Paper SAS4157-2020

Python and the SAS® Quality Knowledge Base for Better Data Quality and Entity Resolution

Arnold Toporowski, SAS Institute (Canada) Inc

ABSTRACT

Python coders can now leverage the power of the SAS® Quality Knowledge Base and dramatically improve their data quality and data matching results. This session explores the capabilities now available to Python coders and gives coding examples and demonstrations showing how to leverage SAS Quality Knowledge Base capabilities such as parsing, standardization, and match-coding to better prepare data for analytics. Techniques for entity resolution and duplicate elimination are also explored.

INTRODUCTION

Data quality is a pervasive problem. There are some Python packages for data quality, but they are mostly about detecting or reporting on data quality, not for improving data quality. If you are using Python to improve data quality today, you are probably writing a lot of your own regular expressions and np.where() code.

For over twenty years SAS users have been able to use SAS® Data Quality and the SAS Quality Knowledge Base (QKB) to quickly and easily improve data quality, enrich data, and to facilitate duplicate elimination and entity resolution using fuzzy matching techniques.

Now, with SAS® Data Quality on SAS® Viya®, Python coders have access to the rules-based AI contained in the QKB and can leverage that same rich set of data quality capabilities.

Those capabilities include the following:

- Identification Analysis (to highlight data that is in the wrong place)
- Parsing (to take apart data into its constituent parts)
- Standardization (to correct inconsistent formatting)
- Gender Analysis (to get gender values from name information)
- Match Coding (to generate consistent codes for similar data values)
- And more! (Casing, Extraction, Pattern Analysis, Locale Guess, Language Guess)

The SAS QKB currently has support for 42 locales in 27 different languages, including languages using non-latin characters, such as Chinese, Japanese, Arabic, and Russian.

This paper will use a typical data quality problem and show you how the first five capabilities of the SAS QKB mentioned above can be leveraged from a Python/Jupyter notebook to transform data exhibiting a poor level of data quality into a higher level of data quality that you would require for downstream analytics.

EXAMPLE PROBLEM

The spreadsheet shown below contains the data that we will be using for our example.

	A	B	C	D	E	F	G	H	I
1	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email
2	201	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50	(819)-555-2334	
3	202	Tony Sarducci	2125 31 Ave	Calgary	Alberta	T2T 1T5	100		
4	203	Anthony Sarducci					400	tony.duke@telus.ca	
5	204	Amar Singh	5264 Joel Avenue	Burlington,ON	L7L3Y7		300		
6	205	Jack Plant					50	555-2334	JPlante@gmail.com
7	206	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa ON K1Z 1E9		613 755-2313	90		ArnoldT@sas.com
8	207	MJ Belanger	4500 Sherbrooke St W	Montreal	QC	H3Z 1E6	950	514 799 9239	
9	208	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary AB T2T1T5			10	403.265.5177	tony.sarducci@bell.ca
10	209	Plant, Jack	201-14 Denis Rd	Gatineau		J8V3J5	150	555-2334	
11	210	Mme Marie-Josée Bélanger	4500, rue Sherbrook Ouest	Montréal	PQ		900		MJBelanger@bell.ca
12	211	Ms. M.-J. Bélanger	4500 Sherbrooke O	Mon.	QC	H3Z 1E6	100	799-9239	
13	212	Jacques Plante	14, Chemin Denis, app 201	Cantley	Quebec		40		
14	213	Amar Singh	5264 Joel Av	Burlington	ON		100	(905) 637 5119	amar.singh@lost.com
15	214	Arnie Toperowski	38, privé Metropole, app 1605	Otawa	Ont	K1Z1E9	400		
16	215	Jacques Plante	14 Denise Unit 201	Cantley, QC, J8V 3J5			10		
17	216	JF Tremblay	P.O. Box 123	St-Marc-du-Lac-Long	QC		200	819-555-4545	
18	217	Jean-Francois Tremblay	CP 123	Saint Marc QC	G0L 1TO		90		JFTremblay@bell.ca
19	218	Tremblay, JF	CP 123	St-Marc	Quebec	GOL 1T0	50	555-4545	
20	219	A. Toporowski	38 Metropole Private, Unit 1605	Ottawa	Ontario	K1Z 1E9	100	7552313	ArnoldT@sas.com
21	220	Tony Sarducci	2125 31 Av	Calgary	AB		400		tony.duke@telus.ca

Figure 1. Spreadsheet Data That Exhibits Poor Data Quality

This data contains multiple rows per individual. Say for the type of analysis we want to do with this data we would like to have just six rows, one for each person, with a total Amount per person, and the other data cleansed and consolidated. Also, we would like to have a Gender field (Male/Female/Unknown) added.

Here are the data quality challenges that we are facing:

- no unique identifier per individual.
- various formats of names, including nick-names and initials.
- various formats of phone and address information.
- missing data and incomplete data (e.g., some phone numbers missing area codes).
- data in the wrong places (e.g., Email in the Phone column).
- concatenated data that needs parsing (e.g., "Calgary AB T2T1T5" all in the City field).
- typos ("Toporowski" vs "Toperowski", PostCode "G0L1TO" vs "GOL1T0", etc.).

