Paper XXX-2025

Fuzzy Matching with SAS® PROC SQL

Mark Jordan, SAS Jedi (Retired)

ABSTRACT

We must often join data from disparate data sources where exact matching by primary key is not possible. In this paper, I'll discuss and demonstrate fuzzy matching techniques that can help get your data ready for analysis.

INTRODUCTION

In this paper I'll discuss:

- Standardizing values for cleaner matches
- Identifying exact matches
- Conducting fuzzy matching using traditional SAS functions, including:
  - SOUNDEX()
  - SPEDIS()
  - COMPLEV()
  - COMPGED()
- Generating and using Data Quality match codes in SQL

THE PROBLEM

As the owners of "Cats Adore" cat food, we've purchased a mailing list from "The Cat Gourmet" magazine with the goal of identifying potential new customers. We want to offer them a free sample in hopes they will buy our product. We have a nice, clean dataset containing our own customer names, addresses, and demographics such as birthdays, gender, etc. The mailing list from Cat Lovers Magazine is just the information used to generate mailing labels. The data looks like this:

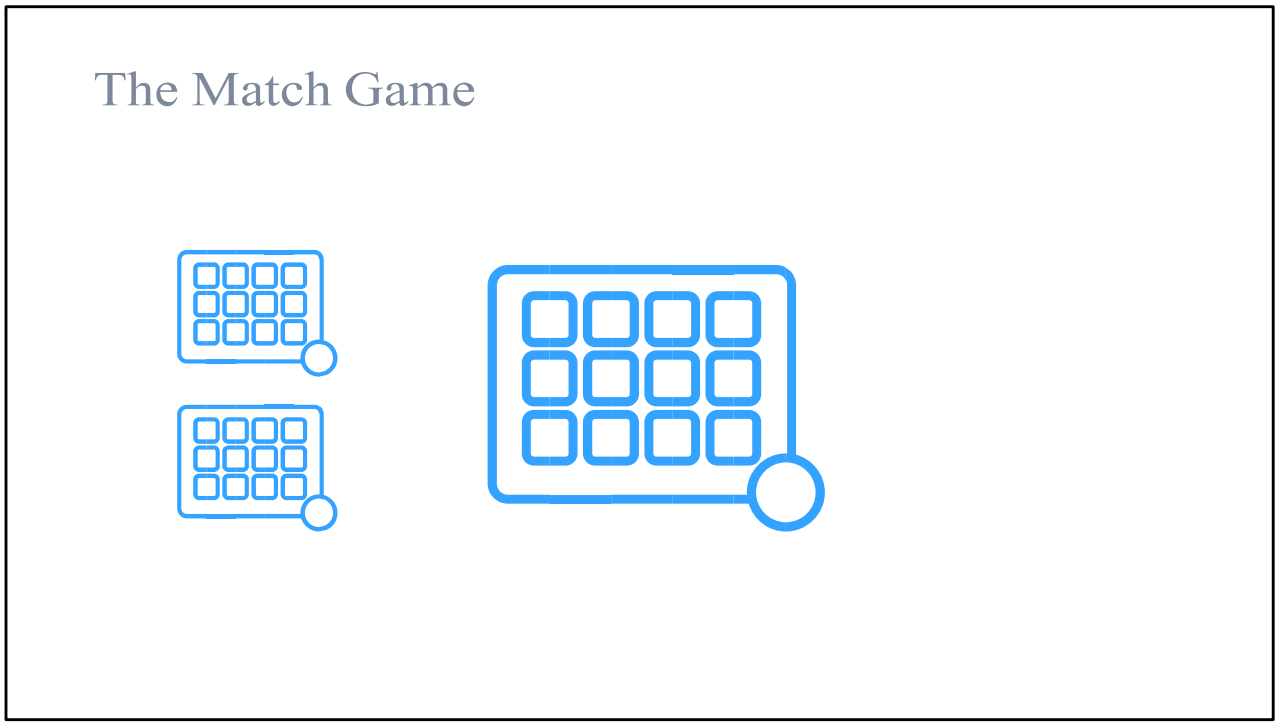
Our Customers

	Variable	Type	Length
1	ID	num	8
2	Family_Name	char	20
3	Given_Name	char	20
4	Birthdate	num	8
5	Gender	char	6
6	Address	char	50
7	City	char	30
8	State	char	2
9	Zip	char	10
10	Start_Date	num	8

