Paper SAS6661-2016

Finding the One: Using SAS® Code and SAS® Data Quality Server to Create a Matched Record

Kimberly Hare and Elena Shtern, SAS Institute Inc.

ABSTRACT

In this era of data analytics, you are often faced with a challenge of joining data from multiple legacy systems. When the data systems share a consistent merge key, such as ID or SSN, the solution is straightforward. However, what do you do when there is no common merge key? If one data system has a character value ID field, another has an alphanumeric field, and the only common fields are the names or addresses or dates of birth, a standard merge query does not work. This paper demonstrates fuzzy matching methods that can overcome this obstacle and build your master record through Base SAS® coding. The paper also describes how to leverage the SAS® Data Quality Server in SAS® code.

INTRODUCTION

Fuzzy matching is described as "the technique of finding strings that match a pattern approximately rather than exactly."(Wikipedia 2016) This paper demonstrates two fuzzy matching methods that can be used to build a master record if there are no common fields.

We will first review a custom scoring method implemented in Base SAS®. This method is aimed at users with an intermediate programming skill level, and is appropriate to apply to data with moderate data quality. The second method leverages SAS® Data Quality Server. It requires basic SAS® skills and is more appropriate when working with weaker quality data that would benefit from more robust standardization and data quality improvement routines.

DEMO DATA SETS

Two data sets, DEMOGRAPHIC and PURCHASE, are used to demonstrate the application of fuzzy matching methods.

Note: These data sets are a part of the training course for SAS® DataFlux®, and have been slightly modified for the purpose of this paper.

The demographics data set contains standard demographics fields such as gender, education, income level, and household information. The purchase data set contains information related to vehicle purchases. The common fields between the two data sets are name, address, city, state, zip, and phone number. The objective is to enhance demographics data with purchase information to analyze any potential relationships that exist between demographics indicators and vehicle purchases.

If we performed a standard merge by original name, address, state, and phone number, this would result in only four matching records. While on the surface there should be more matches, the low hit rate is due to misspellings and inconsistent use of middle names on some records but not the others, or the use of nicknames versus full names, such as "Sam" versus "Sammy", or inconsistent use of street types, such as "Ave" versus "Avenue" or "Rd" versus "Road". Those types of data quality issues are common when working with any text fields, especially names and addresses. To overcome those issues and improve the hit rate, data standardization and fuzzy matching methods have to be used.

SCORING METHOD

We will begin with the scoring method that is implemented in Base SAS®. This method uses a combination of merges and joins, simple standardization, fuzzy matching functions (Russell 2016), and a scoring algorithm to match with robustness despite inconsistent data quality.

SCORING METHODOLOGY

The scoring method allows you to find a match, even though all of your common fields are not exact matches. This methodology is robust against typographical errors, misspellings, or blank fields. Using the scoring method, you assign points for each set of fields that match across two data sets. Each field will get a customized point value based on its strength in identifying a record. For example, first name, last name, and date of birth would have higher point values than items like gender or middle name.

Next, you determine an acceptable threshold level for your matches. You then total all the points a record has received and compare that against the threshold. Too few points, and it is not a match. For example, using three high-value items (first name, last name, and date of birth) and two low-value items (gender and middle name), we assign a value of 20 points each to the three high-value items and 10 points each to the two lower-value items. If our threshold is set at 60, that would require that either you have all three high-value characteristics in common, or two high-value characteristics and two low-value characteristics to reach a score of 60.

The more common fields you have between two tables, the higher your threshold should be. Make sure when assigning the points for your scores that you can get a variety of matches that make sense. A key point to consider is that you do not want to determine a match based only on low-value items. If your threshold allows for a match where only your low-value items get points, you need to revisit your point allocation or your threshold level.

In our sample data, we use NAME, ADDRESS, STATE, and PHONE_NUMBER to match on. We parse out the NAME into FIRST_NAME and LAST_NAME, giving us a total of five fields to score with.

Our points are set so that if the FIRST_NAME matches, then it gives us 20 points. LAST_NAME and ADDRESS also get 20 points each, while STATE and PHONE_NUMBER each get 10 points.

Our threshold will be 60. There are multiple ways to get to 60 with these points. Here are a few examples in Figure 1:

Data Set 1	Data Set 2	Points
First_Name	**First_Name**	**First_Name Points**
Jon	Jonathan	0
Last_Name	**Last_Name**	**Last_Name Points**
Smith	Smith	20
Address	**Address**	**Address Points**
123 North Parkway	123 North Parkway	20
State	**State**	**State Points**
VA	VA	10
Phone_Number	**Phone_Number**	**Phone_Number Points**
703-123-4567	703-123-4567	10
	Total Score:	**60**

Data Set 1	Data Set 2	Points
First_Name	**First_Name**	**First_Name Points**
Eric	Eric	20
Last_Name	**Last_Name**	**Last_Name Points**
Anderson	Anderson	20
Address	**Address**	**Address Points**
456 South Parkway	456 South Parkway	20
State	**State**	**State Points**
VA	MD	0
Phone_Number	**Phone_Number**	**Phone_Number Points**
703-123-4567	703-123-4567	10
	Total Score:	**70**

Figure 1. Scoring Sample

The easiest way to meet the threshold is by matching on FIRST_NAME, LAST_NAME, and ADDRESS, our high point value fields. However, looking at the first example, we see the importance of the low point value fields. The threshold is met by combining LAST_NAME, ADDRESS, STATE, and PHONE_NUMBER. Keep in mind that not every high-value field will be populated or receive match points, but by combining points from our low-value fields with some of the high-value field points, we have a better chance of meeting the threshold.

Now that we have discussed our process, let's look at what you really want to see—SAS code!

IMPLEMENTATION

The code is divided into four major sections: Data Prep, Simple Matches, Scoring Matches, and Bringing All the Data Together. Each section is discussed in detail below.

Step 1. Data Prep

The first step is to parse out the NAME field into FIRST_NAME and LAST_NAME. We also do some simple standardization by setting the field length and using the UPCASE function:

```
/* Demographic Data */
   data demo_standardize (rename=(address_keep = address ct = city st =
      state));
    format ct $14. st $2.;
    set sgf.demographic_data;
    last_name = upcase(scan(name, -1, ' ')); /* Negative value scans from
```

```
  right to left */
   first_name = upcase(scan(name, 1, ' ')); /* Positive value scans from
                                                              left to right */
   address_keep = upcase(address);
   ct = upcase(city);
   st = upcase(state);
   drop address city state;
  run;

/* Purchase Data */
  data purchase_standardize (rename=(address_keep = address ct = city st =
      state));
   format ct $14. st $2.;
   set sgf.purchase_data;
   last_name = upcase(scan(name, -1, ' '));
   first_name = upcase(scan(name, 1, ' '));
   address_keep = upcase(address);
   ct = upcase(city);
   st = upcase(state);
   drop address city state;
  run;
```

Step 2. Simple Matches

Next, we grab the low-hanging fruit in terms of matching. Using a simple MERGE statement, select any records that have an exact match and set them aside, and create two separate tables for our unmatched records to use in scoring. We keep only the fields that matter most to us for matching, to improve our processing time:

```
  proc sort data=demo_standardize;
   by first_name last_name address state phone;
  run;

  proc sort data=purchase_standardize;
   by first_name last_name address state phone;
  run;

  data demo_purchase_match demo_only (keep= first_name last_name address
                        state phone) /* Trim down to fields to match on */
            purchase_only (keep= first_name last_name address state phone);
   merge demo_standardize (in=a) purchase_standardize (in=b);
   by first_name last_name address state phone;
   if a and b then output demo_purchase_match;
   if a and not b then output demo_only;
   if b and not a then output purchase_only;
  run;
```

Step 3. Scoring

This is where the code starts to get interesting. We know that to score, we have to compare common fields from two different data sets. To do this, we have to get them on the same row, but we want to keep that row only if the common fields actually match. We will execute a nested loop with a modified match-crossing method (McAllaster 2016) to find our matching records.

Since all of our data will be on a single row, we need to rename the records of one of the data sets. For this example, the unmatched records from the DEMO_ONLY data set were renamed by appending "_d" to each field name:

```
  data demo_only_rename (rename=(first_name = first_name_d last_name =
      last_name_d state = state_d address = address_d phone = phone_d));
```

```
 set demo_only;
run;
```

The next step continues to prepare data for the loop. The loop needs to go through both data sets, so a global macro value is created with the number of records for each DATA step. These macro variables will be used to stop each loop:

```
data _null_;
 %let dsid = %sysfunc(open(demo_only_rename)); /* Opens the
                                                 DEMO_ONLY_RENAME table */
 %let demo_last_rec = %sysfunc(attrn(&dsid,nobs)); /* Counts the total
number of records and assigns that value to DEMO_LAST_REC */
 rc = %sysfunc( close(&dsid) ); /* Closes the table */

/* Repeats process for PURCHASE_ONLY */
 %let dsid = %sysfunc(open(purchase_only));
 %let purchase_last_rec = %sysfunc(attrn(&dsid,nobs));
 rc = %sysfunc(close(&dsid));
run;
```

Now we get to our nested loop and cross-match. The macro %DO loop is driven by our main table, the demographic data table. The second DO UNTIL loop is driven by our second table, the purchase information. Those global macro values we just made tell the loops when to stop.

The macro %DO loop must exist inside of a macro statement, hence the simple replacement of the table names and creation of the MATCH macro. The macro %DO loop creates a macro variable out of i, pointing to a record number in the DEMO_ONLY_RENAME data set:

```
%macro match (table1, table2);

%do i = 1 %to &demo_last_rec.; /* Number of times loop will execute */

data min_score_60 ;
 set &table1.;
 if _n_ = &i;
 do j = 1 to &purchase_last_rec. until (tot_score >=60);
```

If you were to replace &i with the number 4, it would take the fourth record from the DEMO_ONLY_RENAME data set, and process it through the rest of the DATA step. See Figure 1:

DEMO_ONLY_RENAME

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send

	state_d	address_d	phone_d	last_name_d	first_name_d
1	CA	4211 S Rushford St	860-952-3496	SARGENT	ABIGAIL
2	CA	5024 Fairbanks W...	618-121-6649	HULBERT	ANDRE
3	OH	P.O. Box 6239	271-475-5054	BEY	ASHLEY
4	MO	4400 NC Highway...	718-922-0353	LAPP	CATHY
5	MO	515 E. Broad St.,...	949-161-1908	PRENTISS	CINDY
6	CA	824 Valerie Dr Uni...	806-295-2544	GRASSI	DAVID

Figure 1. The &i = 4 Record from DEMO_ONLY_RENAME

Rather than copy and paste our DATA step 26 times, updating the &i value each time (because we know we are better programmers than that), we use the macro %DO loop to run through all the records.

