Fuzzy Matching with SAS: Data Analysts Tool to Cleaner Data

Josh Fogarasi

Agenda

- What is Fuzzy Matching Anyways? Why is it relevant to a Data Professional?
- Introducing some useful SAS Text Functions
- Fuzzy Matching Cycle
- Simple Business Case: Required to verify Customer Data and have mistyped information (Dirty Data)
- Example: Small Scale Data Merge
- Example: Bulk Processing-Overload

Things to Consider

- Computer Science": Fuzzy String Searching"
- Approximate join or a linkage between observations that is not an exact 100% one to one match
- Applies to strings/character arrays
- There is no one direct method or algorithm that solves the problem of joining mismatched data
- Fuzzy Matching is often an iterative process

Evaluating External Data

- Data Auditing: Access how clean your organizations quality level of the data that exists
- Marketing: Generate a Lead List from data from an external source with relevant contact information and exclude pre-existing customers
- Validating Data between two different databases (Access,Sybase,DB2,Excel Files,flat files)
- Correcting mistyped data fields among data sources

(Manually Maintained Spreadsheets)

Evaluating Internal Data

Data Stewardship-Maintain the Quality of Data for internal stakeholders

- Comparing Historical Data-Names, contact information, Addresses change over time
- Technology Migration between systems
- Different Data Sources that do not communicate with one another

Useful SAS Text Functions

Text Parsing Functions: Scan, Substring
Text Positioning-Index
String Modification:
Compress-removes all blanks, special characters
 Strip-removes leading/trailing blank spaces
Trim-removes trailing , used in concatenation functions
Case Manipulation-
Upcase()
Lowcase()
Propcase()

Useful SAS Text Functions

Text Extraction Functions: Scan() , Substring()
Index Function-gives position within a string for a specific text
String Modification:
Compress()- removes all blanks, special characters
Strip() -removes leading/trailing blank spaces
Trim() -removes trailing , used in concatenation functions
Case Manipulation-
Upcase()
Lowcase()
Propcase()
Other useful text Functions
Left(),right(),length()

Useful SAS Text Functions

Text Extraction Functions: Scan() , Substring()

Index Function-gives position within a string for a specific text

String Modification:

Compress()- removes all blanks, special characters

Strip() -removes leading/trailing blank spaces

Trim() -removes trailing , used in concatenation functions

Case Manipulation-

Upcase()

Lowcase()

Propcase()

Other useful text Functions

Left(),right(),length()

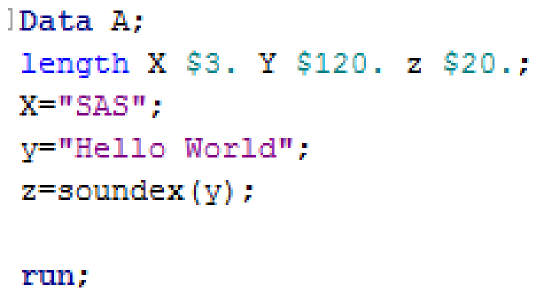
Functions useful for Fuzzy Matching

SOUNDEX

- generates a unique key/code for the string
- Phonetic coding system
- can be used with combination of the “*= “ sounds like operator for both Proc SQL or within the where statement of a data step
- Useful in simplifying long character strings and is computationally less expensive during the merge processing stage