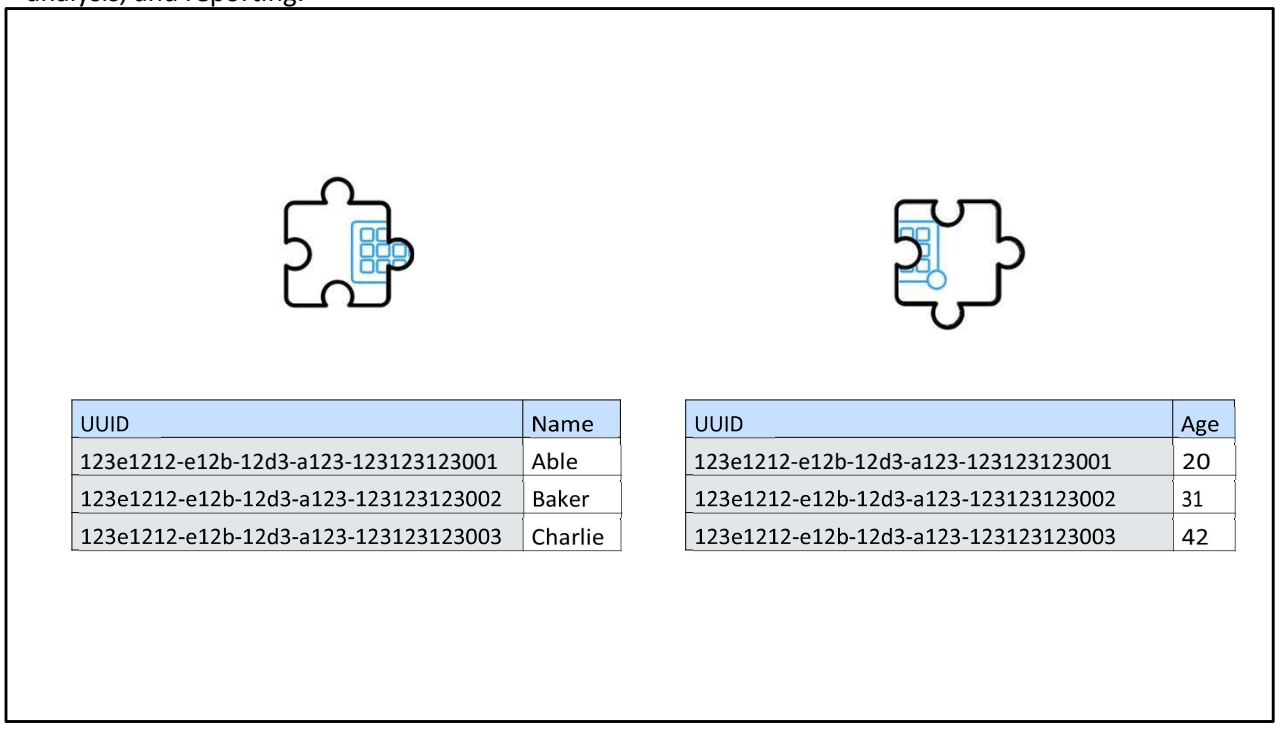
Cat Lovers Mailing List

	Variable	Type	Length
1	Name	char	50
2	Address1	char	100
3	City	char	50
4	State	char	20
5	Zip	num	8
6	SubscriptionID	num	8

MATCHING IN BASE SAS



We often need to assemble data from separate tables into a single table for further processing, analysis, and reporting.



UUID	Name
123e1212-e12b-12d3-a123-123123123001	Able
123e1212-e12b-12d3-a123-123123123002	Baker
123e1212-e12b-12d3-a123-123123123003	Charlie

UUID	Age
123e1212-e12b-12d3-a123-123123123001	20
123e1212-e12b-12d3-a123-123123123002	31
123e1212-e12b-12d3-a123-123123123003	42

Fuzzy Matching with SAS® PROC SQL

In a perfect world, the individual tables would contain a common unique identifier field, making for a simple and precise join. Of course, we don't live in a perfect world, so this scenario isn't as common as we'd like...

Name	Birthday	Birthday
Jacqueline Smith	1994-06-30	
John Smith	1994-06-30	
Joseph Jay	2000-01-01	

Date	Result
2023-02-21	Good
2023-02-22	Bad
2023-02-23	Avg

Name	BirthdayB hday
Jackie Smithqueline Smith	1994-06-30
John Smytheith	1994-06-30
Joseph J.ay	2000-01-01

PatientID	Age
0021345	25
0021354	52
0021368	66

More commonly, we will need to use the information in multiple columns to match records. If the data all comes from the same system, this often is also a simple join. But often, variations in how the data is entered can make this task trickier than it seems, requiring the use of fuzzy matching techniques to successfully join the records.

CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode
100000001	Katherin	Thomas	3005 Brookside Drive	Guin	AL	35563
100000002	Anthony	Feldmann	4026 Maple Lane	Huntsville	AL	35802
100000003	Sam	Tennyson	2715 Mulberry Avenue	Donaldson	AR	71941
100000004	Kristiana	Curry	399 Masonic Hill Road	Little Rock	AR	72212
100000016	Margaret	Robinson	622 Chestnut Street	Tampa	FL	33610

SubscriberID	Name	Address1	City	State	Zip
10010060	CHRISTY CURRY	399 MASONIC HILL RD	LITTLE ROCK	AR	72212-2122
10010127	JENNIFER RYAN	4974 WATER ST	FREMONT	CA	94539-9354
10010188	ANNA CHRISTIAN	3323 MORGAN ST	FT WALTON BEACH	FL	32548-8452
10010264	PEGGY ROBINSON	622 CHESTNUT ST	TAMPA	FL	33610-0163
10010305	MICHEAL TILLMAN	4854 POPLAR ST	CALUMET CITY	IL	60409-9040