Now that we've reviewed that macro, we'll cover the second loop. This is a DO UNTIL loop driven by our second table (PURCHASE_ONLY). Our &i has selected our record from DEMO_ONLY_RENAME, and

the second loop will take that record and compare it through all the records in PURCHASE_ONLY until our condition is met—a score of 60 or above. This section is all about the j value driving it:

```
data min_score_60 ;
 set &table1.;
 if  n  = &i;
 do j = 1 to &purchase_last_rec. until (tot_score >=60);
```

We are taking the same record from the DEMO_ONLY_PURCHASE table, and comparing it to everything in our PURCHASE_ONLY table, one line at a time. The DEMO_ONLY_RENAME record is driven by that &i, but the j value is what will cycle us through all of the PURCHASE_ONLY records until either the total score is greater than or equal to 60, or the j value exceeds &PURCHASE_LAST_REC.

Let's walk through that loop assuming that our &i value is 6. See Figure 2:

DEMO_ONLY_RENAME

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	state_d	phone_d	last_name_d	first_name_d	address_d
1	CA	860-952-3496	SARGENT	ABIGAIL	4211 S RUSHFORD ST
2	CA	618-121-6649	HULBERT	ANDRE	5024 FAIRBANKS WAY
3	OH	271-475-5054	BEY	ASHLEY	P.O. BOX 6239
4	MO	718-922-0353	LAPP	CATHY	4400 NC HIGHWAY, PO BOX 44
5	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD ST., STE. 12
6	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNIT 30
7	CT	660-469-0704	NATH	DENISE	1215 N CALDWELL ST

Figure 2. The &i = 6 Record from DEMO_ONLY_RENAME

Now, we get down to the j section of the loop. It will start with j = 1, which we see in Figure 3:

PURCHASE_ONLY

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	state	PHONE	last_name	first_name	address
1	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239
2	MO	718-922-0353	LAP	CATHY	4400 NC HIGHWAY, PO BOX 44
3	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD ST., SUITE. 12
4	CA	806-295-2544	GRASSI	DAVE	824 VALERIE DR UNIT 30

Figure 3. The j = 1 Record from PURCHASE_ONLY

The loop will put them on the same row and then it will go through the scoring conditions (explained after this section) to get a total score (the field TOT_SCORE).

	state_d	phone_d	last_name_d	first_name_d	address_d	j	tot_score	state	PHONE	last_na	first_name	address
1	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	1	0	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239

Figure 4. j = 1 scoring results

However, since TOT_SCORE is not 60, and j, our counter in the PURCHASE_ONLY table, does not equal &PURCHASE_LAST_REC (which has a value of 21), neither of our DO UNTIL loop conditions have been met, so it will move on to the next record. Figures 6–8 illustrate the table as the j value increases.

	state_d	phone_d	last_name_d	first_name_d	address_d	j	tot_score	state	PHONE	last_na	first_name	address
1	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	1	0	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239
2	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	2	0	MO	718-922-0353	LAP	CATHY	4400 NC HIGHW...

Figure 5. j = 2 scoring, total score is less than 60

	state_d	phone_d	last_name_d	first_name_d	address_d	j	tot_score	state	PHONE	last_na	first_name	address
1	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	1	0	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239
2	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	2	0	MO	718-922-0353	LAP	CATHY	4400 NC HIGHW...
3	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	3	0	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD S...

Figure 6. j = 3, total score is less than 60

	state_d	phone_d	last_name_d	first_name_d	address_d	j	tot_score	state	PHONE	last_na	first_name	address
1	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	1	0	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239
2	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	2	0	MO	718-922-0353	LAP	CATHY	4400 NC HIGHW...
3	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	3	0	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD S...
4	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNI...	4	60	CA	806-295-2544	GRASSI	DAVE	824 VALERIE DR...

Figure 7. j = 4, total score IS greater than or equal to 60

In Figure 7 we reach a score that is greater than or equal to 60 and the logic in our DO UNTIL loop tells it to stop and output the record.

The &i value will now increase to 7, and that record will go through this same process, being compared to every record in the PURCHASE_ONLY table until either it finds a value greater than or equal to 60 and outputs that record, or it reaches the end of the PURCHASE_ONLY data set with no matches. If it reaches the end of PURCHASE_ONLY with no matches, it will not output a record and the &i counter will move on to 8, and send the next DEMO_ONLY_RENAME record through the process.

Now that we have a better sense of what is happening as it moves through the loop, let's look at the code driving the logic for our scoring method. Once in the loop, we use a BY statement with a FIRST. variable to make sure the baseline points are set to zero, and then we apply our conditions:

```
set &table2.;
 by first_name
 last_name
 address
 state
 phone;

 if first.first_name or first.last_name or first.address or first.state
 or first.phone then do;

   first_name_score = 0;
   last_name_score = 0;
   address_score = 0;
   state_score = 0;
   phone_score = 0;


   if COMPGED(first_name,first_name_d) <= 70 then first_name_score = 20;
   if COMPGED(last_name,last_name_d) <= 30 then last_name_score = 20;
   if COMPGED(address, address_d) <= 500 then address_score = 20;
   if state = state_d then state_score = 10;
   if phone = phone_d then phone_score = 10;

   tot_score = first_name_score + last_name_score + address_score +
   state_score + phone_score;
  end;
```

Because we know that we'll run into issues like minor typographical errors in names (last name Tratter versus Trater), COMPGED is used to allow some fuzzy matching with name and address. The lower the COMPGED output value, the more similar the text values are. With first name, we set a higher value of 70, but with last name, we want a tighter match at 30. Testing the address values, we found that 500 was an appropriate value for the example data set to maintain data quality while also allowing for some moderate differences in the ADDRESS field.

Using the records from Figure 7, here is what the scoring output looks like in Figure 9:

	j	tot_score	state	PHONE	last_na	first_name	address	first_name_score	last_name_score	address_score	state_score	phone_score
1	1	0	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239	0	0	0	0	0
2	2	0	MO	718-922-0353	LAP	CATHY	4400 NC HIGHW...	0	0	0	0	0
3	3	0	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD S...	0	0	0	0	0
4	4	60	CA	806-295-2544	GRASSI	DAVE	824 VALERIE DR...	0	20	20	10	10

Figure 8. Scoring Output

Even though the loop is finding many records with TOT_SCORE values below 60, we don't want those records. We want only our records with a TOT_SCORE greater than or equal to 60 to be output, so a simple IF statement solves that for us:

```
if tot_score >= 60 then do;
 output min_score_60;
end;
```

The j DO UNTIL loop ends, and the DATA step is completed.

The final step in our MATCH macro writes out all the matching values to the MATCH_VALS data set. The scoring values are all dropped from the final output data set. The macro %DO loop reaches the %END, and then our &i value increases by one, and repeats the whole process for the next record in the DEMO_ONLY_RENAME data set:

```
 data match_vals (drop=j tot_score first_name_score last_name_score
 address_score state_score phone_score);
  set  min_score_60 match_vals;
 run;
%end;
%mend match;
%match (demo_only_rename, purchase_only);
```

Viewing our MATCH_VALS table we see that we have 18 matches. Highlighted in red are some interesting differences in our data. The scoring method is robust against these minor data quality issues, and it provides successful matches. Figure 10 shows the results:

MATCH_VALS

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	state_d	phone_d	last_name_d	first_name_	address_d	state	PHONE	last_name	first_name	address
1	OH	419-566-4321	TRETTER	SIDNEY	161 NORTHFORK RD	OH	419-566-4321	TRETER	SIDNEY	161 NORTHFORK RD
2	CA	902-861-5137	MANDELBAUM	SHARON	3540 WILSHIRE BLVD	CA	902-861-5137	MANDELBAM	SHARON	3540 WILSHIRE BLVD.
3	MO	731-887-4557	HARLEEN	SAMUEL	102 ECHO GLEN DR	MO	731-887-4557	HARLEEN	SAM	102 ECHO GLEN DR
4	MO	297-073-4204	RUBENSTEIN	ROB	3939 RUFFIN RD	MO	297-073-4204	RUBENSTEIN	ROBERT	3939 RUFFIN ROAD
5	UT	456-854-5248	PIETRON	PAT	4305 CENTRAL AVE WEST	UT	456-854-5248	PIETRON	PATRICK	4305 CENTRAL AVENUE WEST
6	MO	956-586-4147	MAZO	MS.	1515 LORD ASHLEY DR	MO	956-586-4147	MAZO	SHANNON	1515 LORD ASHLEY DR
7	WA	357-989-4735	VOSS	LOU	777 S. HARBOR BLVD.	WA	357-989-4735	VOSS	LOUIS	777 S. HARBOR BLVD
8	NH	991-312-5527	SEKERES	K	5541 CENTRAL AVENUE	NH	991-312-5527	SEKERES	KURT	5541 CENTRAL AVE
9	IL	843-042-5960	GEE	JONATHAN	4900 RIVERGRADE RD	IL	843-042-5960	MCGEE	JONATHAN	4900 RIVERGRADE RD
10	MO	615-005-6993	SOON	JOHNATHON	239 N EDGEWORTH ST	MI	615-005-6993	SOON	JOHNATHON	239 N EDGEWORTH SREET
11	TX	738-818-2196	MACIAS	JESSICA	5230 WALNUT GROVE LN	TE	738-818-2196	MACIAS	JESSICA	5230 WALNUT GROVE LANE
12	KS	530-406-2382	EINHORN	ERIC	23 S SAUNDERS RD	KS	530-406-2382	EINHORN	ERICK	23 S SAUNDERS RD
13	VA	500-881-3331	FUSCO	E	2617 RAMSEY RD #8	VI	500-881-3331	FUSCO	E	2617 RAMSEY RD #8
14	MN	850-323-7265	DOTY	DOUGLAS	406 MCCLURE CIR	MN	850-323-7265	DOTY	DOUGLAS	406 MCCLURE CIRCLE
15	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR UNIT 30	CA	806-295-2544	GRASSI	DAVE	824 VALERIE DR UNIT 30
16	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD ST., STE. 12	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD ST., SUITE. 12
17	MO	718-922-0353	LAPP	CATHY	4400 NC HIGHWAY, PO BOX 44	MO	718-922-0353	LAP	CATHY	4400 NC HIGHWAY, PO BOX 44
18	OH	271-475-5054	BEY	ASHLEY	P.O. BOX 6239	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239

Figure 9. Scored Matching Results

Notice that in line 6 under FIRST_NAME_D, there is a title (MS.) instead of a name. However, since the other values matched and were able to score a 60, the match was successful. Many records have minor spelling differences in last names, or first names instead of nicknames. In line 11 we see that the address used "LN" for ADDRESS_D, but "LANE" in address. The COMPGED function added to our address allowed it to have more robustness when matching on the addresses.

Step 4. Bringing All the Data Together

Now that we have our matched records, we can tie them back to our original data sets.

