Paper 3209-2015

Standardizing the Standardization Process

Frank Ferriola and Avery Long, Financial Risk Group

ABSTRACT

A prevalent problem surrounding ETL development is the ability to apply consistent logic/manipulation of source data when migrating to target data structures. Certain inconsistencies that add a layer of complexity include, but are not limited to, naming conventions and data types associated to multiple sources, numerous solutions applied by an array of developers, and multiple points of updating. In this paper, we examine the evolution of implementing a best practices solution during the process of data delivery, with respect to standardizing data. The solution begins with injecting the transformations of the data directly into the code at the standardized layer via Base SAS® or SAS® Enterprise Guide™. A more robust method that we will explore is to apply these transformations with SAS® Macro Facility™. This provides the capability to apply these changes in a consistent manner across multiple sources. We further explore this solution by implementing the macros within SAS® Data Integration Studio™ processes on the Data Management Platform. We consider these issues within the financial industry, but the proposed solution can be applied across multiple industries.

INTRODUCTION

As data sourcing expands exponentially over the years, the issue of data quality and standardization becomes more and more important. Unfortunately, in most organizations there is limited time to gather the data and prepare the data for analytics so less and less attention is paid to the data quality issues in the data. Fortunately for us, SAS provides many ways to streamline this process no matter which tools the developer needs to use. From Base SAS® to SAS® Enterprise Guide™ to SAS® Data Integration Studio and Data Management Tools, there are many ways to improve the process of data cleansing and Standardization, no matter the size of the organization, the number and experience of the developers, or the size of the project and data.

We will present some of the problems the developer can encounter and explore ways to make the process more efficient. Although we cannot account for every situation that you will encounter, hopefully the ideas presented here can steer you to resolving other issues as you encounter them.

WHY SHOULD WE STANDARDIZE?

When we develop an ETL process it is important to have certain rules to apply across your team. Even if you are the only developer of the process it is important to have certain rules to apply in your code. The primary reason being that it makes it easier for someone else to take over when you or the team move on to bigger and better things.

With many people working on a development project, you can have many styles of coding throughout the development cycle using different procedures, functions or other methods in implementing the solution. By standardizing the process, you still allow for some variations in the process, but you will have certain commonalities across the development team.

It's also important to standardize in order to avoid problems that may occur because of limitations in you servers such as space or processing limitations. Understanding these limitations as a team, helps you to define adequate measures of standardization and helps to avoid larger problems later and possibly having to redesign large portions of code as you approach the projects deadlines.

HURDLES OF STANDARDIZATION

What makes SAS® such an invaluable tool is the robust capabilities of its software. Within the ETL environment, this can present a problem for optimizing the data migration process. A prime example is the varied experience and methods of developers. A reoccurring theme is for developers to provide solutions to data migration/manipulation in many different fashions. Although, the desired outcome is accomplished, the solution lacks consistency. This can cause problems for future development where changes are needed in multiple places. Debugging the code also presents challenges. Having multiple solutions to the data migration process forces a development team to be cognizant of each developers' efforts, ultimately, resulting in time wasted.

Other environmental constraints, such as time, make a unified standardization process essential. A pivotal driver for the banking/financial industry is regulatory and internal reporting. The delivery expectation of these reports, whether annually, quarterly, or even daily, creates an inherent time constraint for ETL developers. The data delivery process plays a critical role as the upstream source for multiple analytical processes that need data quickly and accurately.

Many times you also have information restraints in which you are not given all the information you need to make informed decisions on standardization. This can include gaps in sourcing data, or changeable spreadsheets as input into your data stream. You also may not be put in touch with the correct Subject Matter Experts (SME's) who can more quickly help you resolve your issues.

Many industry-leading financial institutions utilize vast amounts of resources to combat time constraints, but resources should still be considered when determining the hurdles to standardization. Resources can consist of employees, geographical locations, and technology. The obvious hurdle of not having enough employees to satisfy deadlines is an ever present concern. Communication lags/gaps need to be considered as a resource constraint when multiple geographical locations are present. While cost

efficiency must be a consideration, the data delivery platform is limited by the technology a company incorporates.

