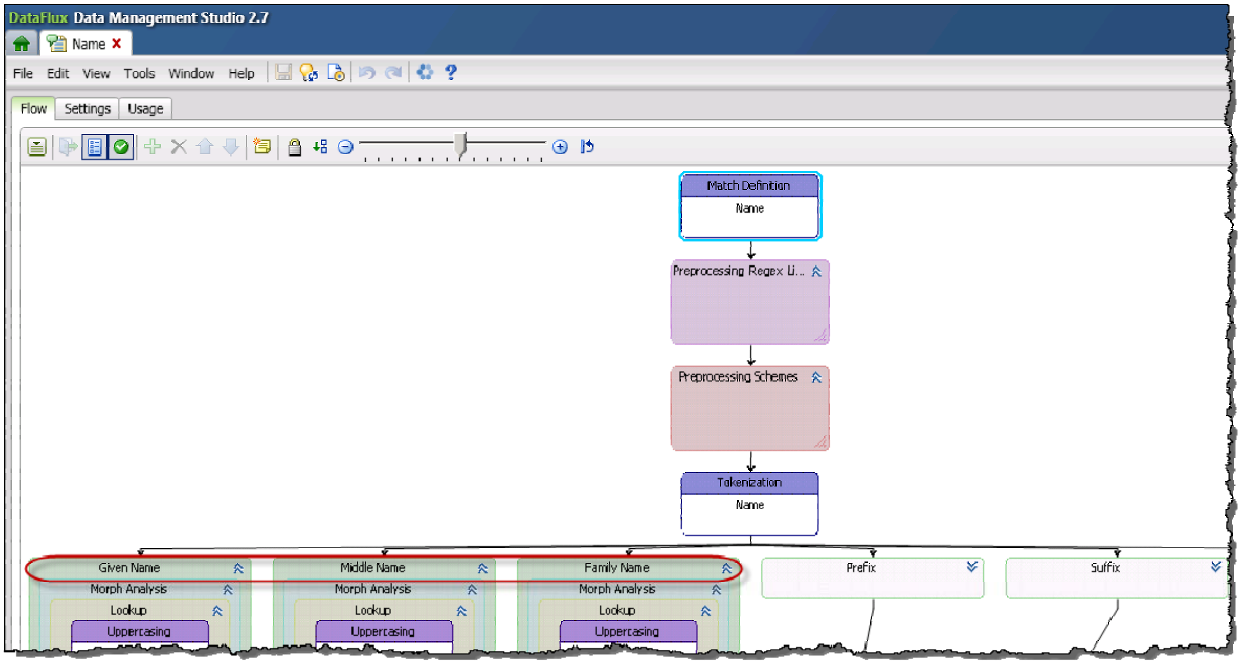


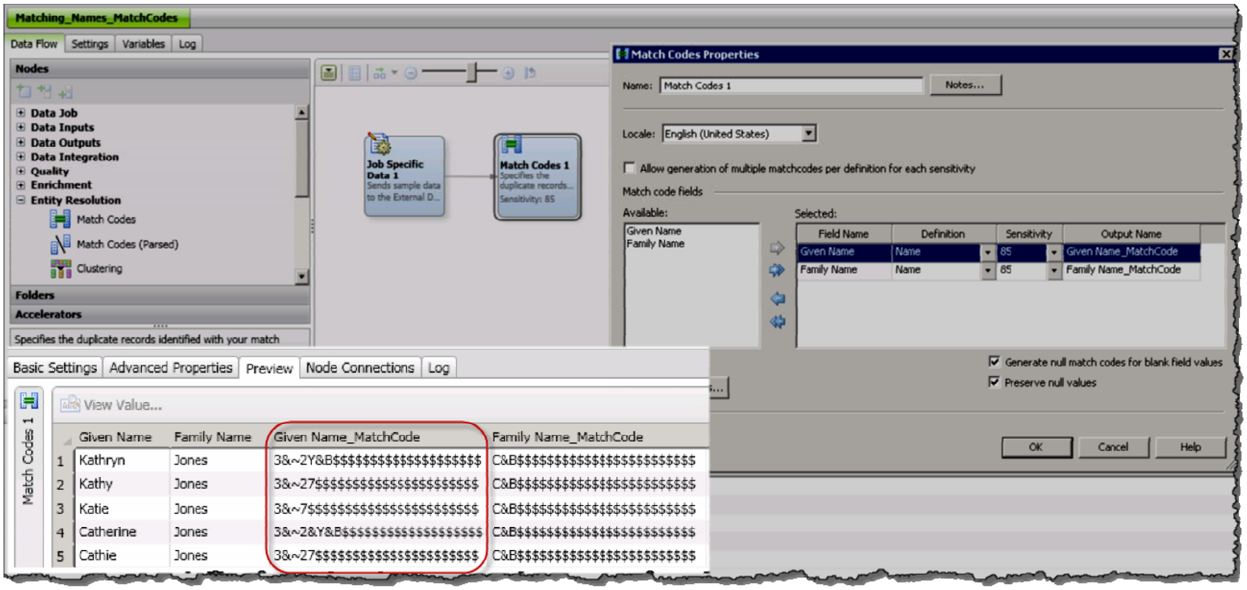
Figure 4. Name Match Definition Tokens

Now look at a case where you want to have separate match codes for Given Name and Family Name and your data is the following:

Given Name	Family Name
Kathryn	Jones
Kathy	Jones
Katie	Jones
Catherine	Jones
Cathie	Jones

Table 1. List of Names

Here are the match codes that were generated using the *Match Codes* node with the *Name* Match definition (*English – United States* locale) at a sensitivity of 85.



Match Codes Properties

Name: Match Codes 1

Locale: English (United States)

□ Allow generation of multiple matchcodes per definition for each sensitivity

Match code fields

Available: Given Name, Family Name

Selected:

Field Name	Definition	Sensitivity	Output Name
Given Name	Name	85	Given Name_MatchCode
Family Name	Name	85	Family Name_MatchCode