For example, say we've determined that cat owners are a better credit risk than the overall population. We'd like to determine if any of our existing customers own cats. To help with this endeavor, we purchased the Cat Lovers Magazine mailing list. Now, while there is no ID column that provides a direct link between the Customers and CatLovers tables, we do have the individual's name and address information in both tables. This could be enough for matching. There are some obvious issues we'll have to deal with. For example:

- In the Customer table, data is entered in proper case, while the mailing list data is all upper case.
- In the customer table, ZipCode is a numeric column containing 5-digit zip codes, but in the CatLovers table, Zip is a character column containing ZIP+4 codes.
- Not every instance of a name is spelled the same, and some folks even use nicknames.

```
proc sql; create table xref as select CustomerID,
SubscriberID from catyums.customers as c inner
join catyums.catlovers as cat
        on scan(cat.Name,-1,' ')=upcase(LastName) and scan(cat.Name,1,' ')
            =upcase(FirstName) and cat.State=c.State and cat.City
            =upcase(c.City)
            and put(Zipcode,z5.)=scan(Zip,1,'-')
; quit;
```

```
NOTE: Table WORK.XREF created, with 2 rows and 2 columns
```

This join accounts for differences in text casing and zip code formats. The code runs without errors, but surely there must be more than 2 cat lovers in our customer database! This calls for a little fuzzy matching magic.

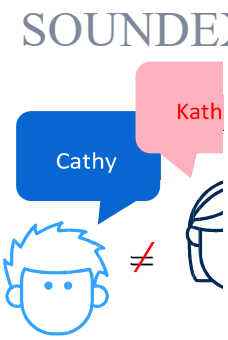
CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode
100000001	Katherin	Thomas	3005 Brookside e Driv	Guin	AL	35563
100000002	Anthony	Feldmann	4026 Maple Lane	Huntsville	AL	35802
100000003	Sam	Tennyson	2715 Mulberry Ave ue	Donaldson	AR	71941
100000004	Kristiana	Curry	399 Masonic Road Hill	Little Rock	AR	72212
100000016	Margaret	Robinson	622 Chestnut Street	Tampa	FL	33610

SubscriberID	Name	Address1	City	State	Zip
10010060	CHRISTY CURRY	399 MASONIC RD HILL	LITTLE ROCK	AR	72212-2122
10010127	JENNIFER RYAN	4974 WATER ST	FREMONT	CA	94539-9354
10010188	ANNA CHRISTIAN	3323 MORGAN ST	FT WALTON BEACH	FL	32548-8452
10010264	PEGGY ROBINSON	622 CHESTNUT ST	TAMPA	FL	33610-0163
10010305	MICHEAL TILLMAN	4854 POPLAR ST	CALUMET CITY	IL	60409-9040

Fuzzy Matching with SAS® PROC SQL

Fuzzy matching, sometimes referred to as fuzzy name matching or fuzzy string matching, uses techniques to identify text elements in a data set that are similar, but not necessarily identical. While both the SAS 9 Data Quality Server and SAS Viya provide the most powerful and configurable techniques for this type of work, there are several functions available in base SAS that can help tremendously.

SOUNDEX




```
proc sql;
select FirstName, LastName
                ,soundex(FirstName) from
        catyums.customers where CustomerID in
                (100000004,000100000008);
select Name
                ,soundex(scan(Name,1,'       '))       from
        catyums.catlovers where SubscriberID in
                (10010060,10010127);
quit;
```

FirstName	LastName	Score
Kristiana	Curry	K6235
Jennifer	Ryan	J516

Name	Score
CHRISTY CURRY	C623
JENNIFER RYAN	J516

Soundex is a phonetic algorithm that compares the sound of two text segments as they would be pronounced in English. It is often used for fuzzy matching based on names. Most SQL-based databases include a SOUNDEX function, and so does base SAS. SOUNDEX accepts a single argument consisting of a text string. In addition to the English bias, this algorithm is particularly sensitive to spelling diversions early in the text, so while it is very easy to use, it often misses matches that you or I would find exceptionally easy to detect. There is no way to control the scoring with SOUNDEX – so scores either match or don't.

```
proc sql; create table xref as select CustomerID,
SubscriberID from catyums.customers as c inner join
catyums.catlovers as cat
        on SOUNDEX(scan(cat.Name,-1,' '))=SOUNDEX(LastName) and
            SOUNDEX(scan(cat.Name,1,' '))=SOUNDEX(FirstName) and
            cat.State=c.State and cat.City =upcase(c.City)
            and put(Zipcode,z5.)=scan(Zip,1,'-')
; quit;
```

```
NOTE: Table WORK.XREF created, with 3 rows and 2 columns
```

Using the SOUNDEX function in our join criteria identifies an additional cat lover among our customers. But I'm sure that more cat lovers are lurking in there!