It is likely that you will never work on a development process in which you can overcome all of the above hurdles (think of it as a cone zone on the ETL developmental highway) but when you are aware of the many hurdles in your way (and with more experience) you can at least try to address them and ultimately will improve the process.

BEFORE YOU START CODING

If you were to set out on a long road trip by car, do you just get in the car and drive? Usually it requires a lot of preparation including consulting road maps to find a route, having the car serviced, planning stops packing food and drinks as well as other tasks.

In the same way, when developing an ETL solution, there are many steps to take before setting out on the journey of coding it.

The most important objective for a team is to know what they are ultimately trying to accomplish. This might seem obvious, but we have worked on many projects where this was not readily apparent. There must be an understanding of how to migrate from the business requirements to the target solution. A high level approach used to accomplish this task is through design and data modeling. During the design process, it is useful to understand the data being delivered and how it will be sourced.

The ultimate destination is usually defined in some final deliverables or milestones in the project plan. It is not enough to know that you have to produce a report or database in the end, you also need to know the content/structure of that report or database.

Once you know the destination, it is important to understand the starting point or sources that are going into your process. Our philosophy is that everything should have an ultimate source in some system. If someone has the data in a spreadsheet for example there should be a system source or sources that went into the spreadsheet. Most times we never find all the sources at least in the initial development, but strive to find an automated system source for as much of the data that you can to avoid problems in the future.

Once you know the starting point(s) and the destination, you will need to put together the route that will get you there. And of course you will create the legend (standardization) that you will use in the process.

Practical experience proves that the design phase is one of the most important steps in a project, but it is also the one step that usually gets short-changed. We have found the more time you spend mapping the process out before you go full speed (or higher) trying to code it, the simpler it will be to put together the actual process.

And as we will discuss in the next section you should consider EVERYTHING in the standardization and design process.

METHODS OF STANDARDIZATION

The design phase before coding allows the methods of standardization to be put into place. During the migration process, it is typical for data to come from many different systems and in many different formats. This is where the value of the standardization process comes to fruition. The objective is to ensure every source adheres to the naming conventions and structure of the target.

Note: The following standardization methods are examples and not necessarily hard and fast rules.

Standardize Libraries--a target can be a single table or input table to a star schema data mart model. It is recommended to come up with a standardized way to name the various target tables using libnames as well as table names to achieve this. For instance let's say there are three main steps to your process:

1. Extract from source
2. Transform data
3. Load Data to Database

You might consider putting each set of data in a separate library e.g. EXTRACT, TRANSFRM, and LOAD. This would separate the data into logical areas to go back and find the information. It also allows you to put security on areas where users cannot see the data (Extract and Trans libraries) but have visibility to the final output. You could also have additional libraries for each end user application from the main database.

Standardize Table Names—Once the libraries have been determined you can then use a naming convention for the tables. For instance You can use a prefix for each step followed by information about where the data is coming from or going to, add a frequency (daily, weekly, monthly) and finally a target table name.

Here is an extract library table name: EXTR_ROADMAP_WKLY_ROUTES

This data will be used to transform the data which is then sent to the TRANS library as TRNS_ROADMAP_WKLY_ROUTES

Or maybe it's combined with additional sources or filtered and your output will be TRNS_TEXAS_WKLY_ROUTES

This data may be then loaded to the LOAD library as LOAD_TEXAS_ROUTES.

These will vary but the point is to come up with a consistent way of naming your tables.

Standardize Variable Names—The difficulty of this step is that you are stuck with the source variables that are coming in from the source. At many clients, the extraction step is simply reading in the source table as is and landing an exact copy on our server. This preserves the data that is used in the rest of the process and creates an audit trail even if the source table changes later.

In this case you may not want to make many if any changes to the variable names in the extract. That in itself is a standardization rule. However in the Transform and Load steps you will want to move towards truly standardizing the data.