The simplified business rules that we will use for this example are that we would like to get the longest Name, Address, City, Phone, and Email for each individual, along with total Amount, standardized Province, PostCode, Phone, and a generated gender code, like this:

	A	B	C	D	E	F	G	H	I
1	Name	gender	Address	City	Prov	PostCode	Amount	Phone	Email
2	Mr. Jacques Plante	M	14, Chemin Denis, app 201	Gatineau	QC	J8V 3J5	300	(819) 555 2334	JPlante@gmail.com
3	Mr. Anthony Sarducci	M	2125 31 Ave SW	Calgary	AB	T2T 1T5	910	(403) 265 5177	tony.sarducci@bell.ca
4	Amar Singh	M	5264 Joel Avenue	Burlington	ON	L7L 3Y7	400	(905) 637 5119	amar.singh@lost.com
5	Mr. Arnold Toporowski	M	38 Metropole Private, Unit 1605	Ottawa	ON	K1Z 1E9	590	(613) 755-2313	ArnoldT@sas.com
6	Mme Marie-Josée Bélanger	F	4500, rue Sherbrook Ouest	Montréal	QC	H3Z 1E6	1950	(514) 799 9239	MJBelanger@bell.ca
7	Jean-Francois Tremblay	M	P.O. Box 123	St-Marc-du-Lac-Long	QC	G0L 1T0	340	(819) 555 4545	JFTremblay@bell.ca

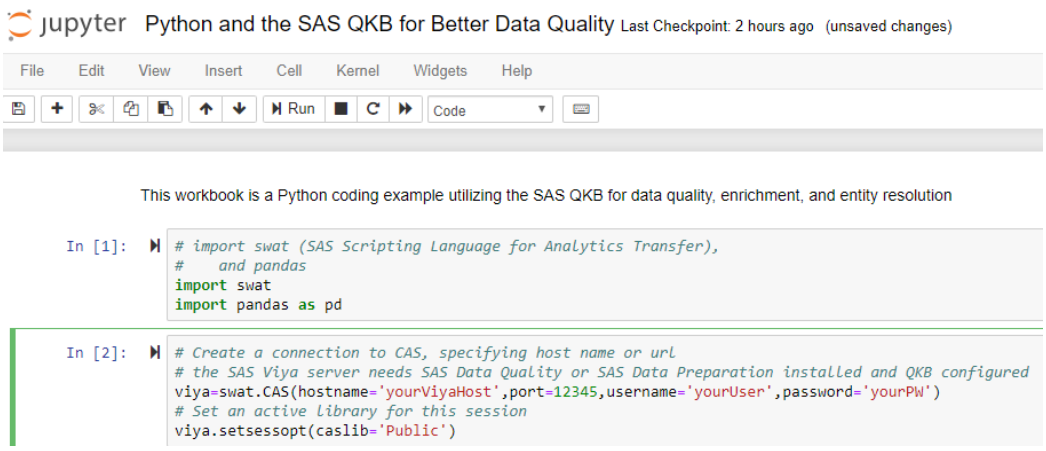
Figure 2. Results of the Data Quality Processing That We Would Like to Achieve

LOADING DATA INTO SAS® CLOUD ANALYTIC SERVICES

SAS Cloud Analytic Services (CAS) is the cloud-based server running on the SAS Viya high-performance, fault-tolerant analytics architecture. The smallest unit of work for the CAS server is a CAS action. CAS actions can load data, transform data, perform analytics, and create output.

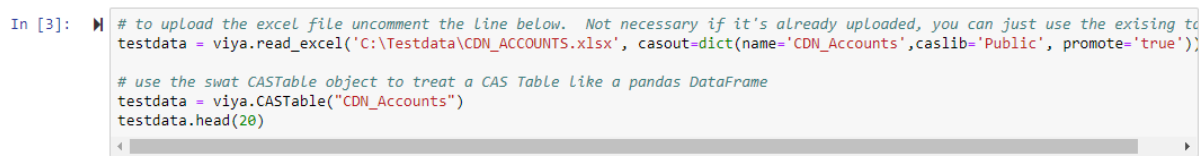
To use CAS in Python and Jupyter notebook you need to make sure that you have the SAS Scripting Wrapper for Analytics Transfer (SWAT) installed. See https://github.com/sassoftware/python-swat for details.

From a Jupyter notebook we can then import the SAS SWAT package (import swat) and connect to SAS CAS (swat.CAS), and then set the active library with the setsessopt action.