SPEDIS

SPEDIS

SPEDIS(compare word, base word)

```
proc sql;
select Name, FirstName, LastName
      ,sum(SPEDIS(scan(name, 1,' '),upcase(FirstName))
            ,SPEDIS(scan(name,-1,' '),upcase(LastName))) as Score
    from catyums.customers
        ,catyums.catlovers
    where calculated score <75
;
quit;
```

Name	FirstName	LastName	Score
JENNIFER RYAN	Jennifer	Ryan	0
ANNA CHRISTIAN	Anna	Christian	0
MICHEAL TILLMAN	Micheal	Tillman	0
JIM CARPENTER	James	Carpenter	66
MICHEAL KAUFMAN	Micheal	Tillman	71
MICHEAL KAUFMAN	Michael	Kaufman	7
MICKEY KAUFMAN	Michael	Kaufman	41

The asymmetric spelling distance algorithm is used to calculate the likelihood that two words match. SPEDIS is a base SAS function which makes that calculation using a sequence of operations to make the strings match, each with a specific base "cost". The total cost is divided by the length

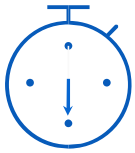
of the string to produce the final score. The resulting score falls between 0 and 200, with 0 indicating a perfect match. You choose the cutoff point to maximize true matches while minimizing false positives. We'll discuss two more fuzzy matching functions, COMPGED and COMPLEV. Of the three, SPEDIS is generally the slowest processing.

COMPGED

COMPGED

COMPGED(string-1, string-2 <,cutoff><,modifiers>)

```
proc sql;
select Name, FirstName, LastName
          ,sum(compged(scan(name, 1,' '),FirstName,'n')
               ,compged(scan(name,-1,' '),LastName,'n')) as Score
     from catyums.customers
          ,catyums.catlovers
     where calculated score <250
;
quit;
```



Name	FirstName	LastName	Score
JENNIFER RYAN	Jennifer	Ryan	0
ANNA CHRISTIAN	Anna	Christian	0
BOB KELLEY	Robert	Kelley	230
MICHEAL TILLMAN	Micheal	Tillman	0
JIM CARPENTER	James	Carpenter	120

The generalized edit distance function, or COMPGED, is a generalization of Levenshtein edit distance. The cost of each edit is additive, with a maximum cost of 200 and 0 indicating a perfect match. The COMPGED modifiers allow some control over processing and are applied in the order listed. Here the 'n' modifier removes quotation marks and ignores the case for both arguments. When choosing a cutoff point, the values returned by COMPGED are much larger than SPEDIS, making choosing a cutoff point to maximize true matches while minimizing false positives a bit easier.

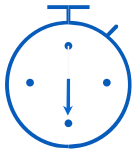
COMPGED processes faster than SPEDIS, but slower than COMPLEV.

COMPLEV

COMPLEV

COMPLEV(string-1, string-2 <,cutoff><,modifiers>)

```
proc sql;
select Name, FirstName, LastName
      ,sum(COMPLEV(scan(name, 1,' '),FirstName,'n')
          ,COMPLEV(scan(name,-1,' '),LastName,'n')) as Score
    from catyums.customers
        ,catyums.catlovers
    where calculated score <=3
;
quit;
```



Name	FirstName	LastName	Score
JENNIFER RYAN	Jennifer	Ryan	0
ANNA CHRISTIAN	Anna	Christian	0
MICHEAL TILLMAN	Micheal	Tillman	0
JIM CARPENTER	James	Carpenter	3
MICHEAL KAUFMAN	Michael	Kaufman	2
MICKEY KAUFMAN	Michael	Kaufman	3

The Levenshtein edit distance function, or COMPLEV, produces a score based on the number and type of edits required to make two strings match. Once again, a score of 0 indicates a perfect match. The COMPLEV modifiers the same as those for COMPGED, but the values returned are very small integers, making choosing a cutoff point to maximize true matches while minimizing false positives a little tricky. But on the plus side, COMPLEV is faster than both COMPGED and SPEDIS.

DEMO 1 - FUZZY MATCHING WITH BASE SAS

Using base SAS functionality with PROC SQL for fuzzy matching rows in tables without a common unique identifier using program files:

- **Fuzzy Matching - 1 - Why I need it.sas**
- **Fuzzy Matching - 2 - SOUNDEX.sas**
- **Fuzzy Matching - 3 - SPEDIS.sas**
- **Fuzzy Matching - 4 - COMPLEV.sas**
- **Fuzzy Matching - 5 - COMPGED.sas**

HOUSEHOLDING

Customers

Name	Street Address	City	State	Zip
Catherine M. Thomas	3005 Brookside Drive	Guin	AL	35563
János L. von Neumann	4026 Maple Lane	Huntsville	AL	35802
Sam C. Tennyson	2715 Mulberry Avenue	Donaldson	AR	71941
Kristi R. Curry	399 Masonic Hill Road	Little Rock	AR	72212
Wanda R. Smith	2190 Cedar Street	Pine Bluff	AR	71601