This is simply a matter of naming conventions. If an amount is used, always give the variable name a suffix of _AMT or _AM or even _AMOUNT. But make it consistent. You may also want to consider that the last suffix is always the same number of characters say three _AMT for Amount, _CDE for Code, _PCT for Percent, _FLG for Flag, IND for indicator etc.

Speaking of Flags and indicators, try to be consistent with the values that can go into either. A flag may be what you use to indicate either a "Yes" or "No" condition (usually "Y" or "N") or maybe a True/False Condition ("T" or "N"). An indicator may may just be a binary switch 0 or 1. Our suggestion is to either employ one of these methods consistently or if more than one is used than create a separate suffix for each. This makes it easier to know what to expect from a Flag or Indicator.

As for the rest of the name, try to keep consistent with like naming. We work with Mortgages a lot and some of the common elements are the terms "Loan", "Mortgage", "Loan to Value", "Principal", "Interest" etc. So if you abbreviate Mortgage as Mort in one place use the same abbreviation elsewhere, otherwise you will see Mortgage, Mort, and mrtg which makes it more difficult to follow.

Coding Standards— One of the core concepts of the standardization process is that you create and maintain written coding standards where consistent coding and methods of development are defined. For instance, a variable data type needs to be converted to match the target structure. It is important, that all developers apply this conversion in a consistent manner. This consistency will add ease when new sources are introduced or updated logic needs to be applied.

A recommend naming convention for a standardized STD_<SOURCE>_<FREQUENCY>_<TARGET_TABLE>.sas7bdat. In regards to standardizing variables, it is important that all columns and associated data types be converted to adhere to the naming conventions of the target specified in the design documentation.

One of the core concepts of the standardization process is that coding standards and consistent methods of development are defined. For instance, a variable data type needs to be converted to match the target

structure. It is important that all developers apply this conversion in a consistent manner. This consistency will add ease when new sources are introduced or updated logic needs to be applied.

USING BASE SAS

Often companies conduct ETL development using base SAS. This is a viable option, but it is important that methods among developers are consistent. SAS is a great tool and allows processing of data to be handled in many different ways. As a result, incorporating functions into the standardization process must be addressed. For example, if data needs to be concatenated, how does a developer choose between the catx, cats, catt, or the concatenation operator '||'. They are all slight variations to the task trying to be accomplished, but could jeopardize the accuracy of the data if done different ways. Another example could be the option to use the compress, trim, or strip functions. They all could serve their purpose, but could compromise the data integrity. Standardization must be conducted consistently during the data migration process.

A more robust method of ensure data is standardized consistently is to utilize the SAS® macro facility. A key advantage to using macros is that the logic or functionality applied to the data is centralized to a single location. This allows for data manipulation across multiple areas of the ETL process. Any logical or functional updates that need to be made only have to be applied once. Often data from multiple sources does not follow the same data standards. A common example of this is how loan to value ratios are sourced. Sometimes ltv values are received as a whole number while other sources may express the data as a ratio. It is important to have an agreed method of expressing the value for further calculations and reporting. Without utilizing a macro, we lose standardization commonality and must make updates in multiple locations. We can standardize the values and column naming conventions using predefined macros in the below example:

```
data SOURCE_A;
      SRC_LTV=72.519;OUTPUT;
      SRC_LTV=22.044;OUTPUT;
run;

data SOURCE_B;
      SRC_LTV_SCORE=0.45223;OUTPUT;
      SRC_LTV_SCORE=0.92586;OUTPUT;
run;

proc sql;
      create table std_target_1 as
      select * , SRC_LTV/100 as ltv_ratio
            from SOURCE_A
            union all
      select * , SRC_LTV_SCORE as ltv_ratio
            from SOURCE_B
            order by 2;
quit;

%macro ltv_format (input_column=default);
      ifn(&input_column <= 1, &input_column, &input_column/100)
%mend ltv_format;

proc sql;
      create table std_target_2 as
      select A.*, %ltv_format(input_column=SRC_LTV) as ltv_ratio
            from SOURCE_A AS A
            union all
      select B.*,%ltv_format(input_column=SRC_LTV_SCORE) as ltv_ratio
            from SOURCE_B AS B
            order by 2;
```

```
quit;
```