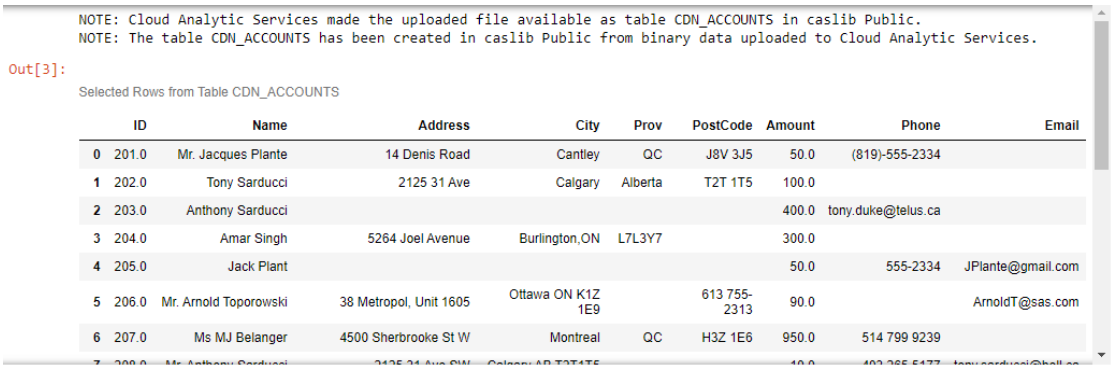


Display 1. Setting Up the Environment and Connecting to CAS

Then we need to upload the data, in this case we used the read_excel method. To display the uploaded data, we can use CASTable, a swat DataFrame-like object:



Display 2. Uploading Data and Then Fetching That Data Back for Display



```
NOTE: Cloud Analytic Services made the uploaded file available as table CDN_ACCOUNTS in caslib Public.
NOTE: The table CDN_ACCOUNTS has been created in caslib Public from binary data uploaded to Cloud Analytic Services.
```

Out[3]:

Selected Rows from Table CDN_ACCOUNTS

	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email
0	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819)-555-2334	
1	202.0	Tony Sarducci	2125 31 Ave	Calgary	Alberta	T2T 1T5	100.0		
2	203.0	Anthony Sarducci					400.0	tony.duke@telus.ca	
3	204.0	Amar Singh	5264 Joel Avenue	Burlington,ON	L7L3Y7		300.0		
4	205.0	Jack Plant					50.0	555-2334	JPlante@gmail.com
5	206.0	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa ON K1Z 1E9		613 755-2313	90.0		ArnoldT@sas.com
6	207.0	Ms MJ Belanger	4500 Sherbrooke St W	Montreal	QC	H3Z 1E6	950.0	514 799 9239	

Display 3. Output From the Display Data Action

DATA QUALITY AND DATA ENRICHMENT OPERATIONS

IDENTIFICATION ANALYSIS

The first data quality operation we will invoke is the dqIdentify function, through the dataStep.runCode CAS action. This inspects the City, Prov, PostCode, and Phone using the "Field Content" definition, in the English Canadian locale (ENCAN) of the QKB.

The results that come back show where these fields are empty, or where we are getting data different than we expected, using the newly created _Ident fields:

- The Phone_Ident field shows where Phone is empty or looks like an email.
- The Post_Ident field shows where PostCode is empty or looks like a phone number.
- The Prov_Ident field show where the Prov Field is empty or contains Postal Code.
- The City_Ident field shows where the City fields contains more than just City.

In [4]:

```
# run dataStep code invoking dq function to Identify contents of City, Province, PostCode fields
viya.dataStep.runCode(
    code=''' data public.testexcel_dq_from_Python ;
             set public.CDN_Accounts ;
                 City_Ident = dqIdentify(City,'Field Content','ENCAN');
                 Prov_Ident = dqIdentify(Prov,'Field Content','ENCAN');
                 Post_Ident = dqIdentify(PostCode,'Field Content','ENCAN');
                 Phone_Ident = dqIdentify(Phone,'Field Content','ENCAN');
             run;''')
# let's look at just the first six rows of our data ...
dq = viya.CASTable('testexcel_dq_from_Python')
dq.head(6)
```

Out[4]:

le TESTEXCEL_DQ_FROM_PYTHON

Address	City	Prov	PostCode	Amount	Phone	Email	City_Ident	Prov_Ident	Post_Ident	Phone_Ident
14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819)-555-2334		CITY	STATE/PROVINCE	POSTAL CODE	PHONE
2125 31 Ave	Calgary	Alberta	T2T 1T5	100.0			CITY	STATE/PROVINCE	POSTAL CODE	EMPTY
				400.0	tony.duke@telus.ca		EMPTY	EMPTY	EMPTY	E-MAIL
5264 Joel Avenue	Burlington,ON	L7L3Y7		300.0			CITY-STATE/PROVINCE-POSTAL CODE	POSTAL CODE	EMPTY	EMPTY
				50.0	555-2334	JPlante@gmail.com	EMPTY	EMPTY	EMPTY	PHONE
38 Metropol, Unit 1605	Ottawa ON K1Z 1E9		613 755-2313	90.0		ArnoldT@sas.com	CITY-STATE/PROVINCE-POSTAL CODE	EMPTY	PHONE	EMPTY

Display 4. Commands and Output from the Identification Analysis

For a real-world problem, you might want to invoke the dqIdentify function on more fields, or even all fields, depending on the extent of the problems. In this case, we are only working on these four fields since we know from previous data profiling work on the spreadsheet that the problem of misplaced data is limited to these four fields.

RIGHT-FIELDING

Next, we want to use the intelligence gained with the use of the dqIdentify function within SAS DATA step code that is invoked with the dataStep.runCode CAS action.