Cat Lover's Magazine Mailing List

Name	Address	City	State	Zip
BEAUTIFUL KITTY	3005 BROOKSIDE DR	GUIN	AL	35563
JOHN VON NEUMANN	4026 MAPLE LN	HUNTSVILLE	AL	35802
SAM CHARLES	3750 MULBERRY AVE	GROTON	CT	71941
CHRISTY CURRY	399 MASONIC HILL RD	LITTLE ROCK	AR	72212
TIGER MAN	2190 CEDAR ST	PINE BLUFF	AR	71601

Sometimes the problems are a lot tougher. This calls for either a ton of coding, or the use of some power tools. Here's an example of some of these problems. The computer wouldn't match any of the rows in these tables, yet as a human, I can clearly see several matches:

- It looks like our customer Catherine Thomas has subscribed her cat, Beautiful Kitty, to Cat Lover's magazine.
- Our customer Yanosh von Neumann is listed as the cat lover John von Neumann.
- Customer Kristi R. Curry and the subscriber Christy Curry are most probably the same person.
- And it looks like Wanda Smith has also subscribed her cat, Tiger Man, to the magazine.

We may have to ignore names altogether when matching the table in this example.

Fuzzy Matching with SAS® PROC SQL

Customers

Name	Street Address	City	State	Zip
Catherine M. Thomas	3005 Brookside Drive	Guin	AL	35563
János L. von Neumann	4026 Maple Lane	Huntsville	AL	35802
Sam C. Tennyson	2715 Mulberry Avenue	Donaldson	AR	71941
Kristi R. Curry	399 Masonic Hill Road	Little Rock	AR	72212
Wanda R. Smith	2190 Cedar Street	Pine Bluff	AR	71601

Cat Lover's Magazine Mailing List

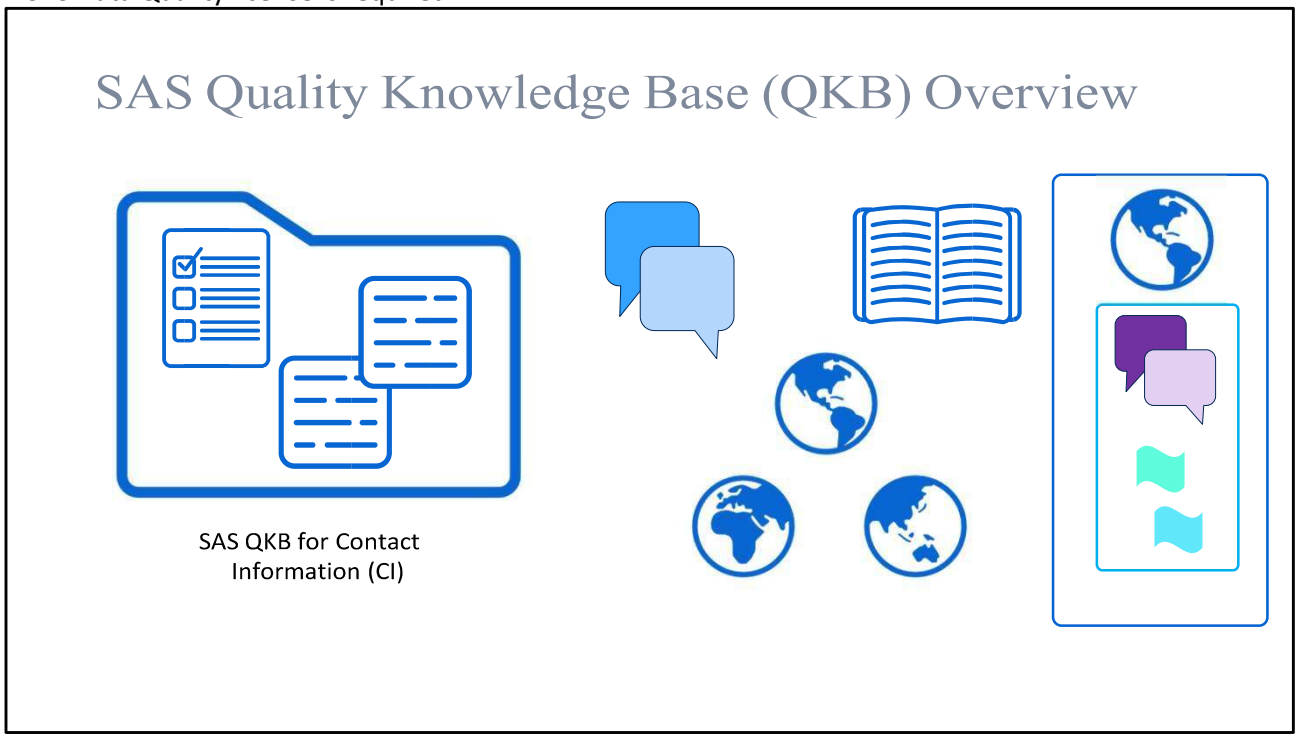
Name	Address	City	State	Zip
BEAUTIFUL KITTY	3005 BROOKSIDE DR	GUIN	AL	35563
JOHN VON NEUMANN	4026 MAPLE LN	HUNTSVILLE	AL	35802
SAM CHARLES	3750 MULBERRY AVE	GROTON	CT	71941
CHRISTY CURRY	399 MASONIC HILL RD	LITTLE ROCK	AR	72212
TIGER MAN	2190 CEDAR ST	PINE BLUFF	AR	71601

```
Address=tranwrd(Address,'ST','Street');
Address=tranwrd(Address,'RD','Road');
Address=tranwrd(Address,'AVE','Avenue'); …
```