☑ Generate null match codes for blank field values

☑ Preserve null values

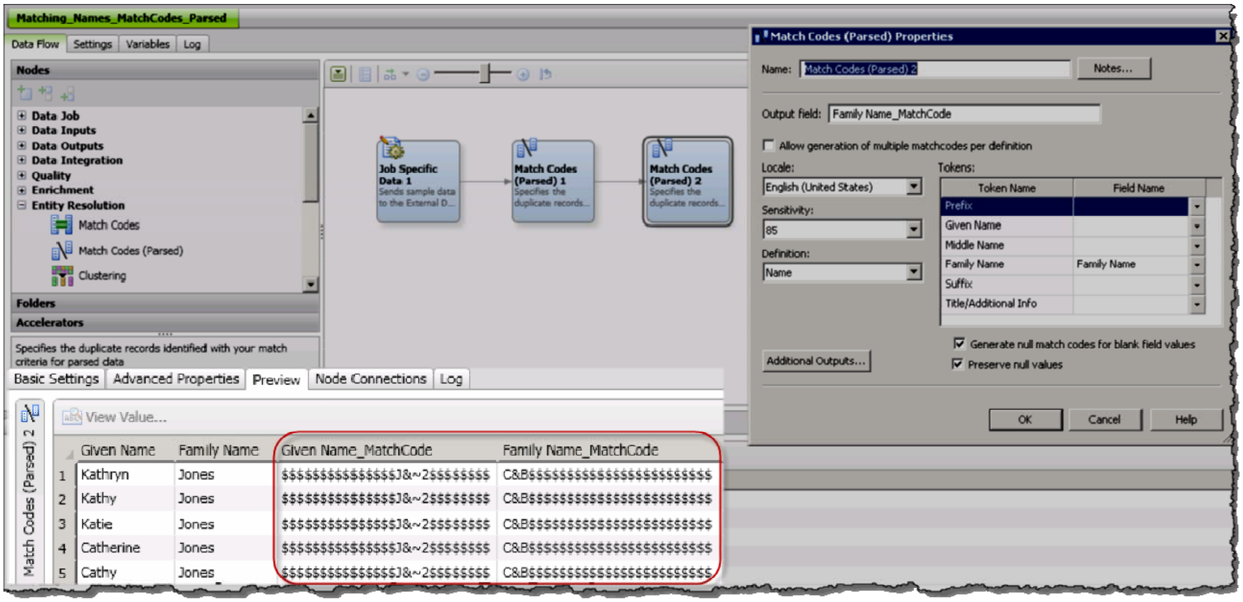
OK Cancel Help

	Given Name	Family Name	Given Name_MatchCode	Family Name_MatchCode
1	Kathryn	Jones	3&~2Y&B$$$$$$$$$$$$$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
2	Kathy	Jones	3&~27$$$$$$$$$$$$$$$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
3	Katie	Jones	3&~7$$$$$$$$$$$$$$$$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
4	Catherine	Jones	3&~2&Y&B$$$$$$$$$$$$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
5	Cathie	Jones	3&~27$$$$$$$$$$$$$$$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$

Figure 5. Name Match Codes Node Results

As shown in the figure above all the records would NOT match using this approach since the *Given_Name_MatchCodes* are not the same for all the records. The reason they do not match is that if only one name is supplied when calling the *Name* Match definition, then in most cases when the Name is parsed it assumes you supplied only the Family Name and the nickname equivalents of Given Name are not applied to the input.

The following match codes are generated using the *Match Codes (Parsed)* node with the *Name* Match definition (*English – United States* locale) at a sensitivity of 85. This is accomplished by feeding in the Given Name and Family Name field in the appropriate tokens in separate calls to the node.



Match Codes (Parsed) Properties

Name: Match Codes (Parsed) 2

Output field: Family Name_MatchCode

□ Allow generation of multiple matchcodes per definition

Locale: English (United States)

Sensitivity: 85

Definition: Name

Tokens:

Token Name	Field Name
Prefix	
Given Name	
Middle Name	
Family Name	Family Name
Suffix	
Title/Additional Info	

☑ Generate null match codes for blank field values

☑ Preserve null values

Additional Outputs...

OK Cancel Help

	Given Name	Family Name	Given Name_MatchCode	Family Name_MatchCode
1	Kathryn	Jones	$$$$$$$$$$$$$$$J&~2$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
2	Kathy	Jones	$$$$$$$$$$$$$$$J&~2$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
3	Katie	Jones	$$$$$$$$$$$$$$$J&~2$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
4	Catherine	Jones	$$$$$$$$$$$$$$$J&~2$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$
5	Cathy	Jones	$$$$$$$$$$$$$$$J&~2$$$$$$$$	C&B$$$$$$$$$$$$$$$$$$$$$$$$

Figure 6. Name Match Codes (Parsed) Node Results

In the case displayed above all the names will match since both the *Given_Name_MatchCode* and *Family_Name_MatchCode*s are the same for all the records. Using the *Match Codes (Parsed)* node ensures that the names are assigned to the proper tokens and the proper logic will be applied to each token. Therefore, this is the approach that you should use if trying to match Names based on just a portion of the name (e.g., Given Name and/or Family Name).

Note: You can also use the ParsedView of the function DQMATCH to generate match codes for parsed data. Refer to the *SAS® 9.4: Data Quality Server Reference* guide's section on "DQMATCH - Example 4: Creating Match Codes for Parsed Values" (available at http://support.sas.com/documentation/cdl/en/dqclref/68376/PDF/default/dqclref.pdf) for more information.