In the simple example code below, we find those cases where Email, Phone and Postal Code are in the wrong spot, and move them to the correct spot (and update the _Ident indicator fields as well).

```
In [5]:  # using the above results of the SAS QKB Identity Analysis,
         # run dataStep code to move stray phone, email, Postal code info to their correct places
         viya.dataStep.runCode(
             code=''' data public.testexcel_dq_from_Python;
                      set public.testexcel_dq_from_Python;
                          if Phone_Ident = 'E-MAIL' then do;
                                Email = Phone;
                                Phone = '';
                                Phone_Ident = 'EMPTY';
                             end;
                          if Post_Ident = 'PHONE' then do;
                                Phone = PostCode;
                                PostCode = '';
                                Phone_Ident = 'PHONE';
                                Post_Ident = 'EMPTY';
                             end;
                          if Prov_Ident = 'POSTAL CODE' then do;
                                PostCode = Prov;
                                Prov     = '';
                                Post_Ident = 'POSTAL CODE';
                                Prov_Ident = 'EMPTY';
                             end;
                       run;''')
         dq.head(6)
```

	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email	City_Ident	Prov_Ident	Post_Ident	Phone_Ident
0	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819)-555-2334		CITY	STATE/PROVINCE	POSTAL CODE	PHONE
1	202.0	Tony Sarducci	2125 31 Ave	Calgary	Alberta	T2T 1T5	100.0			CITY	STATE/PROVINCE	POSTAL CODE	EMPTY
2	203.0	Anthony Sarducci					400.0		tony.duke@telus.ca	EMPTY	EMPTY	EMPTY	EMPTY
3	204.0	Amar Singh	5264 Joel Avenue	Burlington,ON		L7L3Y7	300.0			CITY-STATE/PROVINCE-POSTAL CODE	EMPTY	POSTAL CODE	EMPTY
4	205.0	Jack Plant					50.0	555-2334	JPlante@gmail.com	EMPTY	EMPTY	EMPTY	PHONE
5	206.0	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa ON K1Z 1E9			90.0	613 755-2313	ArnoldT@sas.com	CITY-STATE/PROVINCE-POSTAL CODE	EMPTY	EMPTY	PHONE

Display 5. The Code Used for Right-Fielding and the Results

PARSING

Next, we will use the dqParse and dqParseTokenGet functions to fix those rows where City, Province, and Postal Code have been concatenated together.

In [6]:

```
# run dataStep code to combine City, Province, PostCode fields for problem rows and parse out correct info
# using the SAS QKB parse definiton for City-State/Province-Postal Code.
viya.dataStep.runCode(
    code=''' data public.testexcel_dq_2(drop=City_Ident Prov_Ident Post_Ident Phone_Ident parsedCPP);
             set public.testexcel_dq_from_Python;
             if City_Ident ^= 'CITY' and (Prov_Ident='EMPTY' or Post_Ident='EMPTY') then do;
                 parsedCPP = dqParse(CATX(' ',City,Prov,PostCode), 'City - State/Province - Postal Code', 'ENCAN');
                 City     = dqParseTokenGet(parsedCPP, 'City', 'City - State/Province - Postal Code', 'ENCAN');
                 Prov     = dqParseTokenGet(parsedCPP, 'State/Province', 'City - State/Province - Postal Code', 'ENCAN');
                 PostCode = dqParseTokenGet(parsedCPP, 'Postal Code', 'City - State/Province - Postal Code', 'ENCAN');
                end;
             run;''')
dq2 = viya.CASTable('testexcel_dq_2')
dq2.to_frame()
```

Out[6]:

Selected Rows from Table TESTEXCEL_DQ_2

	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email
0	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819)-555-2334	
1	202.0	Tony Sarducci	2125 31 Ave	Calgary	Alberta	T2T 1T5	100.0		
2	203.0	Anthony Sarducci					400.0		tony.duke@telus.ca
3	204.0	Amar Singh	5264 Joel Avenue	Burlington	ON	L7L3Y7	300.0		
4	205.0	Jack Plant					50.0	555-2334	JPlante@gmail.com
5	206.0	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa	ON	K1Z 1E9	90.0	613 755-2313	ArnoldT@sas.com
6	207.0	Ms MJ Belanger	4500 Sherbrooke St W	Montreal	QC	H3Z 1E6	950.0	514 799 9239	
7	208.0	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary	AB	T2T1T5	10.0	403.265.5177	tony.sarducci@bell.ca
8	209.0	Plant, Jack	201-14 Denis Rd	Gatineau		J8V3J5	150.0	555-2334	
9	210.0	Mme Marie-Josée Bélanger	4500, rue Sherbrook Ouest	Montréal	PQ		900.0		MJBelanger@bell.ca
10	211.0	Ms. M.-J. Bélanger	4500 Sherbrooke O	Mon.	QC	H3Z 1E6	100.0	799-9239	
11	212.0	Jacques Plante	14, Chemin Denis, app 201	Cantley	Quebec		40.0		
12	213.0	Amar Singh	5264 Joel Av	Burlington	ON		100.0	(905) 637 5119	amar.singh@lost.com
13	214.0	Arnie Toperowski	38, privé Metropole, app 1605	Otawa	Ont	K1Z1E9	400.0		
14	215.0	Jacques Plante	14 Denise Unit 201	Cantley	QC	J8V 3J5	10.0		
15	216.0	JF Tremblay	P.O. Box 123	St-Marc-du-Lac-Long	QC		200.0	819-555-4545	
16	217.0	Jean-Francois Tremblay	CP 123	Saint Marc	QC	G0L 1TO	90.0		JFTremblay@bell.ca
17	218.0	Tremblay, JF	CP 123	St-Marc	Quebec	GOL 1T0	50.0	555-4545	
18	219.0	A. Toporowski	38 Metropole Private, Unit 1605	Ottawa	Ontario	K1Z 1E9	100.0	7552313	ArnoldT@sas.com
19	220.0	Tony Sarducci	2125 31 Av	Calgary	AB		400.0		tony.duke@telus.ca

Display 6. The Code Used for Parsing Apart the City, Province, and Postal Code, and the Results

Now that we have all information in the correct place, we can move on to standardizations, corrections, and enrichment.