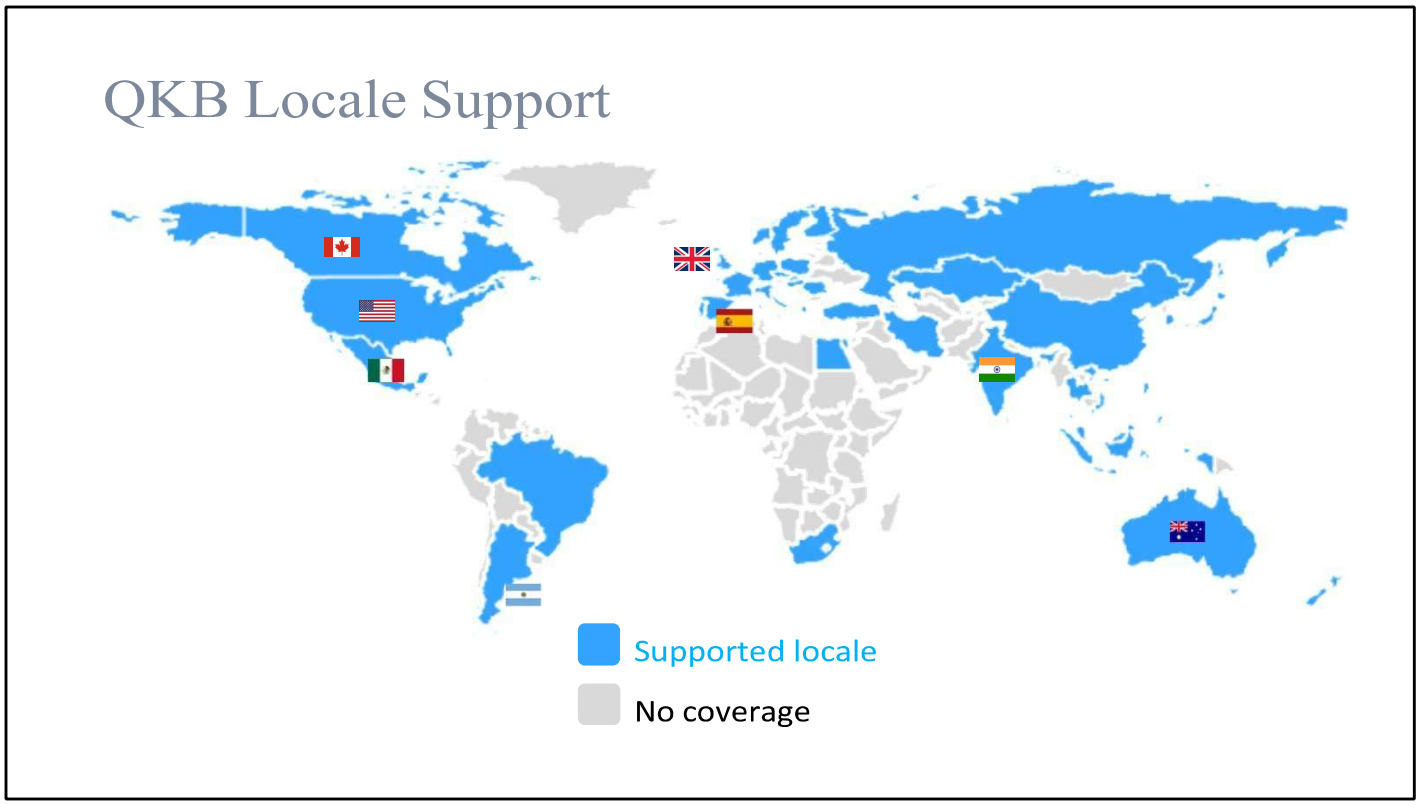
Instead, we could try to match households, using address, city, state, and zip code as the keys. But there are several problems with the text. The mailing list is all upper case and uses standard abbreviations like “ST” for “Street” and “AVE” for “Avenue”. <<**>>The customer data information is proper cased and spells out all of the words. This could be a coding nightmare…

SAS DATA QUALITY

SAS Data Quality provides SAS language elements that perform data quality operations such as matching and standardization to improve data accuracy and consistency. Higher quality data yields higher-value results. SAS Data Quality comes with SAS Viya, but if you are a SAS 9 user, a SAS Data Quality license is required.