CROSS-FIELD MATCHING

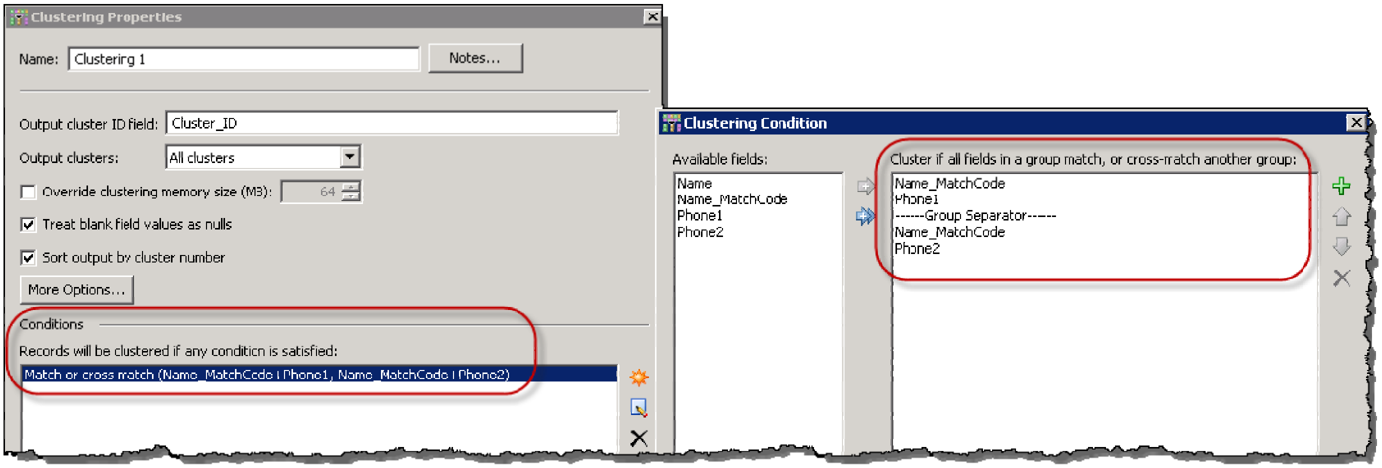
Cross-field matching is a matching feature in the *Clustering* node that has been available since the release of *DataFlux Data Management Studio 2.6*. It provides the ability to build rules that will cross over into other related columns.

Here is some sample input data used to illustrate the concept of cross-field matching.

Name	Phone1	Phone2
Michael T Smith	919-531-1212	919-123-4567
Mike Smith	919-123-4567	
Michael Smith		919-531-1212

Table 2. List of Names and Phone Numbers

This feature enables you to match the three records above without having to temporarily duplicate rows and/or swap columns to match the phone numbers as you would have to do in the old clustering engine. Now, you need to only create a match code for *Name* and write your cluster condition as follows in order for all 3 of these records to match.



Name: Clusterirg 1

Output cluster ID field: Cluster_ID

Output clusters: All clusters

☐ Override clustering memory size (MB): 64

☑ Treat blank field values as nulls

☑ Sort output by cluster number

Conditions

Records will be clustered if any condition is satisfied:

Match or cross match (Name_MatchCode | Phone1, Name_MatchCode | Phone2)

Available fields:

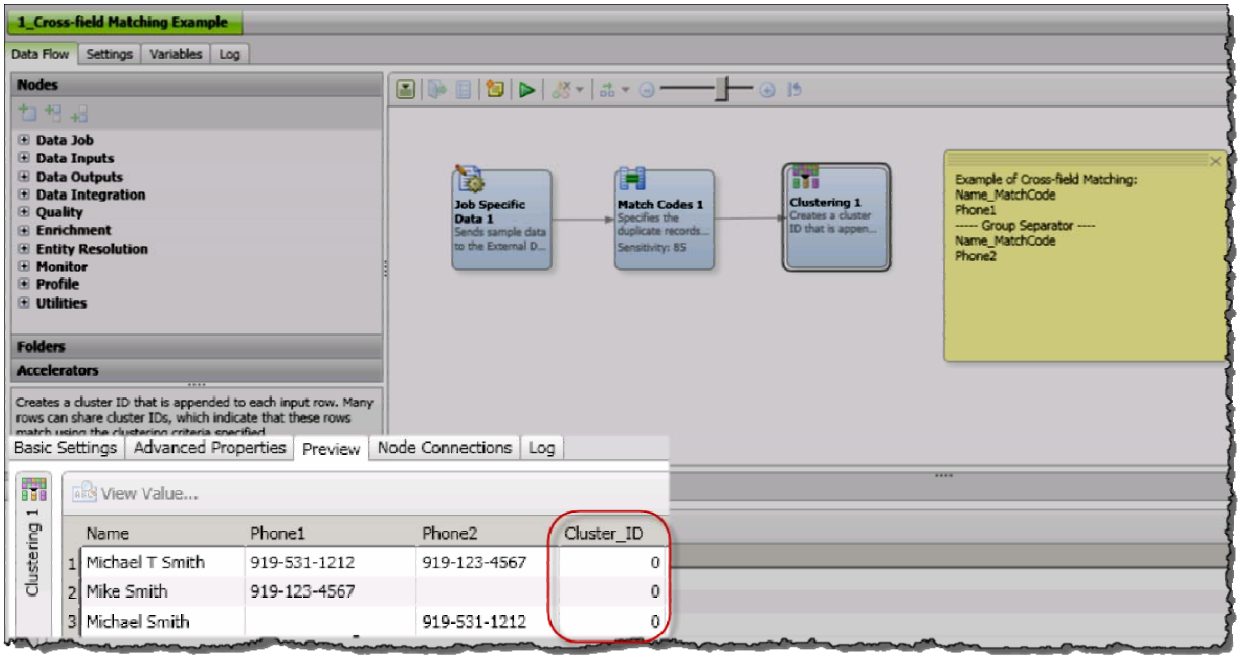
Name
Name_MatchCode
Phone1
Phone2

Cluster if all fields in a group match, or cross-match another group:

Name_MatchCode
Phone1
------Group Separator------
Name_MatchCode
Phone2

Figure 7. Cross-field Matching Clustering Node Properties

The *Group Separator* option is used to group match or cross-match with another group of conditions. Previewing the results of the *Clustering* node shows that all three records are placed in the same cluster.



Creates a cluster ID that is appended to each input row. Many rows can share cluster IDs, which indicate that these rows match using the clustering criteria specified

Example of Cross-field Matching:
Name_MatchCode
Phone1
----- Group Separator -----
Name_MatchCode
Phone2

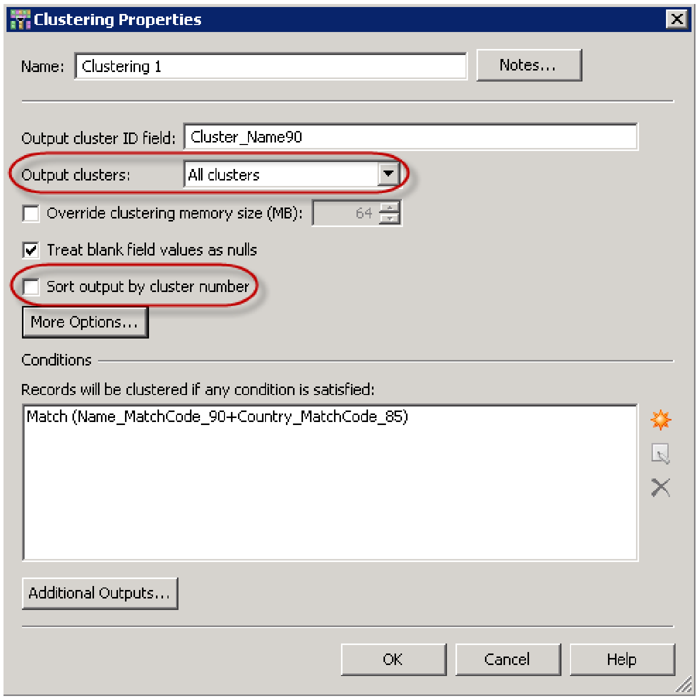
	Name	Phone1	Phone2	Cluster_ID
1	Michael T Smith	919-531-1212	919-123-4567	0
2	Mike Smith	919-123-4567		0
3	Michael Smith		919-531-1212	0

Figure 8. Cross-field Matching Results

CLUSTER COMPARISON

The *Cluster Diff* node compares the results of two different *Clustering* nodes based on the same input data. This is useful for comparing the results of different cluster conditions and/or different match code sensitivities.