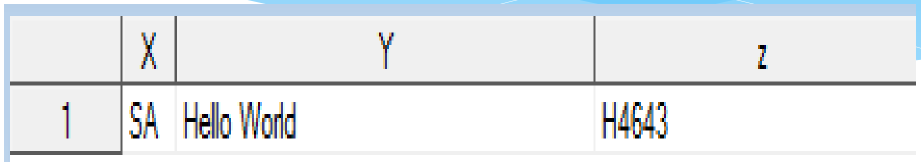
Functions useful for Fuzzy Matching

SOUNDEX-generates a unique key/code for the string

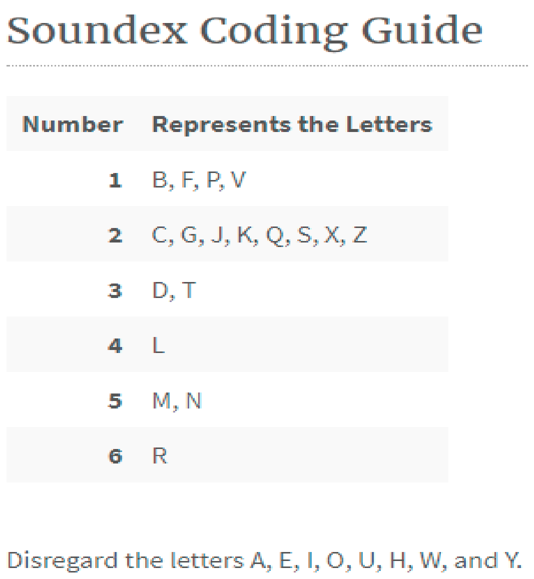


```
Data A;
length X $3. Y $120. z $20.;
X="SAS";
y="Hello World";
z=soundex(y);

run;
```



	X	Y	z
1	SA	Hello World	H4643



Soundex Coding Guide

Number	Represents the Letters
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Disregard the letters A, E, I, O, U, H, W, and Y.

Functions useful for Fuzzy Matching

COMPGED

- Computes the Levenshtein Edit Distance between two strings
- Scoring algorithm for (Replacement, deletion, or insertion) of characters within the string

COMPLEV

- Computes special case of the Levenshtein Distance
- Not as versatile as Compged, good for small strings

SPEDIS

- Measures the propensity of two strings matching

COMPARE

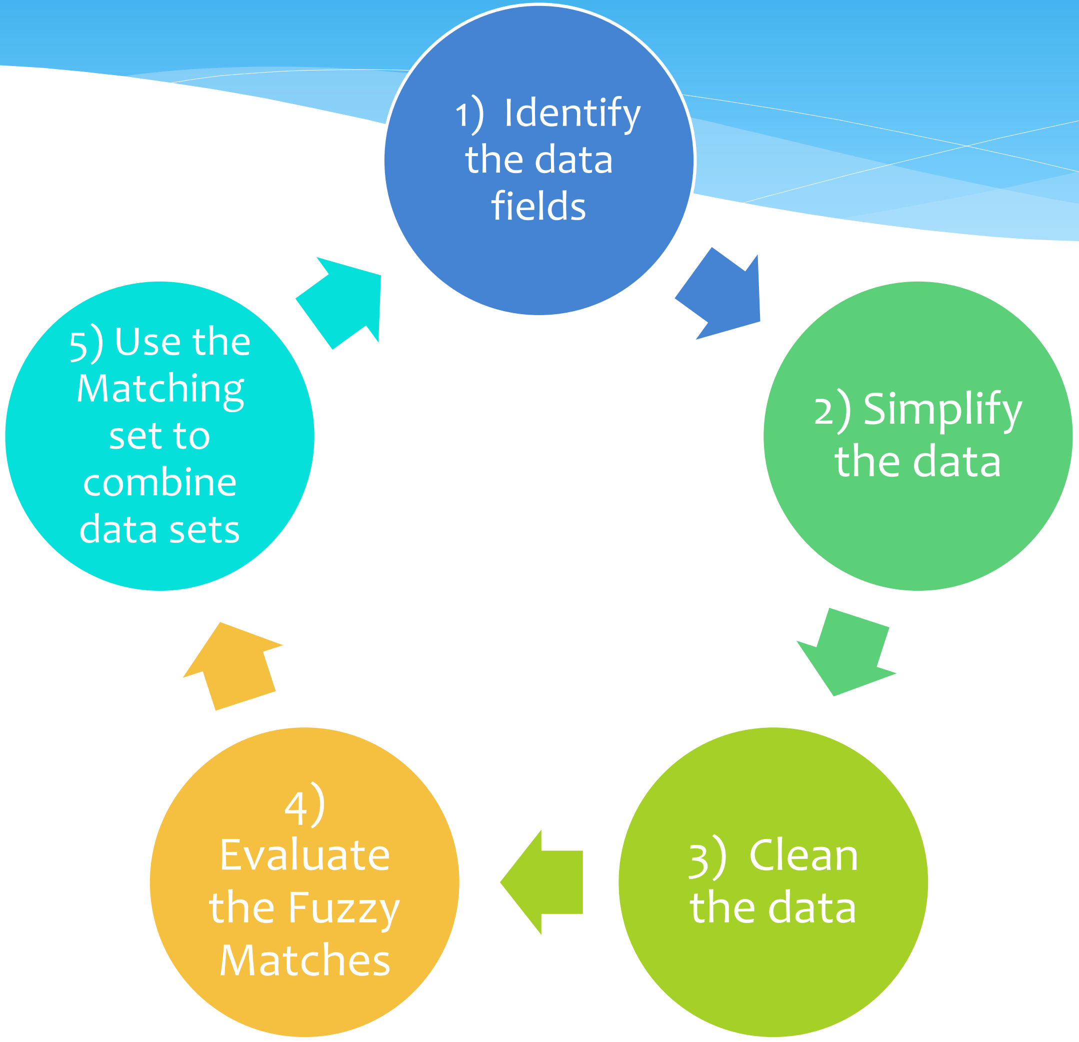
- Evaluates two strings and returns the left most character if they are different or a zero if they are the exact same