We start by bringing over all the standardized values from our PURCHASE_STANDARDIZE table. This is done with a simple JOIN where we keep all of our matched value fields, but add in CAR_MAKE, CAR_MODEL, and CAR_YEAR. The results are in Figure 11:

```
/* Takes our matching results and pulls in all the purchase information. */
proc sql;
```

```
 create table add_purchase as
 select a.car_make, a.car_model, a.car_year, b.*
 from purchase_standardize a, match_vals b
 where a.first_name = b.first_name and a.last_name = b.last_name and
  a.address = b.address and a.state = b.state and a.phone = b.phone
;
quit;
```

ADD_PURCHASE

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	CAR_MAKE	CAR_MODEL	CAR_YEAR	state_d	phone_d	last_name_d	first_name_d	address_d	state	PHONE	last_name	first_name	address
1	Acura	MDX	2009	OH	419-566-4321	TRETTER	SIDNEY	161 NORTHFOR...	OH	419-566-4321	TRETER	SIDNEY	161 NORTHFOR...
2	BMW	M6	2005	CA	902-861-5137	MANDELBAUM	SHARON	3540 WILSHIRE...	CA	902-861-5137	MANDELBAM	SHARON	3540 WILSHIRE...
3	Honda	Accord	2011	MO	731-887-4557	HARLEEN	SAMUEL	102 ECHO GLEN...	MO	731-887-4557	HARLEEN	SAM	102 ECHO GLEN...
4	Hyundai	Accent	2013	MO	297-073-4204	RUBENSTEIN	ROB	3939 RUFFIN RD	MO	297-073-4204	RUBENSTEIN	ROBERT	3939 RUFFIN RO...
5	Toyota	Camry	2012	UT	456-854-5248	PIETRON	PAT	4305 CENTRAL A...	UT	456-854-5248	PIETRON	PATRICK	4305 CENTRAL A...
6	BMW	X5	2006	MO	956-586-4147	MAZO	MS.	1515 LORD ASHL...	MO	956-586-4147	MAZO	SHANNON	1515 LORD ASHL...
7	Hyundai	Elantra	2010	WA	357-989-4735	VOSS	LOU	777 S. HARBOR...	WA	357-989-4735	VOSS	LOUIS	777 S. HARBOR...
8	Acura	RDX	2009	NH	991-312-5527	SEKERES	K	5541 CENTRAL A...	NH	991-312-5527	SEKERES	KURT	5541 CENTRAL A...
9	Honda	Accord	2015	IL	843-042-5960	GEE	JONATHAN	4900 RIVERGRA...	IL	843-042-5960	MCGEE	JONATHAN	4900 RIVERGRA...
10	Hyundai	Santa Fe	2013	MO	615-005-6993	SOON	JOHNATHON	239 N EDGEWOR...	MI	615-005-6993	SOON	JOHNATHON	239 N EDGEWOR...
11	Acura	MDX	2010	TX	738-818-2196	MACIAS	JESSICA	5230 WALNUT G...	TE	738-818-2196	MACIAS	JESSICA	5230 WALNUT G...
12	Hyundai	Tuscon	2008	KS	530-406-2382	EINHORN	ERIC	23 S SAUNDERS...	KS	530-406-2382	EINHORN	ERICK	23 S SAUNDERS...
13	Toyota	Tundra	2013	VA	500-881-3331	FUSCO	E	2617 RAMSEY R...	VI	500-881-3331	FUSCO	E	2617 RAMSEY R...
14	Mercedes-Benz	E350	2014	MN	850-323-7265	DOTY	DOUGLAS	406 MCCLURE CI...	MN	850-323-7265	DOTY	DOUGLAS	406 MCCLURE CI...
15	Toyota	Camry	2012	CA	806-295-2544	GRASSI	DAVID	824 VALERIE DR...	CA	806-295-2544	GRASSI	DAVE	824 VALERIE DR...
16	Mercedes-Benz	C300	2007	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD S...	MO	949-161-1908	PRENTISS	CINDY	515 E. BROAD S...
17	Acura	RDX	2009	MO	718-922-0353	LAPP	CATHY	4400 NC HIGHW...	MO	718-922-0353	LAP	CATHY	4400 NC HIGHW...
18	BMW	M2	2014	OH	271-475-5054	BEY	ASHLEY	P.O. BOX 6239	OH	271-475-5054	BEY	ASHLEY	PO BOX 6239

Figure 10. ADD_PURCHASE Results

Next, we add our standardized demographic information, but as we do this we start to drop some of our intermediate fields used for matching. Refer to Figure 4 again—notice that we have two of every field. Here we will just keep the values that come from our standardized demographic table, but add on our CAR_MAKE, CAR_MODEL, and CAR_YEAR. Then we bring in our values from that first, simple match that we did through a DATA step. The results are in Figure 12:

```
proc sql;
 create table add_demo as
 select a.*, b.car_make, b.car_model, b.car_year
 from demo_standardize a, add_purchase b
 where a.first_name = b.first_name_d and a.last_name = b.last_name_d and
  a.address = b.address_d and a.state = b.state_d and a.phone = b.phone_d
;
quit;

/* Adds the matches we got at the beginning */
data combine_all_matches;
 set add_demo DEMO_PURCHASE_MATCH;
run;
```

ADD_DEMO

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	city	state	NAME	ZIP	PHONE	DOB	Gender	Education	Income_Level	Household	last_name	first_name	address	CAR_MAKE	CAR_MODEL	CAR_YEAR
1	N. RIDGEVI...	OH	Ashley Bey	44039	271-475-50...	14MAR1972	F	Bachelor's deg...	$100,000 to $149,...	Single, never...	BEY	ASHLEY	P.O. BOX 62...	BMW	M2	2014
2	ST. LOUIS	MO	Cathy Lapp	63134	718-922-03...	28JAN1982	F	Doctorate degr...	$75,000 to $99,999	Widowed	LAPP	CATHY	4400 NC HIG...	Acura	RDX	2009
3	ST LOUIS	MO	Cindy Prentiss	63146	949-161-19...	26JUN1974	F	Doctorate degr...	$50,000 to $74,999	Divorced	PRENTISS	CINDY	515 E. BROA...	Mercedes-Benz	C300	2007
4	POMONA	CA	David Grassi	91768	806-295-25...	20OCT1985	M	Bachelor's deg...	Less than $25,000	Single, never...	GRASSI	DAVID	824 VALERI...	Toyota	Camry	2012
5	MINNEAPOL...	MN	Douglas Doty	55431	850-323-72...	14FEB1970	M	Doctorate degr...	$75,000 to $99,999	Divorced	DOTY	DOUGLAS	406 MCCLU...	Mercedes-Benz	E350	2014
6	HERNDON	VA	E Fusco	22070	500-881-33...	08JUN1983	M	Bachelor's deg...	$75,000 to $99,999	Single, never...	FUSCO	E	2617 RAMSE...	Toyota	Tundra	2013
7	HUTCHINS	KS	Eric Einhorn	67504...	530-406-23...	12FEB1973	M	Bachelor's deg...	$75,000 to $99,999	Single, never...	EINHORN	ERIC	23 S SAUND...	Hyundai	Tuscon	2008
8	SHERMAN	TX	Jessica Maci...	75090...	738-818-21...	20JUN1977	F	Bachelor's deg...	$100,000 to $149,...	Widowed	MACIAS	JESSICA	5230 WALN...	Acura	MDX	2010
9	ST LOUIS	MO	Johnathon So...	63104	615-005-69...	17JUN1976	F	Master's degree	$100,000 to $149,...	Married or dom...	SOON	JOHNATHON	239 N EDGE...	Hyundai	Santa Fe	2013
10	BANNOCKB...	IL	Jonathan Mc...	60015	843-042-59...	19OCT1974	F	Bachelor's deg...	$150,000 to $199,...	Married or dom...	GEE	JONATHAN	4900 RIVER...	Honda	Accord	2015
11	MARLOW	NH	K Sekeres	03456	991-312-55...	06AUG1971	F	Master's degree	$50,000 to $74,999	Widowed	SEKERES	K	5541 CENTR...	Acura	RDX	2009
12	BELLEVUE	WA	Lou Voss	98006...	357-989-47...	02FEB1967	M	Some high sch...	$75,000 to $99,999	Widowed	VOSS	LOU	777 S. HARB...	Hyundai	Elantra	2010
13	ST. LOUIS	MO	Ms. Shannon...	63128	956-586-41...	04SEP1981	F	Bachelor's deg...	Less than $25,000	Single, never...	MAZO	MS.	1515 LORD...	BMW	X5	2006
14	OREM	UT	Pat Pietron	84058	456-854-52...	10JUL1977	M	Some high sch...	$75,000 to $99,999	Divorced	PIETRON	PAT	4305 CENTR...	Toyota	Camry	2012
15	ST LOUIS	MO	Rob Rubenst...	63021	297-073-42...	14OCT1968	M	Some high sch...	$35,000 to $49,999	Divorced	RUBENSTEI...	ROB	3939 RUFFI...	Hyundai	Accent	2013
16	ST. CHARLE...	MO	Samuel Harle...	63301	731-887-45...	29JUL1992	M	Master's degree	$200,000 or more	Married or dom...	HARLEEN	SAMUEL	102 ECHO G...	Honda	Accord	2011
17	REDWOOD...	CA	Sharon Mand...	94065	902-861-51...	28SEP1960	F	Bachelor's deg...	$200,000 or more	Married or dom...	MANDELBA...	SHARON	3540 WILSHI...	BMW	M6	2005
18	LIMA	OH	Sidney Tretter	45801	419-566-43...	15NOV1972	F	Master's degree	$50,000 to $74,999	Widowed	TRETTER	SIDNEY	161 NORTH...	Acura	MDX	2009

Figure 11. ADD_DEMO Results

Now we have all our matches in one data set. Remember that there are 30 records in the source DEMOGRAPHIC table, but only 21 records in the PURCHASE table. We wanted to enhance our DEMOGRAPHIC table with the PURCHASE items, but because the record counts are not equal, we know that some of our DEMOGRAPHIC records will not match against our PURCHASE table. Since this table has only our matches, we know that we are missing some records from the DEMOGRAPHIC table.