STANDARDIZATION AND ENRICHMENT

We want to standardize the Prov, PostCode, and Phone fields using the dqStandardize function. You want to use the correct standardization definition on each data type. (for example, the "State/Province (Postal Standard)" definition on the Prov field). We also specify the locale as "ENCAN", so that the data gets standardized to "English, Canadian" standards (note that the ENCAN and FRCAN definitions handle data in both French and English). If we had some USA data, we would want to use the "ENUSA" definition on those rows.

Here we also invoke the dqGender function to generate the gender field. It will get a value of M, F, or U, depending on what it finds in the Name field. It considers name prefixes (Mr., Ms, Mme, etc.) and uses a lookup table of given names that skew toward a specific gender.

In [7]:

```
# use the SAS QKB Standardize defintions for Province, PostCode, and Phone to standardize those columns in place
# use the SAS QKB Gender Analysis definition to enrich the data with a gender field, based on the Name
viya.dataStep.runCode(
    code=''' data public.testexcel_dq_2;
             set public.testexcel_dq_2;
             Prov    = dqStandardize(Prov,'State/Province (Postal Standard)','ENCAN');
             Phone   = dqStandardize(Phone,'Phone');
             PostCode= dqStandardize(PostCode,'Postal Code','ENCAN');
             gender  = dqGender(Name,'Name','ENCAN') ;
            run;''')
dq2.to_frame()
```

Out[7]:

Selected Rows from Table TESTEXCEL_DQ_2

	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email	gender
0	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819) 555 2334		M
1	202.0	Tony Sarducci	2125 31 Ave	Calgary	AB	T2T 1T5	100.0			M
2	203.0	Anthony Sarducci					400.0		tony.duke@telus.ca	M
3	204.0	Amar Singh	5264 Joel Avenue	Burlington	ON	L7L 3Y7	300.0			M
4	205.0	Jack Plant					50.0	555 2334	JPlante@gmail.com	M
5	206.0	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa	ON	K1Z 1E9	90.0	(613) 755 2313	ArnoldT@sas.com	M
6	207.0	Ms MJ Belanger	4500 Sherbrooke St W	Montreal	QC	H3Z 1E6	950.0	(514) 799 9239		F
7	208.0	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary	AB	T2T 1T5	10.0	(403) 265 5177	tony.sarducci@bell.ca	M
8	209.0	Plant, Jack	201-14 Denis Rd	Gatineau		J8V 3J5	150.0	555 2334		M
9	210.0	Mme Marie-Josée Bélanger	4500, rue Sherbrook Ouest	Montréal	QC		900.0		MJBelanger@bell.ca	F
10	211.0	Ms. M.-J. Bélanger	4500 Sherbrooke O	Mon.	QC	H3Z 1E6	100.0	799 9239		F
11	212.0	Jacques Plante	14, Chemin Denis, app 201	Cantley	QC		40.0			M
12	213.0	Amar Singh	5264 Joel Av	Burlington	ON		100.0	(905) 637 5119	amar.singh@lost.com	M
13	214.0	Arnie Toperowski	38, privé Metropole, app 1605	Otawa	ON	K1Z 1E9	400.0			M
14	215.0	Jacques Plante	14 Denise Unit 201	Cantley	QC	J8V 3J5	10.0			M
15	216.0	JF Tremblay	P.O. Box 123	St-Marc-du-Lac-Long	QC		200.0	(819) 555 4545		U
16	217.0	Jean-Francois Tremblay	CP 123	Saint Marc	QC	G0L 1T0	90.0		JFTremblay@bell.ca	M
17	218.0	Tremblay, JF	CP 123	St-Marc	QC	G0L 1T0	50.0	555 4545		U
18	219.0	A. Toporowski	38 Metropole Private, Unit 1605	Ottawa	ON	K1Z 1E9	100.0	755 2313	ArnoldT@sas.com	U
19	220.0	Tony Sarducci	2125 31 Av	Calgary	AB		400.0		tony.duke@telus.ca	M

Display 7. Standardization of Prov, Phone, and PostCode, and Generating Gender

ENTITY RESOLUTION OPERATIONS

MATCH-CODING

Next is the dqMatch function. Sensitivity is the third parameter, which determines how much "fuzziness" is allowed in the matchcode. The standard setting for sensitivity is 85, but can be set higher to require closer matches, or set lower to permit "fuzzier" matches. Both Canadian locales (ENCAN and FRCAN) will handle bilingual English and French data and generate the same match code for similar data no matter the language. If we had some USA data, we would probably want to use the "ENUSA" definition on those rows.