Functions useful for Fuzzy Matching

Levenshtein Edit Distance Algorithm

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Fuzzy Matching Cycle


1) Identify the data fields
2) Simplify the data
3) Clean the data
4) Evaluate the Fuzzy Matches
5) Use the Matching set to combine data sets

Simple Example: Identify the Data Fields

Problem: We have two customer lists with no unique key to match them on in order to combine the data sets

Data Set 1- Name, Mailing Address, Postal code, City
Data Set 2- Name and E-mail, Phone Number

Result- Data Set that contains All Parameters

Solution: Without a unique key such as a client_id or account_id then we are required to join these some how

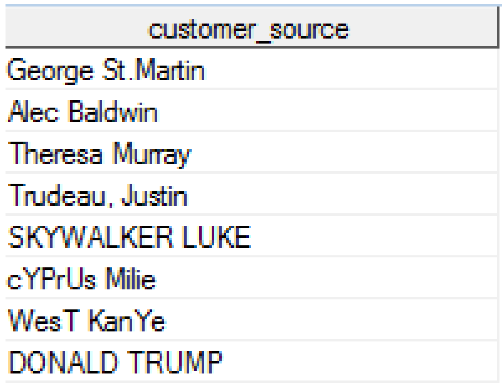
Twist-None of the Names were entered the same

Simple Example:

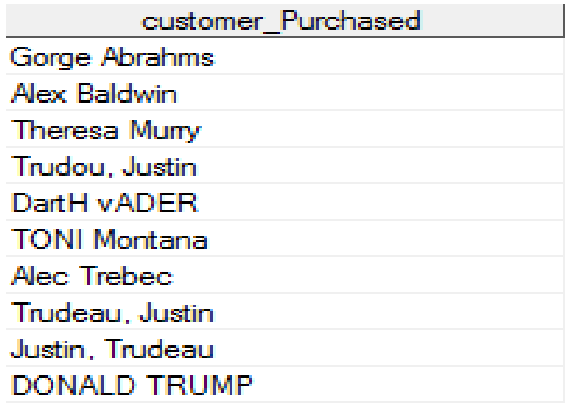
- Step 2- Simply the Data

-I prefer to use column vectors for Fuzzy Matching

-saves computational time and allows the process to run cleaner



customer_source
George St.Martin
Alec Baldwin
Theresa Murray
Trudeau, Justin
SKYWALKER LUKE
cYPrUs Milie
WesT KanYe
DONALD TRUMP