One final JOIN will provide us with a table that has both our DEMOGRAPHIC and PURCHASE data. Figure 13 has the final results:

```
proc sql;
 create table Final_Matches as
 select a.*, b.car_make, b.car_model, b.car_year
 from sgf.demographic_data a
 left join combine_all_matches b
 on a.name = b.name and a.state = b.state and a.phone = b.phone
  and upcase(a.address) = b.address
 ;
quit;
```

FINAL_MATCHES

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	NAME	ADDRESS	CITY	STATE	ZIP	PHONE	DOB	Gender	Education	Income_Level	Household	CAR_MAKE	CAR_MODEL	CAR_YEAR
1	Abigail Sargent	4211 S Rushford St	Palo Alto	CA	94303	860-952-3496	16APR1965	F	High School gradu...	Less than $25,000	Single, never marr...			.
2	Andre Hulbert	5024 Fairbanks W...	Sunnyvale	CA	94089	618-121-6649	30SEP1971	M	Bachelor's degree	$100,000 to $149, ...	Separated			.
3	Ashley Bey	P.O. Box 6239	N. Ridgeville	OH	44039	271-475-5054	14MAR1972	F	Bachelor's degree	$100,000 to $149, ...	Single, never marr...	BMW	M2	2014
4	Cathy Lapp	4400 NC Highway...	ST. LOUIS	MO	63134	718-922-0353	28JAN1982	F	Doctorate degree	$75,000 to $99,999	Widowed	Acura	RDX	2009
5	Cindy Prentiss	515 E. Broad St....	St Louis	MO	63146	949-161-1908	26JUN1974	F	Doctorate degree	$50,000 to $74,999	Divorced	Mercedes-Benz	C300	2007
6	David Grassi	824 Valerie Dr Uni...	Pomona	CA	91768	806-295-2544	20OCT1985	M	Bachelor's degree	Less than $25,000	Single, never marr...	Toyota	Camry	2012
7	Denise Nath	1215 N Caldwell St	New Haven	CT	06516	660-469-0704	12NOV1977	F	Some high school,...	Less than $25,000	Separated			.
8	Douglas Doty	406 Mcclure Cir	Minneapolis	MN	55431	850-323-7265	14FEB1970	M	Doctorate degree	$75,000 to $99,999	Divorced	Mercedes-Benz	E350	2014
9	E Fusco	2617 Ramsey Rd...	Herndon	VA	22070	500-881-3331	08JUN1983	M	Bachelor's degree	$75,000 to $99,999	Single, never marr...	Toyota	Tundra	2013
10	Eric Einhorn	23 S Saunders Rd	Hutchins	KS	67504-5282	530-406-2382	12FEB1973	M	Bachelor's degree	$75,000 to $99,999	Single, never marr...	Hyundai	Tuscon	2008
11	Gary Stratman	5153 Camino Ruiz	Sunnyvale	CA	94086	350-964-1700	12FEB1974	M	Bachelor's degree	$100,000 to $149, ...	Married or domest...			.
12	Gwen Story	2120 Raven Glass...	PHILADELPHIA	PA	19178-4955	284-565-7463	17OCT1976	F	Some high school,...	$25,000 to $34,999	Married or domest...			.
13	Jessica Macias	5230 Walnut Grov...	Sherman	TX	75090-4440	738-818-2196	20JUN1977	F	Bachelor's degree	$100,000 to $149, ...	Widowed	Acura	MDX	2010
14	Johnathon Soon	239 N Edgeworth...	St Louis	MO	63104	615-005-6993	17JUN1976	F	Master's degree	$100,000 to $149, ...	Married or domest...	Hyundai	Santa Fe	2013
15	Jonathan Mc Gee	4900 Rivergrade...	Bannockburn	IL	60015	843-042-5960	19OCT1974	F	Bachelor's degree	$150,000 to $199, ...	Married or domest...	Honda	Accord	2015
16	Jose Rochford	2800 Woodlawn D...	Ballwin	MO	63011	721-144-7436	28OCT1970	M	High School gradu...	$150,000 to $199, ...	Divorced			.
17	K Sekeres	5541 Central Ave...	Marlow	NH	03456	991-312-5527	06AUG1971	F	Master's degree	$50,000 to $74,999	Widowed	Acura	RDX	2009
18	Kristina Radley	4430 E Greensbor...	Kansas City	MO	64141 6267	979-842-2568	18NOV1979	F	Doctorate degree	$200,000 or more	Married or domest...	Acura	TLX	2015
19	Lou Voss	777 S. Harbor Blvd.	Bellevue	WA	98006-1800	357-989-4735	02FEB1967	M	Some high school,...	$75,000 to $99,999	Widowed	Hyundai	Elantra	2010
20	Margaret Muench	3001 W Mission R...	Dallas	TX	75247	883-271-9095	15AUG1969	F	Doctorate degree	$75,000 to $99,999	Separated			.
21	Michelle Wan	4000 East Sky Ha...	Chesterfield	MO	63005	426-758-4180	06MAR1974	F	Some high school,...	$75,000 to $99,999	Single, never marr...			.
22	Miranda Andre	510 LightHouse A...	Cupertino	CA	95014	620-959-5034	13JAN1976	F	Doctorate degree	$200,000 or more	Divorced			.
23	Ms. Shannon Mazo	1515 Lord Ashley...	St. Louis	MO	63128	956-586-4147	04SEP1981	F	Bachelor's degree	Less than $25,000	Single, never marr...	BMW	X5	2006
24	Pat Pietron	4305 Central Ave...	Orem	UT	84058	456-854-5248	10JUL1977	M	Some high school,...	$75,000 to $99,999	Divorced	Toyota	Camry	2012
25	Phillip Gerstle	PO Box 13507	Skokie	IL	60076-2999	360-681-2934	09JUL1967	M	Master's degree	$150,000 to $199, ...	Divorced	Acura	MDX	2011
26	Rob Rubenstein	3939 Ruffin Rd	St Louis	MO	63021	297-073-4204	14OCT1968	M	Some high school,...	$35,000 to $49,999	Divorced	Hyundai	Accent	2013
27	Samuel Harleen	102 Echo Glen Dr	St. Charles	MO	63301	731-887-4557	29JUL1992	M	Master's degree	$200,000 or more	Married or domest...	Honda	Accord	2011
28	Sean Nugent	1993 Ernsford Dr	St. Louis	MO	63114	495-764-8564	09NOV1982	M	High School gradu...	$25,000 to $34,999	Divorced	Toyota	Corolla	2011
29	Sharon Mandelba...	3540 Wilshire Blvd	Redwood Shores	CA	94065	902-861-5137	28SEP1960	F	Bachelor's degree	$200,000 or more	Married or domest...	BMW	M6	2005
30	Sidney Tretter	161 Northfork Rd	Lima	OH	45801	419-566-4321	15NOV1972	F	Master's degree	$50,000 to $74,999	Widowed	Acura	MDX	2009

Figure 12. Final Scoring Results

FUZZY MATCHING USING SAS DATA QUALITY SERVER MACROS AND FUNCTIONS

Our next method leverages the SAS® Data Management solution, which includes standardization and a fuzzy matching algorithm as its key functionalities. Specifically, there are two products that address it: SAS® Data Management Studio (formerly known as DataFlux® Data Management Studio) and SAS® Data Quality Server.

Note: In addition to Base SAS or SAS® Enterprise Guide®, a separate license is required for SAS Data Quality Server.

SAS Data Management Studio has a graphical user interface (GUI), whereas SAS Data Quality Server allows SAS programmers to use SAS language elements, such as functions and procedures, to perform data quality operations (*SAS 9.4 Data Quality: Reference* 2015). This section focuses on SAS Data Quality Server functions to parse the name and address strings and create match codes that can be used to join data sets if there are no consistent merge keys. While less known and often overlooked, those functions are very powerful, and easy to use.

KEY TERMS

When working with SAS Data Quality Server functions and procedures, it is important to know and understand the following key terms:

- **SAS®Quality Knowledge Base** (QKB) contains algorithms to identify, standardize, and perform fuzzy matching on data fields (*DataFlux Data Management Studio: Essentials Course Notes* 2014).
- **Locale** can be described as a geographic region that shares both common language and consistent standards for certain data elements, such as names and addresses. At times, QKB and locale are used interchangeably.
- **Definitions** categorize the type of data, and are dependent on the selected locale. Examples are individual names, organizations, and address.
- **Token** is a term used to describe components of a data field. The data field can consist of one or more tokens. For example, a typical US address consists of the following tokens: street number, street name, pre-direction, post-direction, type of street, and so on.
- **Match Codes** are encoded character values. They are generated based on locale, definition, and sensitivity level.
- **Sensitivity** indicates the amount of information included in the match code (*SAS 9.4 Data Quality: Reference* 2015). The sensitivity ranges from 50 to 95, with 85 being the default value.

To explain the SAS Data Quality Server inner workings in layman's terms, we can think of QKB as the brain that tells SAS Data Quality Server how to perform data standardization and fuzzy matching. The locale is the language in which QKB is thinking. It is smart enough to differentiate between dialects used in different countries. For example, a separate locale is used for the United States and United Kingdom. While both countries use the same language, that is, English, there are differences in naming and address conventions that need to be taken into account. Once locale is selected, QKB will know how to categorize data elements, such as individual names, organization names, addresses, and so on, correctly. SAS Data Quality Server refers to this type of data categorization as definitions.

When the locale is selected and definitions are provided, QKB knows how to break each definition down into separate components, known as tokens, and to apply data standardization to each parsed token. In addition to parsing and data standardization, QKB uses its own "secret sauce" algorithms to perform fuzzy matching by creating match codes. Users have a certain degree of leeway to control how exact they want matching values to be. This is done by adjusting match-code sensitivity.

IMPLEMENTATION

The use of SAS Data Quality Server functions in SAS programs usually follows a series of steps. Each of the steps is described in detail below as applicable to our demo data sets and the stated objective.

Step 1. Load QKB

SAS Data Quality Server functions are similar to Base SAS functions. However, there is one prerequisite. At the start of the program, Quality Knowledge Base for a given locale has to be loaded into memory. More than one locale can be loaded at the same time. SAS Data Quality Server provides an autocall macro, DQLOAD**,** to load QKB package. The macro requires two parameters: locale (ENUSA in our example) and the path to the QKB physical location, which can vary depending on the product installation. The code is shown below:

```
%DQLOAD(DQLOCALE=(ENUSA), DQSETUPLOC='C:\Program
Files\SASHome\SASFoundation\9.4\dquality\sasmisc\QltyKB\sample');
```

SAS Data Quality Server installation comes with a sample QKB package appropriate for your region. It is recommended to replace the sample QKB with the latest version available on the SAS download website (http://support.sas.com/downloads/browse.htm?fil=&cat=540).

Note: The QKB package available from the website defaults to the ENUSA locale. To obtain the locale for a different region, please contact your account representative.