All records from the input set must be passed to both *Clustering* nodes and both *Clustering* nodes must pass out all their data in the same order for this comparison to work. To summarize, in both *Clustering* nodes, you must select the *All clusters* output option and you cannot use the *Sort output by cluster number* option.



Name: Clustering 1

Output cluster ID field: Cluster_Name90

Output clusters: All clusters

□ Override clustering memory size (MB): 64

☑ Treat blank field values as nulls

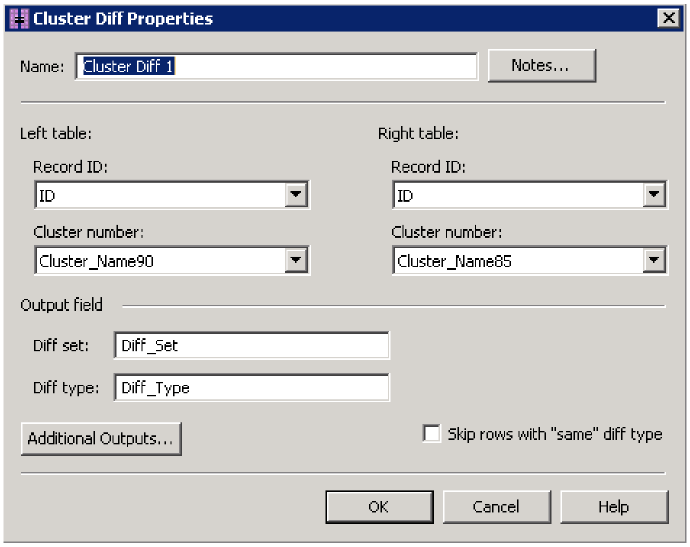
□ Sort output by cluster number

Records will be clustered if any condition is satisfied:

Match (Name_MatchCode_90+Country_MatchCode_85)

Figure 9. Clustering Node Properties

The results of both *Clustering* nodes are then fed into the *Cluster Diff* node. In order to perform the comparison, the *Cluster Diff* node must know the unique identifier for the input records (*Record ID*) and the *Cluster number* that is returned from the respective *Clustering* node.



Cluster Diff Properties

Name: Cluster Diff 1

Left table:

Record ID: ID

Cluster number: Cluster_Name90

Right table:

Record ID: ID

Cluster number: Cluster_Name85

Output field

Diff set: Diff_Set

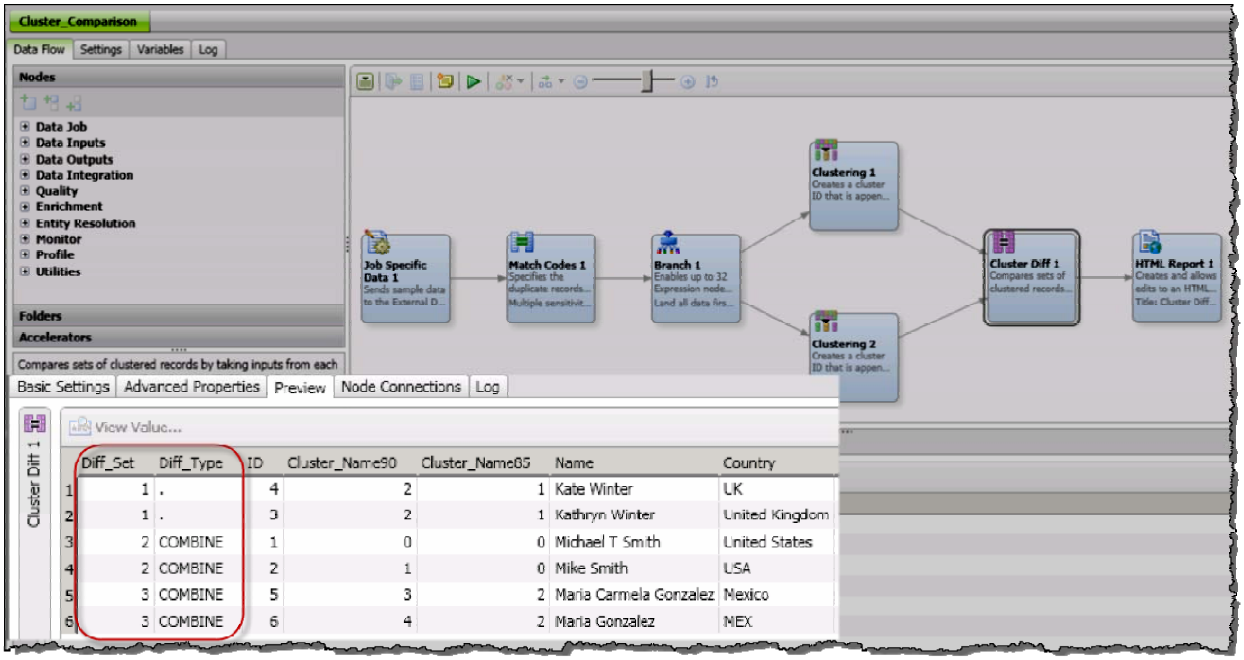
Diff type: Diff_Type

□ Skip rows with "same" diff type

Figure 10. Cluster Diff Node Properties

The *Diff type* value describes the type of change when performing the cluster number comparison between the two *Clustering* nodes. The possible values for *Diff type* include COMBINE, DIVIDE, NETWORK, and SAME which is represented as a period (**.**). When comparing the results of the two *Clustering* nodes the results are reviewed as a *Diff set*. Within a *Diff set*:

- If the records were in different clusters on the “left table” and in the same cluster on the “right table”, then its *Diff type* is COMBINE.
- If the records were in the same cluster on the “left table” and in different clusters on the “right table”, then its *Diff type* is DIVIDE.
- If the records were in same cluster on the “left table” and in the same cluster on the “right table”, then its *Diff type* is “**.**” (SAME).
- If when comparing the “left table” cluster to the “right table” clusters at least one record is added to the cluster AND at least one record is removed from the cluster, then its *Diff type* is NETWORK.



Compares sets of clustered records by taking inputs from each

	Diff_Set	Diff_Type	ID	Cluster_Name90	Cluster_Name85	Name	Country
1	1	.	4	2	1	Kate Winter	UK
2	1	.	3	2	1	Kathryn Winter	United Kingdom
3	2	COMBINE	1	0	0	Michael T Smith	United States
4	2	COMBINE	2	1	0	Mike Smith	USA
5	3	COMBINE	5	3	2	Maria Carmela Gonzalez	Mexico
6	3	COMBINE	6	4	2	Maria Gonzalez	MEX

Figure 11. Cluster Diff Node Results

The *Cluster Diff* node is not a node that is typically used in a production matching job. However, it is a node that is useful in helping you compare and contrast different match code sensitivities and/or cluster conditions that enable you to achieve the best matching results for your data set.

CONCLUSION

Matching records whether for the purposes of eliminating duplicate records or simply for grouping similar records together is an important component of your data quality effort. Using the techniques and tips described in this paper should help you on your way to achieving your perfect match with SAS® Data Management.

RECOMMENDED READING

- DataFlux® Data Management Studio: User’s Guide. Available at http://support.sas.com/documentation/onlinedoc/dfdmstudio/2.7/dmpdmsug/dfUnity.html
- SAS® Data Management Community. Available at https://communities.sas.com/t5/Data-Management/ct-p/data_management
- Rineer, B. 2015. “Garbage In, Gourmet Out: How to Leverage the Power of the SAS® Quality Knowledge Base.” Proceedings of the SAS Global Forum 2015 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings15/SAS1852-2015.pdf.

CONTACT INFORMATION

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

Mary Kathryn Queen
SAS Institute Inc.
MaryKathryn.Queen@sas.com
http://www.sas.com