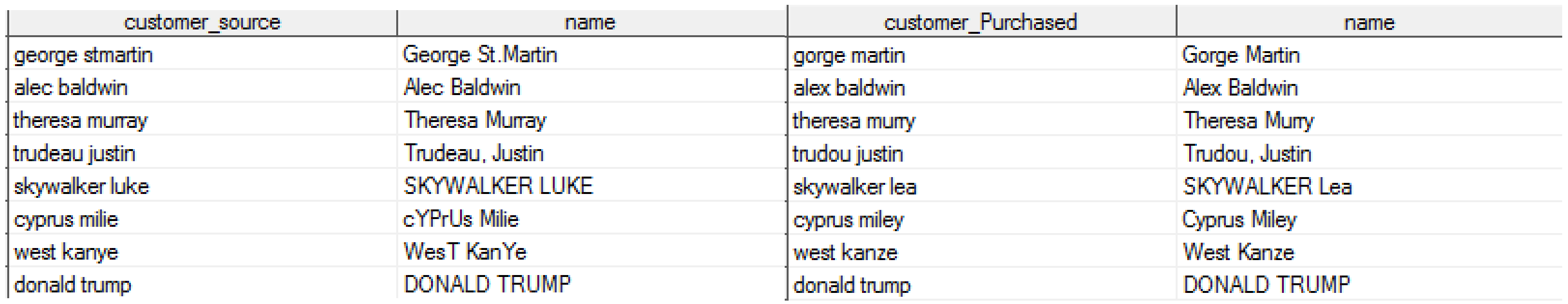


customer_Purchased
Gorge Abrahms
Alex Baldwin
Theresa Murry
Trudou, Justin
DartH vADER
TONI Montana
Alec Trebec
Trudeau, Justin
Justin, Trudeau
DONALD TRUMP

Simple Example:

- Step 3- Clean the Data

-Generally before any matching begins the fields should be made to resemble one another