Step 2. Display Selected Locale Information

This step leverages another SAS Data Quality Server autocall macro, DQPUTLOC. The macro generates information about available data operations and corresponding functions as well as definition names and tokens to be used in SAS Data Quality Server functions. While this step is optional, programmers new to SAS Data Quality Server will find this macro extremely helpful. The syntax is as follows:

```
%DQPUTLOC(Locale);
```

In our example we used ENUSA locale, so the syntax is as follows:

```
%DQPUTLOC(ENUSA);
```

The macro output is written both to the SAS log and to the SAS data set, WORK.DEFINITIONS, as shown in Figure 13:

```
/*-------------------------------------------------------------------*/
/* PARSE DEFINITION(S)                                               */
/*                                                                   */
/* Parse definitions are used by the following:                      */
/*    dqParse function                                               */
/*    dqParseInfoGet function                                        */
/*    dqParseTokenGet function                                       */
/*    dqParseTokenPut function                                       */
/*    dqToken function                                               */
/*-------------------------------------------------------------------*/
    Address
       Token: Street Number
       Token: Pre-direction
       Token: Street Name
       Token: Street Type
       Token: Post-direction
       Token: Address Extension
       Token: Address Extension Number
```

Program* | Log | Output Data

Filter and Sort | Query Builder | Data | Describe | Graph | Analy

	operation	defnname	token	parsedefn
31	PARSE	Address	Street Number	
32	PARSE	Address	Pre-direction	
33	PARSE	Address	Street Name	
34	PARSE	Address	Street Type	
35	PARSE	Address	Post-direction	
36	PARSE	Address	Address Extensi...	
37	PARSE	Address	Address Extensi...	
38	PARSE	Address (Global)	Building Name	

Figure 13: Autocall Macro %DQPUTLOC Output

Step 3. Standardize Data Elements in the Original Data Sets

In our next step we will be standardizing data elements in the original data sets that will be used as merge keys to join the tables. In our example we want to join the DEMOGRAPHIC table with the PURCHASE table by names, addresses, and zip.

Two SAS Data Quality Server functions, DQPARSE and DQPARSETOKENGET, working in conjunction with each other, are specifically designed to help with data standardization. The functions are described below:

DQPARSE Function

This function has the following syntax:

DQPARSE (Field Name, ‘Definition’, ‘Locale’)

This function returns a parsed character value. The parsed character field inserts delimiters to “identify the elements in the value that correspond to the tokens that are enabled by the parse definition” (“DQPARSE Function,” *SAS 9.4 Data Quality: Reference* 2015).

The code used in our example is shown below:

```
  data DEMOGRAPHIC_DATA_STD;
   set DQDEMO.DEMOGRAPHIC_DATA;

/** Standardize the name field **/
   ParsedValueName=dqParse(Name,'Name','ENUSA');

/** Standardize the address field **/
   ParsedValueAdd=dqParse(address,'Address','ENUSA');
  run;

  data PURCHASE_DATA_STD;
   set DQDEMO.PURCHASE_DATA;

/** Standardize the name field **/
   ParsedValueName=dqParse(Name,'Name','ENUSA');

/** Standardize the address field **/
   ParsedValueAdd=dqParse(address,'Address','ENUSA');

  run;
```

The parsed character values returned by the DQPARSE function for the name and address fields are stored in the fields PARSEDVALUENAME and PARSEDVALUEADD as shown in Figure 14:

NAME	ADDRESS	ParsedValueName	ParsedValueAdd
Miranda Andre	510 LightHouse Ave.	/=/Miranda/=//=/Andre/=//=/	510/=//=/LightHouse/=/Ave./=//=//=/
Johnathon Soon	239 N Edgeworth St	/=/Johnathon/=//=/Soon/=//=/	239/=/N/=/Edgeworth/=/St/=//=//=/
Pat Pietron	4305 Central Ave West	/=/Pat/=//=/Pietron/=//=/	4305/=//=/Central/=/Ave/=/West/=//=/
Denise Nath	1215 N Caldwell St	/=/Denise/=//=/Nath/=//=/	1215/=/N/=/Caldwell/=/St/=//=//=/
Kristina Radley	4430 E Greensboro Chapel Hill Rd	/=/Kristina/=//=/Radley/=//=/	4430/=/E/=/Greensboro Chapel Hill/=/Rd/=//=//=/
Ms. Shannon Ma...	1515 Lord Ashley Dr	Ms./=/Shannon/=//=/Mazo/=//=/	1515/=//=/Lord Ashley/=/Dr/=//=//=/
Cathy Lapp	4400 NC Highway, PO Box 44	/=/Cathy/=//=/Lapp/=//=/	4400/=//=/NC/=/Highway/=//=/PO Box/=/44
Sean Nugent	1993 Ernsford Dr	/=/Sean/=//=/Nugent/=//=/	1993/=//=/Ernsford/=/Dr/=//=//=/
E Fusco	2617 Ramsey Rd #8	/=/E/=//=/Fusco/=//=/	2617/=//=/Ramsey/=/Rd/=//=//=/#8
Samuel Harleen	102 Echo Glen Dr	/=/Samuel/=//=/Harleen/=//=/	102/=//=/Echo Glen/=/Dr/=//=//=/
Douglas Doty	406 Mcclure Cir	/=/Douglas/=//=/Doty/=//=/	406/=//=/Mcclure/=/Cir/=//=//=/
Sidney Tretter	161 Northfork Rd	/=/Sidney/=//=/Tretter/=//=/	161/=//=/Northfork/=/Rd/=//=//=/
Eric Einhorn	23 S Saunders Rd	/=/Eric/=//=/Einhorn/=//=/	23/=/S/=/Saunders/=/Rd/=//=//=/
Gwen Story	2120 Raven Glass Pl	/=/Gwen/=//=/Story/=//=/	2120/=//=/Raven Glass/=/Pl/=//=//=/
Jessica Macias	5230 Walnut Grove Ln	/=/Jessica/=//=/Macias/=//=/	5230/=//=/Walnut Grove/=/Ln/=//=//=/
David Grassi	824 Valerie Dr Unit 30	/=/David/=//=/Grassi/=//=/	824/=//=/Valerie/=/Dr/=//=/Unit/=/30

Figure 14: Function DQPARSE Output Results

The fields PARSEDVALUENAME and PARSEDVALUEADD in Figure 14 contain respective tokens of the name and address separated by the delimiter "/=/ ".

As we mentioned above, the tokens depend on the selected locale and definitions. For example, the "Name" definition in ENUSA locale includes the following six tokens: Name Prefix, Given Name, Middle Name, Family Name, Name Suffix, and Name Appendage.

DQPARSETOKENGET Function

This function parses a token from the parsed character field and has the following syntax:

DQPARSETOKENGET (Parsed Field Name, 'Token', 'Definition', 'Locale')

The autocall macro, DQPUTLOC, described earlier, provides information about available definitions and corresponding tokens to be used as parameters both for DQPARSE and DQPARSETOKENGET functions.

The code used in our example on DEMOGRAPHIC table is shown below:

```
  data DEMOGRAPHIC_DATA_STD;
   set DQDEMO.DEMOGRAPHIC_DATA;

 /** Standardize the name field **/
   format FirstName LastName $40.;
   ParsedValueName=dqParse(Name,'Name','ENUSA');
   FirstName=dqParseTokenGet(ParsedValueName,'Given Name', 'Name','ENUSA');
   LastName=dqParseTokenGet(ParsedValueName,'Family Name','Name','ENUSA');

 /** Standardize the address field **/
   format StreetNumber $10. StreetName $40. StreetType $1. StreetPreDir
  StreetPostDir $8. StreetExt StreetExtNum $15.;
   ParsedValueAdd=dqParse(address,'Address','ENUSA');
   StreetNumber=dqParseTokenGet(ParsedValueAdd,'Street
  Number','Address','ENUSA');
   StreetName=dqParseTokenGet(ParsedValueAdd,'Street
  Name','Address','ENUSA');
   StreetType=dqParseTokenGet(ParsedValueAdd,'Street
  Type','Address','ENUSA');
   StreetPreDir=dqParseTokenGet(ParsedValueAdd,'Pre-
  Direction','Address','ENUSA');
   StreetPostDir=dqParseTokenGet(ParsedValueAdd,'Post-
  Direction','Address','ENUSA');
   StreetExt=dqParseTokenGet(ParsedValueAdd,'Address
  Extension','Address','ENUSA');
```

```
 StreetExtNum=dqParseTokenGet(ParsedValueAdd,'Address Extension
Number','Address','ENUSA');

run;
```

The outputs of the DQPARSETOKENGET function are stored in respective fields. See Figure 15:

NAME	ParsedValueName	FirstName	LastName	NameStd
Miranda Andre	/=/Miranda/=//=/Andre/...	Miranda	Andre	Miranda Andre
Johnathon Soon	/=/Johnathon/=//=/Soo...	Johnathon	Soon	Johnathon Soon
Pat Pietron	/=/Pat/=//=/Pietron/=//=/	Pat	Pietron	Pat Pietron
Denise Nath	/=/Denise/=//=/Nath/=//...	Denise	Nath	Denise Nath
Kristina Radley	/=/Kristina/=//=/Radley...	Kristina	Radley	Kristina Radley
Ms. Shannon Ma...	Ms./=/Shannon/=//=/M...	Shannon	Mazo	Shannon Mazo
Cathy Lapp	/=/Cathy/=//=/Lapp/=//=/	Cathy	Lapp	Cathy Lapp
Sean Nugent	/=/Sean/=//=/Nugent/=...	Sean	Nugent	Sean Nugent
E Fusco	/=/E/=//=/Fusco/=//=/	E	Fusco	E Fusco
Samuel Harleen	/=/Samuel/=//=/Harlee...	Samuel	Harleen	Samuel Harleen
Douglas Doty	/=/Douglas/=//=/Doty/=...	Douglas	Doty	Douglas Doty
Sidney Tretter	/=/Sidney/=//=/Tretter/...	Sidney	Tretter	Sidney Tretter
Eric Einhorn	/=/Eric/=//=/Einhorn/=//...	Eric	Einhorn	Eric Einhorn
Gwen Story	/=/Gwen/=//=/Story/=//...	Gwen	Story	Gwen Story
Jessica Macias	/=/Jessica/=//=/Macias...	Jessica	Macias	Jessica Macias
David Grassi	/=/David/=//=/Grassi/=/...	David	Grassi	David Grassi