In [8]:

```
# using the SAS QKB matchcode definitions create matchcodes on Name, Address, City.
viya.dataStep.runCode(
    code=''' data public.testexcel_dq_3;
             set public.testexcel_dq_2;
             Name_matchcode55 = dqMatch(Name,'Name',55,'ENCAN');
             Address_matchcode70 = dqMatch(Address,'Address',70,'ENCAN');
             City_matchcode85 = dqMatch(City,'City',85,'ENCAN');
             if length(Phone) > 7 then Phone7 = substr(Phone,length(Phone)-7,8);
            run;''')
# let's look at results for rows 7 to 10 ...
dq3 = viya.CASTable('testexcel_dq_3')
dq3[['ID','Name','Address','City','Name_matchcode55','Address_matchcode70','City_matchcode85','Phone7']].fetch(from_=7,to=10)
```

Out[8]: **§ Fetch**

Selected Rows from Table TESTEXCEL_DQ_3

	ID	Name	Address	City	Name_matchcode55	Address_matchcode70	City_matchcode85	Phone7
6	207.0	Ms MJ Belanger	4500 Sherbrooke St W	Montreal	MWBF $B $	S5042YMY3 $...	BP~YW $$$	799 9239
7	208.0	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary	4Y8J $ $	HZH9KZ ...	3WFY	265 5177
8	209.0	Plant, Jack	201-14 Denis Rd	Gatineau	NWB~ $C $	Z8P $$$...	F~P $	555 2334
9	210.0	Mme Marie-Josée Bélanger	4500, rue Sherbrook Ouest	Montréal	MWBF $B $	S5042YMY3 $...	BP~YW $$$	

Display 8. Similar Data Values Get the Same Matchcode Using the dqMatch Function

CLUSTERING

Next we need to load the Entity Resolution CAS action set, and use the entityres.match CAS action to cluster together records based on the matching rules that we specify. In this example, we are using the matchcodes to match on the following:

- Name & Street Address & PostalCode - OR -
- Name & City & Province - OR -
- Name & Phone (last 7 digits) - OR -
- Name & Email

In [9]:
```
#Load Entity Resolution CAS action set
viya.loadactionset(actionset="entityRes")
```

```
NOTE: Added action set 'entityRes'.
```

Out[9]:

§ actionset

entityRes

elapsed 0.00169s · sys 0.00165s · mem 0.2MB

In [10]:
```
# use the entityres.match CAS action to match rows on:
#  (Name & Address & Postal Code)  OR  (Name & City & Province)  OR  (Name & Phone)  OR  (Name & Email)
#
dq_clustered = viya.CASTable('test_Clustered', groupBy='CLUSTERID', replace=True)
viya.entityres.match(clusterid='CLUSTERID',
        intable=dq3,
        matchrules=[{'rule':[{'columns':['Name_matchcode55','Address_matchcode70','Postcode']},]},
                    {'rule':[{'columns':['Name_matchcode55','City_matchcode85','Prov']}]},
                    {'rule':[{'columns':['Name_matchcode55','Phone7']}]},
                    {'rule':[{'columns':['Name_matchcode55','Email']}]}
                   ],
        outtable=dq_clustered)

# display cluster results.  CLUSTERID is a 24-byte character string
dq_clustered[['CLUSTERID','ID','Name','Address','City','Phone','Email']].sort_values('CLUSTERID').to_frame()
```

Out[10]:

Selected Rows from Table TEST_CLUSTERED

	CLUSTERID	ID	Name	Address	City	Phone	Email
0	AAAAAAAAAAAAAAAAAAAAAA==	205.0	Jack Plant			555 2334	JPlante@gmail.com
1	AAAAAAAAAAAAAAAAAAAAAA==	215.0	Jacques Plante	14 Denise Unit 201	Cantley		
2	AAAAAAAAAAAAAAAAAAAAAA==	212.0	Jacques Plante	14, Chemin Denis, app 201	Cantley		
3	AAAAAAAAAAAAAAAAAAAAAA==	209.0	Plant, Jack	201-14 Denis Rd	Gatineau	555 2334	
4	AAAAAAAAAAAAAAAAAAAAAA==	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	(819) 555 2334	
5	AAAAAAAAAAABAAAAAAAAAA==	203.0	Anthony Sarducci				tony.duke@telus.ca
6	AAAAAAAAAAABAAAAAAAAAA==	208.0	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary	(403) 265 5177	tony.sarducci@bell.ca
7	AAAAAAAAAAABAAAAAAAAAA==	220.0	Tony Sarducci	2125 31 Av	Calgary		tony.duke@telus.ca
8	AAAAAAAAAAABAAAAAAAAAA==	202.0	Tony Sarducci	2125 31 Ave	Calgary		
9	AAAAAAAAAAADAAAAAAAAAA==	213.0	Amar Singh	5264 Joel Av	Burlington	(905) 637 5119	amar.singh@lost.com
10	AAAAAAAAAAADAAAAAAAAAA==	204.0	Amar Singh	5264 Joel Avenue	Burlington		

Display 9. Clustering Rules Using Matchcodes, and the Resulting CLUSTERID

The result of the entityres.match CAS action is a new column, which is called CLUSTERID here. All rows that fall into the same cluster, according to our matching rules, will get the same CLUSTERID value. This column is a 24-byte character string. Minor differences in the CLUSTERID values are a little difficult for most humans to detect. Therefore, you might want to transform the CLUSTERID into a numeric value.