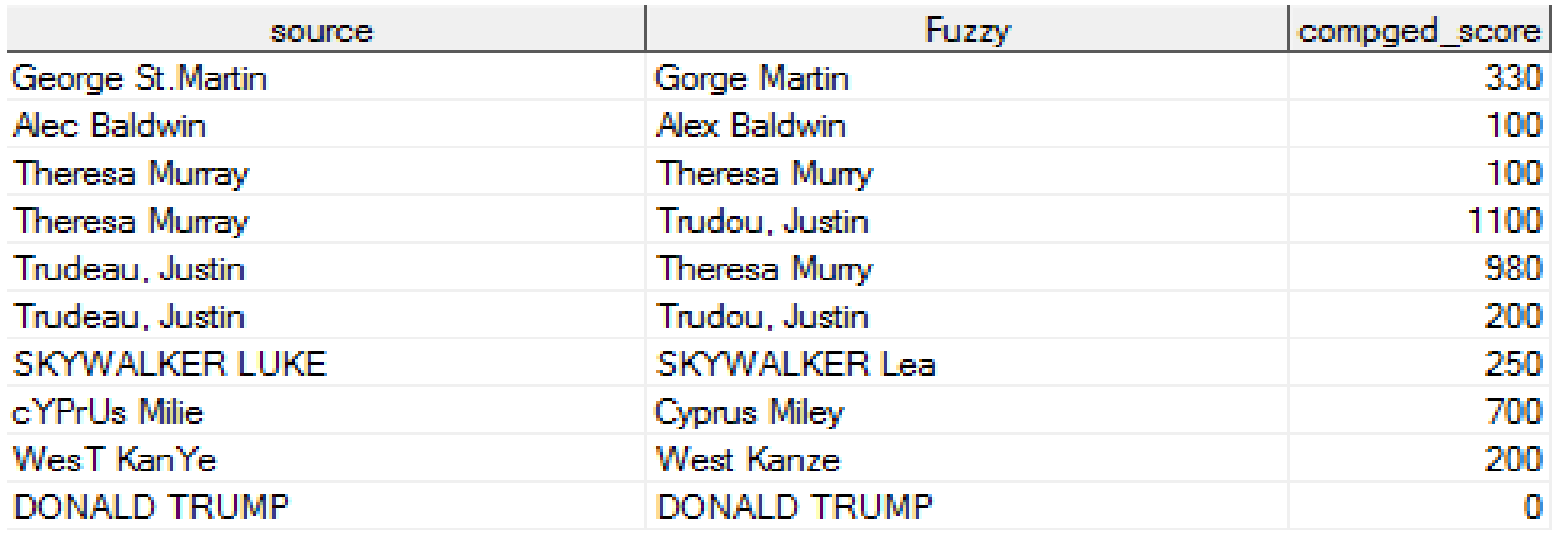


customer_source	name	customer_Purchased	name
george stmartin	George St.Martin	gorge martin	Gorge Martin
alec baldwin	Alec Baldwin	alex baldwin	Alex Baldwin
theresa murray	Theresa Murray	theresa murry	Theresa Murry
trudeau justin	Trudeau, Justin	trudou justin	Trudou, Justin
skywalker luke	SKYWALKER LUKE	skywalker lea	SKYWALKER Lea
cyprus milie	cYPrUs Milie	cyprus miley	Cyprus Miley
west kanye	WesT KanYe	west kanze	West Kanze
donald trump	DONALD TRUMP	donald trump	DONALD TRUMP

Simple Example:

- Step 4- Evaluate the Fuzzy Matches

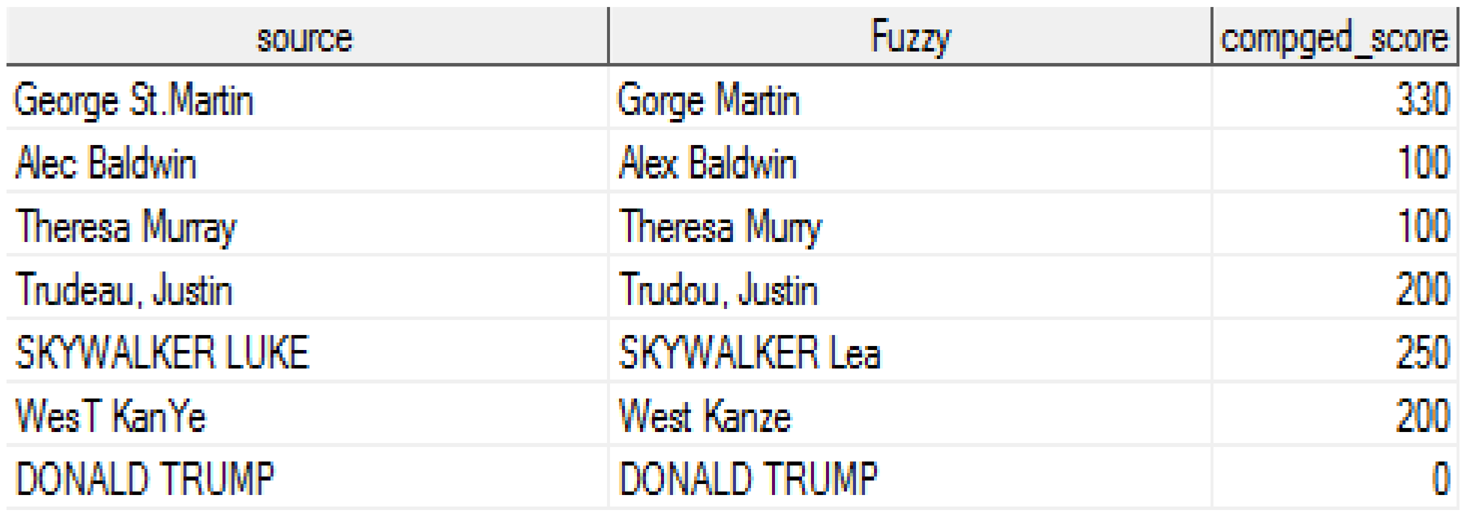
-Using the COMPGED after the Match is Complete