ADDRESS	ParsedValueAdd	StreetNumber	StreetName	StreetType	StreetPreDir	StreetPostDir	StreetExt	StreetExtNum	AddressStd
3540 Wilshire Blvd	3540/=/=/Wilshire/=/Blvd/=//=//=/	3540	Wilshire	B					3540 Wilshire B
4211 S Rushford St	4211/=/S/=/Rushford/=/St/=//=//=/	4211	Rushford	S	S				4211 S Rushford S
777 S. Harbor Blvd.	777/=/S./=/Harbor/=/Blvd./=//=//=/	777	Harbor	B	S.				777 S. Harbor B
PO Box 13507	/=//=//=//=//=/PO Box/=/13507						PO Box	13507	PO Box 13507
3939 Ruffin Rd	3939/=//=/Ruffin/=/Rd/=//=//=/	3939	Ruffin	R					3939 Ruffin R
3001 W Mission Rd Ste A	3001/=/W/=/Mission/=/Rd/=//=/Ste/=/A	3001	Mission	R	W		Ste	A	3001 W Mission R Ste A
2800 Woodlawn Dr, Apt 23B	2800/=//=/Woodlawn/=/Dr/=//=/Apt/=/23B	2800	Woodlawn	D			Apt	23B	2800 Woodlawn D Apt 23B
5541 Central Avenue	5541/=//=/Central/=/Avenue/=//=//=/	5541	Central	A					5541 Central A
510 LightHouse Ave.	510/=//=/LightHouse/=/Ave./=//=//=/	510	LightHouse	A					510 LightHouse A
239 N Edgeworth St	239/=/N/=/Edgeworth/=/St/=//=//=/	239	Edgeworth	S	N				239 N Edgeworth S
4305 Central Ave West	4305/=//=/Central/=/Ave/=/West/=//=/	4305	Central	A		West			4305 Central A West
1215 N Caldwell St	1215/=/N/=/Caldwell/=/St/=//=//=/	1215	Caldwell	S	N				1215 N Caldwell S
4430 E Greensboro Chapel Hill Rd	4430/=/E/=/Greensboro Chapel Hill/=/Rd/=//=//=/	4430	Greensboro Cha...	R	E				4430 E Greensboro Chapel Hill R
1515 Lord Ashley Dr	1515/=//=/Lord Ashley/=/Dr/=//=//=/	1515	Lord Ashley	D					1515 Lord Ashley D

Figure 15: Function DQPARSETOKENGET Output Results

We then make similar changes to the PURCHASE data. The tokens are parsed from the name and address fields and stored in the corresponding fields: FIRSTNAME, LASTNAME, STREETNUMBER, STREETNAME, STREETTYPE, STREETPREDIR, STREETPOSTDIR, STREETEXT, and STREETEXTNUM.

With the individual components of the name and address fields having been standardized, we can now create two new fields, NAMESTD and ADDRESSSTD, by concatenating the outputs of DQPARSETOKENGET function as shown in the code snippet below:

```
NameStd=Catx('',FirstName,LastName);
AddressStd=Catx('',StreetNumber,StreetPreDir,StreetName,StreetType,StreetPo
stDir,StreetExt,StreetExtNum);
```

Step 3. Create Match Codes

Now we are ready to create match codes based on two new fields, NAMESTD and ADDRESSSTD. As we explained above, match codes are encoded character values generated by the SAS Data Quality Server algorithm based on the specified locale, definition, and sensitivity level. DQMATCH is a function that generates match codes in SAS Data Quality Server and has the following syntax:

DQMATCH (Field Name, 'Definition', Sensitivity, 'Locale')

The sensitivity parameter indicates the amount of information included in the match code. The higher sensitivity, the more exact we want our matches to be ("DQMATCH Function," *SAS 9.4 Data Quality: Reference* 2015). The match-code sensitivity ranges from 50 to 95, with 85 being the default value.

You can see the code snippet below and the output of DQMATCH function in Figure 16:

```
/**Create match codes **/
    NameStdMatchCode=dqmatch(NameStd,'Name',85,'ENUSA');
    AddressStdMatchCode=dqmatch(AddressStd,'Address',85,'ENUSA');
```

NAME	NameStd	NameStdMatchCode	ADDRESS	AddressStd	AddressStdMatchCode
Margaret Muench	Margaret Muench	BPJ2$$$BYF7$$$$	3001 W Mission Rd Ste A	3001 W Mission R Ste A	K00Z$B4PY&$$$$$
Jose Rochford	Jose Rochford	YJ2GY~$C4$$$$$$	2800 Woodlawn Dr, Apt 23B	2800 Woodlawn D Apt 23B	HD00$L8WPHKM$$$
K Sekeres	K Sekeres	43Y4$$$3$$$$$$$	5541 Central Avenue	5541 Central A	55SZ$4P~Y&$$$$$
Andre Hulbert	Andre Hulbert	2WMY~$$&P8Y$$$$	5024 Fairbanks Way	5024 Fairbanks W	50HS$GYMP$$$$$$
Jonathan Mc Gee	Jonathan Mc Gee	B3F$$$$C@P$$$$$	4900 Rivergrade Rd	4900 Rivergrade R	S-00$YVYF$$$$$$
Miranda Andre	Miranda Andre	&P8Y$$$BYP8$$$$	510 LightHouse Ave.	510 LightHouse A	5Z0$$W~4$&$$$$$
Johnathon Soon	Johnathon Soon	4P$$$$$C@P$$$$$	239 N Edgeworth St	239 N Edgeworth S	HK-$$_CLY$$$$$$
Pat Pietron	Pat Pietron	N~YP$$$N&~$$$$$	4305 Central Ave West	4305 Central A West	SK05$4P~Y$$$$$$
Denise Nath	Denise Nath	P~2$$$$8_P7$$$$	1215 N Caldwell St	1215 N Caldwell S	ZHZ5$3W8L$$$$$$
Kristina Radley	Kristina Radley	Y8W$$$$JY74$$$$	4430 E Greensboro Chapel Hill Rd	4430 E Greensboro Chapel Hill..	SSK0$FYP4$$$$$$
Ms. Shannon Mazo	Shannon Mazo	B4$$$$$42PP$$$$	1515 Lord Ashley Dr	1515 Lord Ashley D	Z5Z5$WY~48$$$$$

Figure 16: Function DQMATCH Output Results

As mentioned above, match codes are generated by the SAS Data Quality Server internal algorithm and are not intended to be dissected and interpreted. Match codes need to be created for both DEMOGRAPHIC and PURCHASE tables.

Step 4. Join the Data Sets

We can now join two data sets, DEMOGRAPHIC and PURCHASE, by three fields: match codes for names, match codes for addresses, and standardized zip code as shown below:

```
/** Merge by Name Match Code, Address Match Code, and Standardized Zip Code **/
  proc sql;
   create table FinalMatches_DQ as
   select a.Name,a.NameStd,
            a.Address,a.AddressStd,
            a.City,a.State,a.Zip,a.ZipStd,
          a.Gender,a.Education,a.Income_Level,a.Household,
            b.Car_Make,b.Car_Model,b.Car_Year
   from DEMOGRAPHIC_DATA_STD as a
   left join
        PURCHASE_DATA_STD    as b
   on    a.NameStdMatchCode=b.NameStdMatchCode and
         a.AddressStdMatchCode=b.AddressStdMatchCode and
         a.ZipStd=b.ZipStd;
  quit;
```

Note: The original zip code field was standardized using two Base SAS functions such as Left() and Substr(), that is ZipStd=substr(left(Zip),1,5).

Looking at the output data set FINALMATCHES_DQ, we can see that 20 matching records were found between DEMOGRAPHIC and PURCHASE tables. SAS Data Quality Server functions correctly addressed data inconsistencies such as “Sam” versus “Samuel”, “St” versus “Street”, “Ave” versus “Avenue”, “Cir” v “Circle”, “P.O.” versus “PO”, “Ste” versus “Suite”, “Dave” versus “David”, “Ln” versus “Lane,” “Ms. Shannon” versus “Shannon”, “Patrick” versus “Pat”, “Rob” versus “Robert” as well as misspellings such as “Erick” versus “Eric”, “Sreet” versus “Street”, and “Mandelbaum” versus “Mandelbam”.

While successful in identifying most common data inconsistencies, SAS Data Quality Server does not always find every match. In our example, SAS Data Quality Server didn’t recognize that “K Sekeres” and “Kurt Sekeres” refer to the same individual. While the sensitivity level can be lowered to cast a wider net, this approach comes with a trade-off. The wider net can introduce false positive matches, and potentially lead to biased analytic results. Often, the data cleaning process becomes an iterative process of adjusting sensitivity level, selecting different candidates for match codes, and assessing false positives and false negatives with each iteration.

Step 5. Unload QKB

This final step removes QKB from the memory. It should be run after all data quality steps have been completed to free up the memory. SAS Data Quality Server supplies another autocall macro, DQUNLOAD, to perform this task. The macro has the following syntax:

```
%DQUNLOAD;
```

PROS AND CONS OF EACH APPROACH

Each of two approaches described in this paper has its pros and cons. The scoring method is highly customizable and fairly robust. Programmers can choose which variables to use as merge keys, what point values to assign to each variable, and what the overall threshold level is. In our example, the scoring method actually did pick up one more record than the SAS Data Quality Server method. However, the scoring method requires intermediate to advanced SAS programming skills to understand the logic, and to know how to modify the code for different data quality situations. Another limitation comes with parsing certain data elements. For example, splitting a name into individual components such as title, first name, middle name, last name, and name suffix can be very challenging and time consuming to develop, especially if not all values are consistently populated.

The second approach, the SAS Data Quality Server method, does not require advanced programming skills. It is very straightforward as it relies on a set of specific macros and functions, specifically designed to address common data quality issues and help with fuzzy matching. To use the SAS Data Quality Server method, your organization needs an additional license, and you need to be willing to rely on the “black box,” that is, the internal SAS Data Quality Server algorithm, which can’t be directly modified. Finally, parsing and creating match codes are very intensive CPU processes, and CPU requirements need to be considered when using SAS Data Quality Server functions on large data sets.

CONCLUSION

Data analytics has been gaining momentum, and will continue to be a major trend in the future. Organizations of all sizes, large and small, have been accumulating data and are eager to put it to work to support their day-to-day decision making. Many learn that fusing data from different data systems is often a daunting task when data sources lack consistent merge keys. When names, addresses, dates of births, and so on are the only common fields between the data sets, the task of creating a master record demands creativity and application of fuzzy matching methods. Two methods described in this paper have been successfully deployed in real world situations to create master records if there are no consistent merge fields. Now you too can go forth and find your match!

REFERENCES

Wikipedia 2016. “Approximate string matching.” Accessed January 28, 2016. Available https://en.wikipedia.org/wiki/Approximate_string_matching

Russell, Kevin. “How to perform a fuzzy match using SAS functions.” SAS Blogs. Accessed January 28 2016. Available http://blogs.sas.com/content/sgf/2015/01/27/how-to-perform-a-fuzzy-match-using-sas-functions/

McAllaster, Doug. “DATA STEP vs PROC SQL in a Many-to-Many Match-Merge.” Lex Janson 2016. Available http://www.lexjansen.com/nesug/nesug97/sassolu/mcallast.pdf

SAS Institute Inc. 2015. *SAS 9.4 Data Quality: Reference.* 2d ed. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/cdl/en/dqclref/68376/PDF/default/dqclref.pdf.

SAS Institute Inc. 2014. *DataFlux Data Management Studio: Essentials Course Notes.* Available https://www.sas.com/store/books/categories/course-notes/dataflux-data-management-studio-essentials-course-notes/prodCN_E70509_en.html.

SAS Institute Inc. 2015. “DQPARSE Function.” *SAS 9.4 Data Quality: Reference.* 2d ed. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/cdl/en/dqclref/68376/HTML/default/viewer.htm#p0p4cunbfsorb6n1msv5bga88wdh.htm.