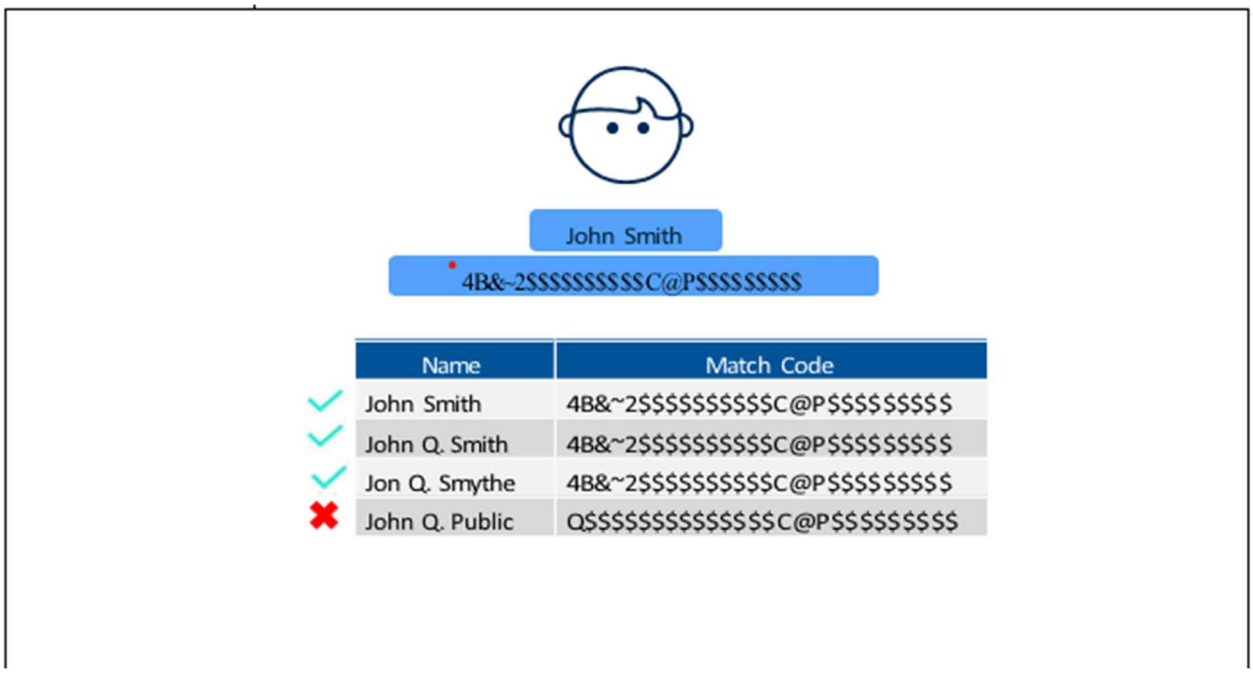
At the heart of SAS Data Quality is the SAS Quality Knowledge Base or QKB. A QKB is a collection of complex rules and reference data used for data standardization and cleansing. Your SAS Data Quality deployment includes a QKB called the SAS Quality Knowledge Base for Contact Information, supporting an array of SAS processes used to simplify and accelerate profiling, standardizing, modifying, and matching data. The QKB contains 10 definition types, inmore than 30 languages available in multiple localized variations. Information in a QKB is hierarchical in nature. The global, language, and locale levels each add more specific data and definitions in addition to those inherited from their parent level.