source	Fuzzy	compged_score
George St.Martin	Gorge Martin	330
Alec Baldwin	Alex Baldwin	100
Theresa Murray	Theresa Murry	100
Theresa Murray	Trudou, Justin	1100
Trudeau, Justin	Theresa Murry	980
Trudeau, Justin	Trudou, Justin	200
SKYWALKER LUKE	SKYWALKER Lea	250
cYPrUs Milie	Cyprus Miley	700
WesT KanYe	West Kanze	200
DONALD TRUMP	DONALD TRUMP	0

Simple Example:

- Step 5- Using the Matches to combine the two data sets



source	Fuzzy	compged_score
George St.Martin	Gorge Martin	330
Alec Baldwin	Alex Baldwin	100
Theresa Murray	Theresa Murry	100
Trudeau, Justin	Trudou, Justin	200
SKYWALKER LUKE	SKYWALKER Lea	250
WesT KanYe	West Kanze	200
DONALD TRUMP	DONALD TRUMP	0

Useful SAS Papers

- 1) **Matching Data Using Sounds-Like Operators and SAS® Compare Functions**
- Amanda Roesch, Educational Testing Service, Princeton, NJ
- 2)**Fuzzy Merges - A Guide to Joining Data sets with Non-Exact Keys Using the SAS SQL Procedure**
- Robert W. Graebner, Quintiles, Overland Park, KS, USA