SAS Institute Inc. 2015. “DQMATCH Function.” *SAS 9.4 Data Quality: Reference.* 2d ed. Cary, NC: SAS Institute Inc. Available http://support.sas.com/documentation/cdl/en/dqclref/68376/HTML/default/viewer.htm#p09nffezbjyj4on11oblz77aq1x6.htm.

APPENDIX

DEMO DATA SETS

Demographic Data

NAME	ADDRESS	CITY	STATE	ZIP	PHONE	DOB	Gender	Education	Income_Level	Household
Sharon Mandelb...	3540 Wilshire Bl...	Redwood Shores	CA	94065	902-861-5137	28SEP1960	F	Bachelor's degree	$200,000 or more	Married or dome...
Abigail Sargent	4211 S Rushford...	Palo Alto	CA	94303	860-952-3496	16APR1965	F	High School grad...	Less than $25,000	Single, never ma...
Lou Voss	777 S. Harbor Bl...	Bellevue	WA	98006-1800	357-989-4735	02FEB1967	M	Some high scho...	$75,000 to $99,9...	Widowed
Phillip Gerstle	PO Box 13507	Skokie	IL	60076-2999	360-681-2934	09JUL1967	M	Master's degree	$150,000 to $199...	Divorced
Rob Rubenstein	3939 Ruffin Rd	St Louis	MO	63021	297-073-4204	14OCT1968	M	Some high scho...	$35,000 to $49,9...	Divorced
Margaret Muench	3001 W Mission...	Dallas	TX	75247	883-271-9095	15AUG1969	F	Doctorate degree	$75,000 to $99,9...	Separated
Jose Rochford	2800 Woodlawn...	Ballwin	MO	63011	721-144-7436	28OCT1970	M	High School grad...	$150,000 to $199...	Divorced
K Sekeres	5541 Central Av...	Marlow	NH	03456	991-312-5527	06AUG1971	F	Master's degree	$50,000 to $74,9...	Widowed
Andre Hulbert	5024 Fairbanks...	Sunnyvale	CA	94089	618-121-6649	30SEP1971	M	Bachelor's degree	$100,000 to $149...	Separated
Ashley Bey	P.O. Box 6239	N. Ridgeville	OH	44039	271-475-5054	14MAR1972	F	Bachelor's degree	$100,000 to $149...	Single, never ma...
Gary Stratman	5153 Camino Ruiz	Sunnyvale	CA	94086	350-964-1700	12FEB1974	M	Bachelor's degree	$100,000 to $149...	Married or dome...
Michelle Wan	4000 East Sky H...	Chesterfield	MO	63005	426-758-4180	06MAR1974	F	Some high scho...	$75,000 to $99,9...	Single, never ma...
Cindy Prentiss	515 E. Broad St,...	St Louis	MO	63146	949-161-1908	26JUN1974	F	Doctorate degree	$50,000 to $74,9...	Divorced
Jonathan Mc Gee	4900 Rivergrade...	Bannockburn	IL	60015	843-042-5960	19OCT1974	F	Bachelor's degree	$150,000 to $199...	Married or dome...
Miranda Andre	510 LightHouse...	Cupertino	CA	95014	620-959-5034	13JAN1976	F	Doctorate degree	$200,000 or more	Divorced
Johnathon Soon	239 N Edgewort...	St Louis	MO	63104	615-005-6993	17JUN1976	F	Master's degree	$100,000 to $149...	Married or dome...
Pat Pietron	4305 Central Av...	Orem	UT	84058	456-854-5248	10JUL1977	M	Some high scho...	$75,000 to $99,9...	Divorced
Denise Nath	1215 N Caldwell...	New Haven	CT	06516	660-469-0704	12NOV1977	F	Some high scho...	Less than $25,000	Separated
Kristina Radley	4430 E Greensb...	Kansas City	MO	64141 6267	979-842-2568	18NOV1979	F	Doctorate degree	$200,000 or more	Married or dome...
Ms. Shannon Ma...	1515 Lord Ashle...	St. Louis	MO	63128	956-586-4147	04SEP1981	F	Bachelor's degree	Less than $25,000	Single, never ma...
Cathy Lapp	4400 NC Highwa...	ST. LOUIS	MO	63134	718-922-0353	28JAN1982	F	Doctorate degree	$75,000 to $99,9...	Widowed
Sean Nugent	1993 Ernsford Dr	St. Louis	MO	63114	495-764-8564	09NOV1982	M	High School grad...	$25,000 to $34,9...	Divorced
E Fusco	2617 Ramsey R...	Herndon	VA	22070	500-881-3331	08JUN1983	M	Bachelor's degree	$75,000 to $99,9...	Single, never ma...
Samuel Harleen	102 Echo Glen Dr	St. Charles	MO	63301	731-887-4557	29JUL1992	M	Master's degree	$200,000 or more	Married or dome...
Douglas Doty	406 Mcclure Cir	Minneapolis	MN	55431	850-323-7265	14FEB1970	M	Doctorate degree	$75,000 to $99,9...	Divorced
Sidney Tretter	161 Northfork Rd	Lima	OH	45801	419-566-4321	15NOV1972	F	Master's degree	$50,000 to $74,9...	Widowed
Eric Einhorn	23 S Saunders Rd	Hutchins	KS	67504-5282	530-406-2382	12FEB1973	M	Bachelor's degree	$75,000 to $99,9...	Single, never ma...
Gwen Story	2120 Raven Gla...	PHILADELPHIA	PA	19178-4955	284-565-7463	17OCT1976	F	Some high scho...	$25,000 to $34,9...	Married or dome...
Jessica Macias	5230 Walnut Gro...	Sherman	TX	75090-4440	738-818-2196	20JUN1977	F	Bachelor's degree	$100,000 to $149...	Widowed
David Grassi	824 Valerie Dr U...	Pomona	CA	91768	806-295-2544	20OCT1985	M	Bachelor's degree	Less than $25,000	Single, never ma...

Purchase Data

NAME	ADDRESS	CITY	STATE	ZIP	PHONE	CAR_MAKE	CAR_MODEL	CAR_YEAR
Sharon Mandelb...	3540 Wilshire Bl...	Redwood Shores	CA	94065	902-861-5137	BMW	M6	2005
Louis Voss	777 S. Harbor Bl...	Bellevu	Washington	98006-1800	357-989-4735	Hyundai	Elantra	2010
Phillip Gerstle	PO Box 13507	Skookie	IL	60076-2999	360-681-2934	Acura	MDX	2011
Robert Rubenste...	3939 Ruffin Road	Saint Louis	MO	63021	297-073-4204	Hyundai	Accent	2013
Kurt Sekeres	5541 Central Ave	Marlow	NH	03456	991-312-5527	Acura	RDX	2009
Ashley Bey	PO Box 6239	North Ridgeville	OH	44039	271-475-5054	BMW	M2	2014
Cindy Prentiss	515 E. Broad St,...	Saint Louis	MO	63146	949-161-1908	Mercedes-Benz	C300	2007
Jonathan McGee	4900 Rivergrade...	Bannockburn	IL	60015	843-042-5960	Honda	Accord	2015
Johnathon Soon	239 N Edgewort...	St Louis	Missori	63104	615-005-6993	Hyundai	Santa Fe	2013
Patrick Pietron	4305 Central Av...	Orem	Utah	84058	456-854-5248	Toyota	Camry	2012
Kristina Radley	4430 E Greensb...	Kansas City	MO	64141 6267	979-842-2568	Acura	TLX	2015
Shannon Mazo	1515 Lord Ashle...	SAINT. Louis	MO	63128	956-586-4147	BMW	X5	2006
Cathy Lap	4400 NC Highwa...	ST. LOUIS	MO	63134	718-922-0353	Acura	RDX	2009
Sean Nugent	1993 Ernsford Dr	St. Louis	MO	63114	495-764-8564	Toyota	Corolla	2011
E Fusco	2617 Ramsey R...	Herndon	Virgina	22070	500-881-3331	Toyota	Tundra	2013
Sam Harleen	102 Echo Glen Dr	St. Charles	MO	63301	731-887-4557	Honda	Accord	2011
Douglas Doty	406 Mcclure Circ...	Minneapolis	MN	55431	850-323-7265	Mercedes-Benz	E350	2014
Sidney Treter	161 Northfork Rd	Lima	OH	45801	419-566-4321	Acura	MDX	2009
Erick Einhorn	23 S Saunders Rd	Hutchins	KS	67504-5282	530-406-2382	Hyundai	Tuscon	2008
Jessica Macias	5230 Walnut Gro...	Sherman	Texas	75090-4440	738-818-2196	Acura	MDX	2010
Dave Grassi	824 Valerie Dr U...	Pomona	California	91768	806-295-2544	Toyota	Camry	2012