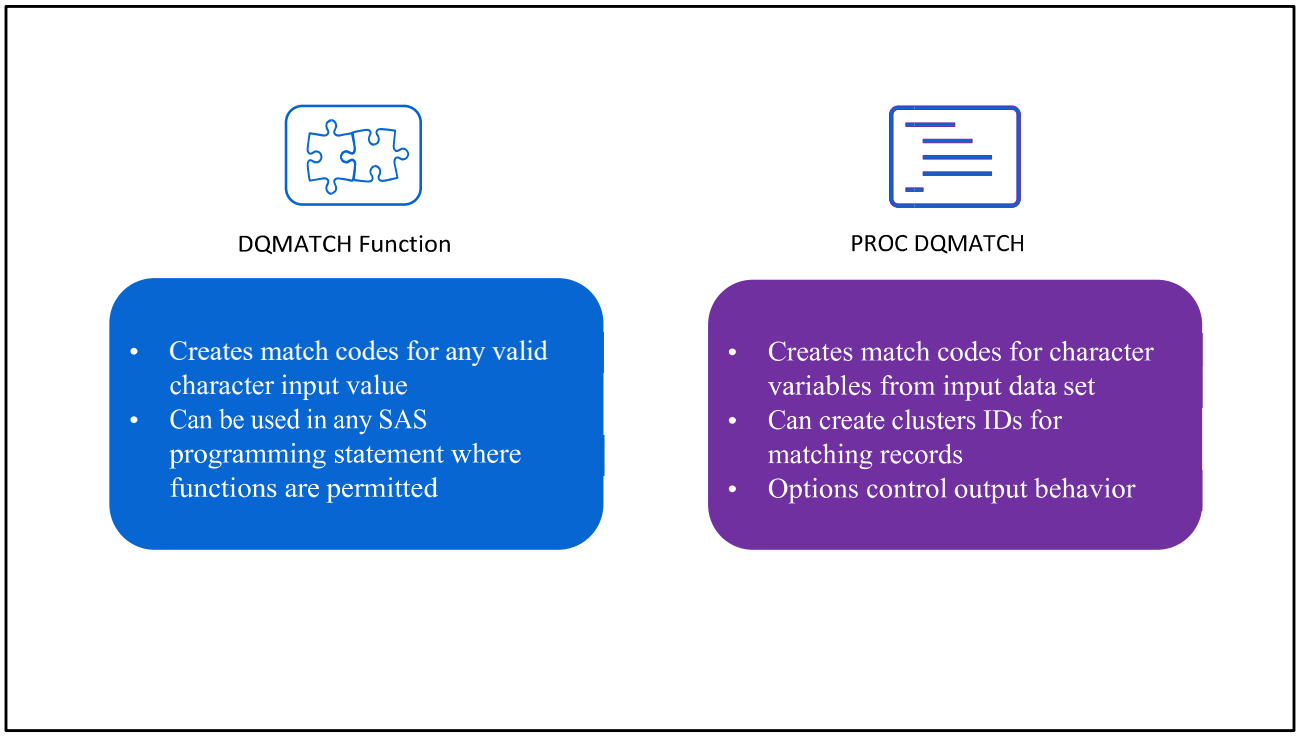
QKB definitions are applied as specified by a locale. A locale is a unique combination of geography and language. For example, English is spoken in many geographies, but there are considerable differences between them. The same is true of Spanish. It's important to know which locale applies to your data, because the locale affects the results produced by SAS Data Quality functions and procedures.

MATCH CODES

Consider this table. If we try to match John Smith to these rows using traditional criteria to join on Name, only the first row of data matches.



SAS Data Quality's "secret sauce" for fuzzy matching is the match code. Match codes are an encoded representation of a text string, computed based on a locale-sensitive combination of the data type, number of tokens, and sensitivity level. Using match codes instead of the raw text values, the computer can now see that these additional rows represent the same person, and still reliably decide that “John Q. Public” is not a match.



Fuzzy Matching with SAS® PROC SQL

SAS Data Quality provides both a DQMATCH function and a DQMATCH procedure. The DQMATCH function creates match codes for any valid text value. You can create a match code for a constant character value, the contents of a character variable, or the result of any valid character expression. You can use the DQMATCH function anywhere functions are permitted, which makes using match codes very flexible. I've used DQMATCH in the DATA step, PROC SQL, FEDSQL, and even DS2. Alternatively, you can use the DQMATCH procedure to create match codes for character variables in the input data set using multiple criteria, and cluster the data to help identify duplicate records. PROC DQMATCH includes several options for controlling how match codes and clusters are created, and for controlling output behavior.

DQMATCH()

DQMATCH Function Syntax and Example

Syntax:

```
DQMATCH(character-value, 'match-definition')<,sensitivity> <) ,'locale'>)
```

Example:

```
Matchcode=dqmatch(Contact, 'NAME');
```

Contact	Matchcode
Mark Abbott	&M&~$$$$$$$$$$BY3$$$$$$$$$
Mark Abbott	&M&~$$$$$$$$$$BY3$$$$$$$$$
Mark & Julie Abbott	&M&~$$$$$$$$$$BY3$$$$$$$$$

For fuzzy matching in PROC SQL, we'll focus on the DQMATCH function. The two required arguments are a character-value, which is the basis for the generated match code, and a match-definition, which specifies how the match code is produced. Two optional arguments provide finer control of match code generation. The default value for Sensitivity is 85, with valid values ranging from 50 to 95. The lower the sensitivity, the higher risk of false positives. The higher the sensitivity, the higher the risk of false negatives. In my experience, the default 85 works best for most applications. Locale specifies the locale to use when generating the match code. The default locale is used if no locale is specified. Both match-definition and locale values are text strings – hard-coded values must be written in quotes. In this example, the DQMATCH function generates match codes for the Contact variable using the NAME match definition. The output shows the match codes generated for each Contact value. In this example, the match codes indicate that all three rows of output refer to the same person.

DEMO 2 - FUZZY MATCHING WITH SAS DATA QUALITY

This demonstration illustrates using SAS Data Quality functions with PROC SQL for fuzzy matching rows in tables without a common unique identifier using program file **Fuzzy Matching - 6 - Complex Problems DQ Solution.sas**

RESOURCES

Download a ZIP file containing this PDF Power room as well as the code for the demos and data used to prepare this presentation at https://bit.ly/SASJediFuzzySQL

CONTACT INFORMATION

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

Mark Jordan, SAS Jedi (Retired)
Email: sas.jedi@gmail.com
X: @SASJedi