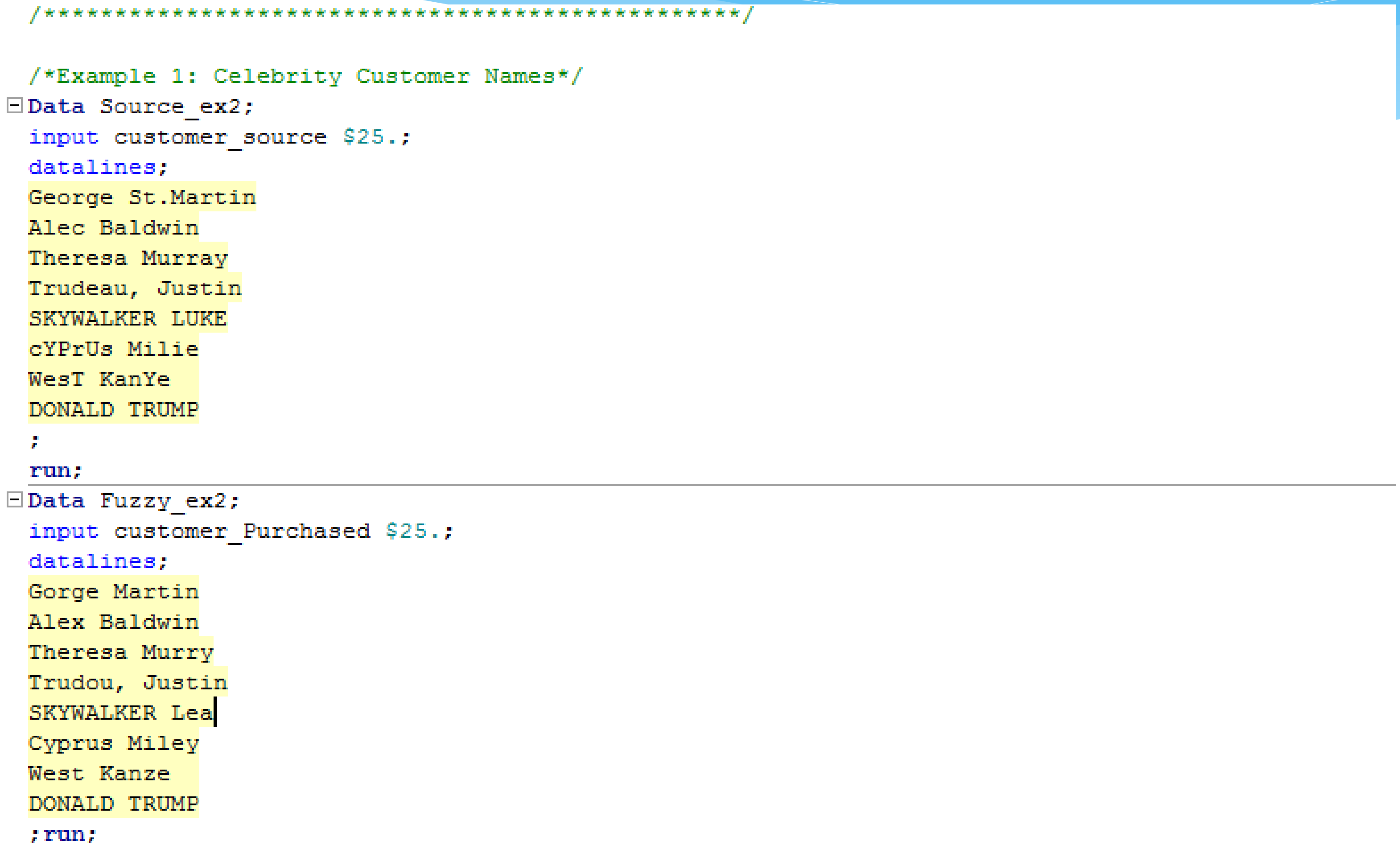
Websites-

a) http://blogs.sas.com/content/sgf/2015/01/27/how-to-perform-a-fuzzy-match-using-sas-functions/