Here we use the simple.groupByInfo CAS action to turn the CLUSTERID 24-byte character string into a numeric value called _GroupID_. Now all rows in the same cluster also have the same numeric _GroupID_ value.

In [11]:

```
# to make the clusters easier to see, use simple.groupByInfo CAS action to genereate numeric GroupID from alphanumeric CLUSTE

dq_clust_nums = viya.CASTable('test_Clust_nums',replace=True)
dq_clustered.simple.groupByInfo(generatedColumns='GROUPID',
                                casout=dq_clust_nums,
                                includeDuplicates=True)
dq_clust_nums[['_GroupID_','ID','Name','Address','City','Prov','PostCode','Amount','Phone','Email']].sort_values('_GroupID_')
```

Out[11]:

Selected Rows from Table TEST_CLUST_NUMS

	GroupID	ID	Name	Address	City	Prov	PostCode	Amount	Phone	Email
0	1.0	205.0	Jack Plant					50.0	555 2334	JPlante@gmail.com
1	1.0	215.0	Jacques Plante	14 Denise Unit 201	Cantley	QC	J8V 3J5	10.0		
2	1.0	212.0	Jacques Plante	14, Chemin Denis, app 201	Cantley	QC		40.0		
3	1.0	209.0	Plant, Jack	201-14 Denis Rd	Gatineau		J8V 3J5	150.0	555 2334	
4	1.0	201.0	Mr. Jacques Plante	14 Denis Road	Cantley	QC	J8V 3J5	50.0	(819) 555 2334	
5	2.0	203.0	Anthony Sarducci					400.0		tony.duke@telus.ca
6	2.0	208.0	Mr. Anthony Sarducci	2125 31 Ave SW	Calgary	AB	T2T 1T5	10.0	(403) 265 5177	tony.sarducci@bell.ca
7	2.0	220.0	Tony Sarducci	2125 31 Av	Calgary	AB		400.0		tony.duke@telus.ca
8	2.0	202.0	Tony Sarducci	2125 31 Ave	Calgary	AB	T2T 1T5	100.0		
9	3.0	213.0	Amar Singh	5264 Joel Av	Burlington	ON		100.0	(905) 637 5119	amar.singh@lost.com
10	3.0	204.0	Amar Singh	5264 Joel Avenue	Burlington	ON	L7L 3Y7	300.0		
11	4.0	214.0	Arnie Toperowski	38, privé Metropole, app 1605	Otawa	ON	K1Z 1E9	400.0		
12	4.0	206.0	Mr. Arnold Toporowski	38 Metropol, Unit 1605	Ottawa	ON	K1Z 1E9	90.0	(613) 755 2313	ArnoldT@sas.com
13	4.0	219.0	A. Toporowski	38 Metropole Private, Unit 1605	Ottawa	ON	K1Z 1E9	100.0	755 2313	ArnoldT@sas.com
14	5.0	207.0	Ms MJ Belanger	4500 Sherbrooke St W	Montreal	QC	H3Z 1E6	950.0	(514) 799 9239	
15	5.0	211.0	Ms. M.-J. Bélanger	4500 Sherbrooke O	Mon.	QC	H3Z 1E6	100.0	799 9239	
16	5.0	210.0	Mme Marie-Josée Bélanger	4500, rue Sherbrook Ouest	Montréal	QC		900.0		MJBelanger@bell.ca
17	6.0	218.0	Tremblay, JF	CP 123	St-Marc	QC	G0L 1T0	50.0	555 4545	
18	6.0	217.0	Jean-Francois Tremblay	CP 123	Saint Marc	QC	G0L 1T0	90.0		JFTremblay@bell.ca
19	6.0	216.0	JF Tremblay	P.O. Box 123	St-Marc-du-Lac-Long	QC		200.0	(819) 555 4545	

Display 10. Transforming the 24-byte CLUSTERID Into a Numeric _GroupID_

You might have noticed that the QKB was not involved in this clustering step. So you could, if you wanted to, download the data from the server and do the clustering on your client, using Python code instead of using the entityres.match CAS action on the server. The advantages of doing it on the server are better performance and reduced network traffic. This doesn't really matter when dealing with just 20 records, but it certainly would matter if we were dealing with 20 million records.

Now that we have all the records for our six people grouped together as we had hoped we would, all that's left to do is sum the amounts for each person, and choose the best information that we have for each person on each of the other columns. This is called Survivorship.

SURVIVORSHIP

Below is some example SAS DATA step code to create a surviving record for each person. In this example, we are simply summing up the Amount for each cluster, and selecting the Name, Address, City, Postal, Phone, and Email with the longest string lengths in the cluster. Real world survivorship rules are usually more robust than this, so please don't consider this code to be a "best-practices" survivorship coding example.