SAS PROGRAMS

SAS Scoring Code

```
libname sgf "C:\SGF 2016\Data";

/**************************************************/
/******************* DATA PREP *******************/
/**************************************************/

/* Parse out Name into First Name and Last Name */

/* Demographic Data */
data demo_standardize (rename=(address_keep = address ct = city st = state));
       format ct $14. st $2.;
       set sgf.demographic_data;
       last_name = upcase(scan(name, -1, ' ')); /* Negative value scans from right to
left */
       first_name = upcase(scan(name, 1, ' ')); /* Positive value scans from left to
right */
       address_keep = upcase(address);
       ct = upcase(city);
       st = upcase(state);
       drop address city state;
run;

/* Purchase Data */
data purchase_standardize (rename=(address_keep = address ct = city st = state));
       format ct $14. st $2.;
       set sgf.purchase_data;
       last_name = upcase(scan(name, -1, ' ')); /* Negative value scans from right to
left */
       first_name = upcase(scan(name, 1, ' ')); /* Positive value scans from left to
right */
       address_keep = upcase(address);
       ct = upcase(city);
       st = upcase(state);
       drop address city state;
run;

/*******************************************************/
/******************* SIMPLE MATCHES ********************/
/*******************************************************/

/* Do a first pass match on "high value" fields such a first name, last name, city,
state - even address
if your data has already been standardized. Use SQL or merge as appropraite. This
example uses a
merge statement */

proc sort data=demo_standardize; by first_name last_name address state phone; run;

proc sort data=purchase_standardize; by first_name last_name address state phone; run;

data demo_purchase_match
       demo_only (keep= first_name last_name address state phone) /* Trim down to
fields to match on */
       purchase_only (keep= first_name last_name address state phone);
       merge demo_standardize (in=a) purchase_standardize (in=b);
       by first_name last_name address state phone;
       if a and b then output demo_purchase_match;
       if a and not b then output demo_only;
       if b and not a then output purchase_only;
```

```
run;

/*********************************************************/
/******************* MATCH USING SCORING *******************/
/*********************************************************/

/* Now that the easy matches are out of the way, start using scoring methods */
/*  First name = 20                            Last Name = 20
       State = 10                                    Address = 20
       Phone = 10 */

/* All the data will end up on one row for us to compare, so rename the key fields
from one
of the data sets */
data demo_only_rename (rename=(first_name = first_name_d last_name = last_name_d state
= state_d
       address = address_d phone = phone_d));
       set demo_only;
run;

/* To compare and score the records a loop will be created - creat one table as the
source set,
and one as the lookup table. The max record count global macro that you'll need to
tell it when
to start and stop */
data _null_;
       %let dsid = %sysfunc( open(demo_only_rename) );
       %let demo_last_rec = %sysfunc(attrn(&dsid,nobs));
       rc = %sysfunc( close(&dsid) );
       %let dsid = %sysfunc(open(purchase_only));
       %let purchase_last_rec = %sysfunc(attrn(&dsid,nobs));
       rc = %sysfunc(close(&dsid));
run;


/* Throughout the loop we'll be writing out the matches. This makes sure the set is
null so that
if there are multiple runs during testing/debugging we don't have holdover records
sneaking in */
data match_vals;
       set _null_;
run;

/* This macro does the heavy lift. The first do loop is a %do loop. It cannot exist in
open code, so we
have to make this run as a macro.

The first %do loop will run through our formatted demographic table one line at a
time. The first
record will go through the data step and match process. If the record matches, it will
be written out
at the bottom of the loop, then go back up to the top starting on the second record
and repeating
the process. The &demo_last_rec variable tells it when to stop this process */
%macro match (table1, table2);

%do i = 1 %to &demo_last_rec.; /* Number of times loop will execute */;

data min_score_60 ;
       set &table1.; /* This is the table we derive the %do loop's last record from */
       if _n_ = &i; /* This tells us what record number we are testing. To debug,
simply comment out
              the %do loop at the top and replace this with the record number you want
```

```
to go through this
              process */
       do j = 1 to &purchase_last_rec. until (tot_score >=60); /* This is our second
loop */
       /* This loop goes through our second table - it takes that one line we get from
our "if _n_ = &i"
       and loops it through, comparing it to everythng in our second table */

              set &table2.; /* Note that this is the table that we did NOT rename the
fields on */
                     by first_name
                     last_name
                     address
                     state
                     phone;

                     if first.first_name or first.last_name or first.address or
first.state or first.phone then do;

       /* Needs to be in this do loop to reset the score with each comparision in the
second table */

                     first_name_score = 0;
                     last_name_score = 0;
                     address_score = 0;
                     state_score = 0;
                     phone_score = 0;

       /* Here we use COMPGED to help with our fuzzy matching. The lower the compged
score, the closer
       the names are. For example, comparing SAM to SAMUEL gives a compged score of
30.
       CATHY to CATHY gives a score of 0. Last names GEE to MCGEE gives 400.
       For this data, I am setting the first name to have more flexability by giving
it a higher COMPGED score. */

                     if COMPGED(first_name,first_name_d) <= 70 then first_name_score =
20;
                     if COMPGED(last_name,last_name_d) <= 30 then last_name_score = 20;
                     if COMPGED(address, address_d) <= 500 then address_score = 20;
                     if state = state_d then state_score = 10;
                     if phone = phone_d then phone_score = 10;

                     tot_score = first_name_score + last_name_score + address_score +
state_score + phone_score;
              end;

              if tot_score >= 60 then do;
                     output min_score_60;
              end;

end;
run;

data match_vals (drop=j tot_score first_name_score last_name_score address_score
state_score phone_score);
       set  min_score_60 match_vals;
run;
%end;
%mend match;

%match (demo_only_rename, purchase_only);
```

```
/* Takes our matching results and pulls in all the purchase information. */
proc sql;
	create table add_purchase as
	select a.car_make, a.car_model, a.car_year, b.*
	from purchase_standardize a, match_vals b
	where a.first_name = b.first_name and a.last_name = b.last_name and a.address =
b.address
		and a.state = b.state and a.phone = b.phone
;
quit;

/* Adds all the demographic information to our matches. Also, note that here I am
keeping the "_d"
values for Name and Address. There were differences in the names and addresses - you
need to decide
based on your data which values to keep. */
proc sql;
	create table add_demo as
	select a.*, b.car_make, b.car_model, b.car_year
	from demo_standardize a, add_purchase b
	where a.first_name = b.first_name_d and a.last_name = b.last_name_d and
a.address = b.address_d
		and a.state = b.state_d and a.phone = b.phone_d
;
quit;

/* Adds the matches we got at the beginning */
data combine_all_matches;
	set add_demo DEMO_PURCHASE_MATCH;
run;

/* Joins all of the matches to our source data set. All of the purchasers had
demographic
information, but not all of the demographic records had purchases. Keep in mind what
your data
is doing when you decide how to do these final joins */
proc sql;
	create table Final_Matches as
	select a.*, b.car_make, b.car_model, b.car_year
	from sgf.demographic_data a
	left join combine_all_matches b
	on a.name = b.name and a.state = b.state and a.phone = b.phone
		and upcase(a.address) = b.address
	;
quit;

/***** END PROGRAM *****/
```

SAS Data Quality Server Code

```
/** Check if the SAS Data Quality Software is installed **/

proc setinit; run;

/** Specify the library **/

libname DQDEMO 'C:\Users\elshte\OneDrive for Business\SGF 2016';

/** Load QKB for a specific locale **/

%DQLOAD(DQLOCALE=(ENUSA), DQSETUPLOC='C:\Program
Files\SASHome\SASFoundation\9.4\dquality\sasmisc\QltyKB\sample');
```

```
/** Display information about specified locale **/

%DQPUTLOC(ENUSA);

/** Parse and create match codes **/

data DEMOGRAPHIC_DATA_STD;
 set DQDEMO.DEMOGRAPHIC_DATA;

 /** Standardize the name field **/
 format FirstName LastName $40.;
 ParsedValueName=dqParse(Name,'Name','ENUSA');
 FirstName=dqParseTokenGet(ParsedValueName,'Given Name', 'Name','ENUSA');
 LastName=dqParseTokenGet(ParsedValueName,'Family Name','Name','ENUSA');
 NameStd=Catx('',FirstName,LastName);

 /** Standardize the address field **/
 format StreetNumber $10. StreetName $40. StreetType $1. StreetPreDir StreetPostDir
$8. StreetExt StreetExtNum $15.;
 ParsedValueAdd=dqParse(address,'Address','ENUSA');
 StreetNumber=dqParseTokenGet(ParsedValueAdd,'Street Number','Address','ENUSA');
 StreetName=dqParseTokenGet(ParsedValueAdd,'Street Name','Address','ENUSA');
 StreetType=dqParseTokenGet(ParsedValueAdd,'Street Type','Address','ENUSA');
 StreetPreDir=dqParseTokenGet(ParsedValueAdd,'Pre-Direction','Address','ENUSA');
 StreetPostDir=dqParseTokenGet(ParsedValueAdd,'Post-Direction','Address','ENUSA');
 StreetExt=dqParseTokenGet(ParsedValueAdd,'Address Extension','Address','ENUSA');
 StreetExtNum=dqParseTokenGet(ParsedValueAdd,'Address Extension
Number','Address','ENUSA');

AddressStd=Catx('',StreetNumber,StreetPreDir,StreetName,StreetType,StreetPostDir,Stree
tExt,StreetExtNum);
 ZipStd=substr(left(Zip),1,5);

 /**Create match codes **/
 NameStdMatchCode=dqmatch(NameStd,'Name',85,'ENUSA');
 AddressStdMatchCode=dqmatch (AddressStd,'Address',85,'ENUSA');

 Drop ParsedValueName ParsedValueAdd;

run;

data PURCHASE_DATA_STD;
 set DQDEMO.PURCHASE_DATA;

 /** Standardize the name field **/
 format FirstName LastName $40.;
 ParsedValueName=dqParse(Name,'Name','ENUSA');
 FirstName=dqParseTokenGet(ParsedValueName,'Given Name', 'Name','ENUSA');
 LastName=dqParseTokenGet(ParsedValueName,'Family Name','Name','ENUSA');
 NameStd=Catx('',FirstName,LastName);

 /** Standardize the address field **/
 format StreetNumber $10. StreetName $40. StreetType $1. StreetPreDir StreetPostDir
$8. StreetExt StreetExtNum $15.;
 ParsedValueAdd=dqParse(address,'Address','ENUSA');
 StreetNumber=dqParseTokenGet(ParsedValueAdd,'Street Number','Address','ENUSA');
 StreetName=dqParseTokenGet(ParsedValueAdd,'Street Name','Address','ENUSA');
 StreetType=dqParseTokenGet(ParsedValueAdd,'Street Type','Address','ENUSA');
 StreetPreDir=dqParseTokenGet(ParsedValueAdd,'Pre-Direction','Address','ENUSA');
 StreetPostDir=dqParseTokenGet(ParsedValueAdd,'Post-Direction','Address','ENUSA');
 StreetExt=dqParseTokenGet(ParsedValueAdd,'Address Extension','Address','ENUSA');
 StreetExtNum=dqParseTokenGet(ParsedValueAdd,'Address Extension
```

```
Number','Address','ENUSA');
 AddressStd=Catx('',StreetNumber,StreetPreDir,StreetName,StreetType, StreetPostDir,
StreetExt, StreetExtNum);
 ZipStd=substr(left(Zip),1,5);


/** Create match codes **/

 NameStdMatchCode=dqmatch(NameStd,'Name',85,'ENUSA');
 AddressStdMatchCode=dqmatch(AddressStd,'Address',85,'ENUSA');

 Drop ParsedValueName ParsedValueAdd;

run;

/** Merge by Name Match Code, Address Match Code, and Standardized Zip Code **/

proc sql;
 create table FinalMatches_DQ as
 select a.Name,a.NameStd,
             a.Address,a.AddressStd,
             a.City,a.State,a.Zip,a.ZipStd,
        a.Gender,a.Education,a.Income_Level,a.Household,
             b.Car_Make,b.Car_Model,b.Car_Year
 from DEMOGRAPHIC_DATA_STD as a
 left join
      PURCHASE_DATA_STD    as b
 on    a.NameStdMatchCode=b.NameStdMatchCode and
       a.AddressStdMatchCode=b.AddressStdMatchCode and
       a.ZipStd=b.ZipStd;
quit;

/** Unload QKB for a specific locale from the memory**/

%DQUNLOAD;
```

ACKNOWLEDGMENTS

Thanks to our colleague at SAS Federal, Kerri Rivers, for her helpful feedback.

CONTACT INFORMATION

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

Kimberly Hare
SAS Federal, LLC
kim.hare@sas.com

Elena Shtern
SAS Federal, LLC
Elena.shtern@sas.com