b) www.lexjansen.com/nesug/nesug11/ps/ps07.pdf

c) www.lexjansen.com/nesug/nesug07/ap/ap23.pd

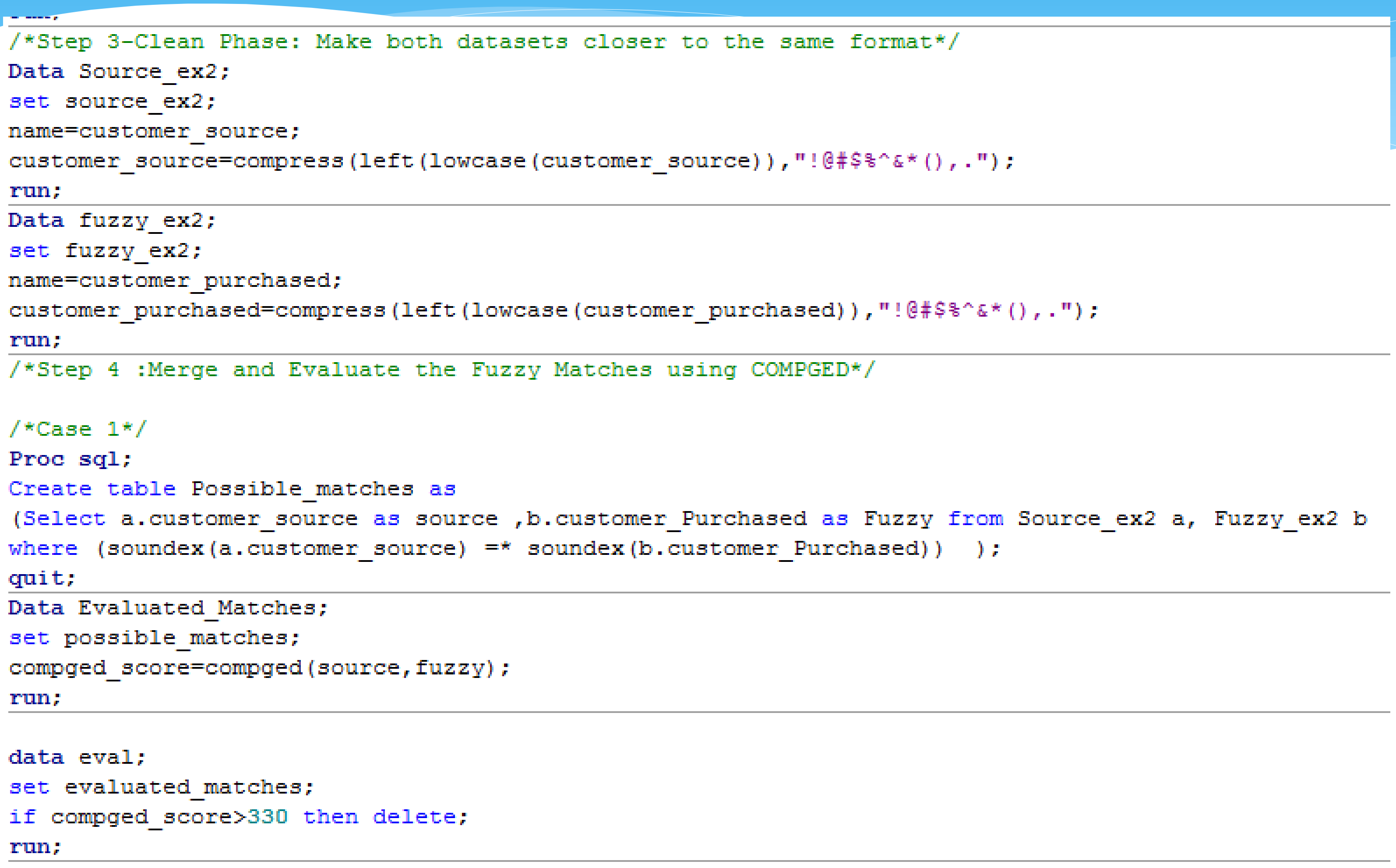
APPENDIX: CODE



```
/*************************************************/

/*Example 1: Celebrity Customer Names*/
Data Source_ex2;
input customer_source $25.;
datalines;
George St.Martin
Alec Baldwin
Theresa Murray
Trudeau, Justin
SKYWALKER LUKE
cYPrUs Milie
WesT KanYe
DONALD TRUMP
;
run;
Data Fuzzy_ex2;
input customer_Purchased $25.;
datalines;
Gorge Martin
Alex Baldwin
Theresa Murry
Trudou, Justin
SKYWALKER Lea
Cyprus Miley
West Kanze
DONALD TRUMP
;run;
```

APPENDIX



```
/*Step 3-Clean Phase: Make both datasets closer to the same format*/
Data Source_ex2;
set source_ex2;
name=customer_source;
customer_source=compress(left(lowcase(customer_source)),"!@#$%^&*(),.");
run;
Data fuzzy_ex2;
set fuzzy_ex2;
name=customer_purchased;
customer_purchased=compress(left(lowcase(customer_purchased)),"!@#$%^&*(),.");
run;
/*Step 4 :Merge and Evaluate the Fuzzy Matches using COMPGED*/

/*Case 1*/
Proc sql;
Create table Possible_matches as
(Select a.customer_source as source ,b.customer_Purchased as Fuzzy from Source_ex2 a, Fuzzy_ex2 b
where (soundex(a.customer_source) =* soundex(b.customer_Purchased))  );
quit;
Data Evaluated_Matches;
set possible_matches;
compged_score=compged(source,fuzzy);
run;

data eval;
set evaluated_matches;
if compged_score>330 then delete;
run;
```

Questions?