Again, this step could be done with Python code on the client side. The advantages of doing it on the server are better performance and reduced network traffic. After this step is done, you only need to download one record per individual to your client. Or leave the data on the server and leverage SAS Analytics in CAS from Python as well!

In [12]:

```
# run datastep code to create golden record for each cluster with best info, and total amount
viya.dataStep.runCode(
    code=''' data public.test_done
                    (drop= max_Name min_gender max_Address max_City max_Postal tot_Amount max_Phone max_Email
                           CLUSTERID ID Name_matchcode55 Address_matchcode70 City_matchcode85 Phone7);
             set public.test_Clust_Nums;
             by _GroupID_;

             retain   max_Name  min_gender  max_Address  max_City  max_Postal  tot_Amount  max_Phone  max_Email;
             if first._GroupID_ then do;
                      max_Name = Name;
                      min_gender = gender;
                      max_Address = Address;
                      max_City = City;
                      max_Postal = PostCode;
                      tot_Amount = Amount;
                      max_Phone = Phone;
                      max_Email = Email;
                      end;
                 else do;
                       if length(Name) < length(max_Name)       then Name = max_Name;       else max_Name = Name;
                       if gender > min_gender                   then gender = min_gender;   else min_gender = gender;
                       if length(Address) < length(max_Address) then Address = max_Address; else max_Address = Address;
                       if length(City) < length(max_City)       then City = max_City;       else max_City = City;
                       if length(PostCode) < length(max_Postal) then PostCode = max_Postal; else max_Postal = PostCode;
                       tot_Amount = Amount + tot_Amount;
                       Amount = tot_Amount;
                       if length(Phone) < length(max_Phone)     then Phone = max_Phone;     else max_Phone = Phone;
                       if length(Email) < length(max_Email)     then Email = max_Email;     else max_Email = Email;
                       end;
             if last._GroupID_ then output;
    run;''')
dq_done = viya.CASTable('test_done')
dq_done[['_GroupID_','Name','gender','Address','City','Prov','PostCode','Amount','Phone','Email']].sort_values('_GROUPID_').h
```

Out[12]:

Selected Rows from Table TEST_DONE

	GroupID	Name	gender	Address	City	Prov	PostCode	Amount	Phone	Email
0	1.0	Mr. Jacques Plante	M	14, Chemin Denis, app 201	Gatineau	QC	J8V 3J5	300.0	(819) 555 2334	JPlante@gmail.com
1	2.0	Mr. Anthony Sarducci	M	2125 31 Ave SW	Calgary	AB	T2T 1T5	910.0	(403) 265 5177	tony.sarducci@bell.ca
2	3.0	Amar Singh	M	5264 Joel Avenue	Burlington	ON	L7L 3Y7	400.0	(905) 637 5119	amar.singh@lost.com
3	4.0	Mr. Arnold Toporowski	M	38 Metropole Private, Unit 1605	Ottawa	ON	K1Z 1E9	590.0	(613) 755 2313	ArnoldT@sas.com
4	5.0	Mme Marie-Josée Bélanger	F	4500, rue Sherbrook Ouest	Montréal	QC	H3Z 1E6	1950.0	(514) 799 9239	MJBelanger@bell.ca
5	6.0	Jean-Francois Tremblay	M	P.O. Box 123	St-Marc-du-Lac-Long	QC	G0L 1T0	340.0	(819) 555 4545	JFTremblay@bell.ca

Display 11. Surviving Just One Record Per Individual

CONCLUSION

This paper has shown that the rules-based AI capabilities in the SAS QKB are a powerful way for you, as a Python coder, to achieve better data quality more quickly and easily than you could by trying to write your own code to achieve the same results.

What's next? I recommend you try out these capabilities on a data quality problem of your own, or take the example shown here and take it further. What if you wanted to standardize the formatting of the Names and remove name prefixes, or add missing name prefixes? (hint: there are Name Standardization and Name Parsing definitions in the QKB).

Finally, you might be starting to think about how to move beyond just an interactive data science session in Jupyter and thinking about how to operationalize this data quality process. Will you want to schedule this as a regular batch job? Or turn it into a web service, callable in real-time by other applications? Or embed it in a process ingesting streaming data? All these deployment options for data quality are available with SAS.

ACKNOWLEDGMENTS

Thanks go out to Kevin D. Smith for his guidance on coding style, and to colleagues Ron Yee, André Lafreniere, and Guy Bourassa for content suggestions for this paper.

RECOMMENDED READING

- *SAS® Blogs: "Using Python to work with SAS Viya and CAS", by Chris Hemedinger*
- *SAS® Viya®: The Python Perspective, by Kevin D. Smith and Xiangxiang Meng*
- *SAS® Data Quality: Getting Started*
- *SAS® Quality Knowledge Base for Contact Information: Online Help*
- *SAS® Data Quality: Language Reference* (the "Functions supported in CAS" section)
- *SAS® Viya®: System Programming Guide* (the Python syntax examples)
- *SAS® Data Quality: CAS Action Programming Guide*
- *SAS® Cloud Analytics Services: CASL Programmer's Guide*

CONTACT INFORMATION

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

Arnold Toporowski
SAS Institute (Canada) Inc
+1 (613) 755-2313
Arnold.Toporowski@sas.com