Another advantage of incorporating the macro facility to the standardization process is to automatically define macros when starting a base SAS session. This can be accomplished by calling macro definitions in the autoexec.sas file with the %include statement. This method ensures that all developers are utilizing the same macro definitions on a server. Below is an example of the macro definition on the ETL server:

```
/*-----------------------------------------------*/
/* Name: LTV_FORMAT                              */
/* Purpose: Standardize LTV values               */
/*                                               */
/* Assumptions: Data will be expressed           */
/*              as a whole number or ratio       */
/*                                               */
/* History:                                      */
/* 01 Jan 2015 Avery Long – Initial Version      */
/*-----------------------------------------------*/
%macro ltv_format (input_column=default);
ifn(&input_column <= 1, &input_column, &input_column/100)
%mend ltv_format;
```

We can also utilize macros to derive new fields in the target table structure. As an example, we can bucket the above ltv values above into a range of values. The below example shows how to create the new variable. As mentioned above, the macro definition can be used in the autoexec.sas file.

```
%macro derived_ltv_tier (input_column=default);
CASE
      WHEN &input_column <  0    THEN '<0%'
      WHEN &input_column <= 0.25 THEN '0% - <=25 %'
      WHEN &input_column <= 0.50 THEN '25.1% - <= 50%'
      WHEN &input_column <  0.75 THEN '50.1% - <= 75%'
      WHEN &input_column >= 0.75 THEN '> 75%'
      ELSE 'OTHER'
END
%mend;

proc sql;
      create table drv_target as
            select *, %derived_ltv_tier(input_column=ltv_ratio) as ltv_tier
            from std_target_2;
quit;
```

Control Tables

Another way to standardize data is to utilize control tables. A common example is when you have multiple sources:

SRC1_STD

ACCT_ID	SRC_SYS_CD	ACCT_STAT_CD
00450001	001	OPEN
00450002	001	OPEN
00450003	001	OPEN
00450004	001	CLOSED

SRC2_STD

ACCT_ID	SRC_SYS_CD	ACCT_STAT_CD
00999069	002	1
00999070	002	1
00999090	002	1
00999084	002	2

In the above example, the values for ACCT_STAT_CD are completely different in the tables (but have the same variable name). In order to standardize this you can create a control table:

CNTL_STANDARDIZE

SRC_SYS_CD	SRC_VAR_NM	DDS_VAR_VL
001	OPEN	O
001	CLOSED	C
002	1	O
002	2	C

By joining the data from each dataset to the control table and joining on:

```
SRC1_STD.SRC_SYS_CD = CNTL_STANDARDIZE.SRC_SYS_CD
AND SRC1_STD.ACCT_STAT_CD = CNTL_STANDARDIZE.SRC_VAR_NM
```

You will get the resulting dataset:

STD_RESULT

CCT_ID	SRC_SYS_CD	ACCT_STAT_CD	DDS_VAR_VL
00450001	001	OPEN	O
00450002	001	OPEN	O
00450003	001	OPEN	O
00450004	001	CLOSED	C
00999069	002	1	O
00999070	002	1	O
00999084	002	2	C
00999090	002	1	O

Most likely you will remove the ACCT_STAT_CD from the select statement, because it will result in type match errors, however it was left in above for

```
illustrative purposes.  The resulting value DDS_VAR_VL would also be restated
as ACCT_STAT_IND.
```

MOVING TO DATA MANAGEMENT PLATFORM

The same standardization practices applied in base SAS® can also be utilized with a data migration platform, specifically the SAS® Data Integration Studio client and server in conjunction with SAS Management Console. Using it well, can help standardize the code and make it more consistent across the various developers.

Connection Profiles

Connections to other servers can be setup once using SAS Management Console. When connecting, it will use the stored credentials of the individual user to access, so it is important that each developer have their own User ID and Password to the server.

Library Objects

Libraries are also set up once which means all code will use the same libname references for all jobs. If a libname reference or destination is changed or moved, all references will altered within the Data Integration Studio code.

Metadata Objects:

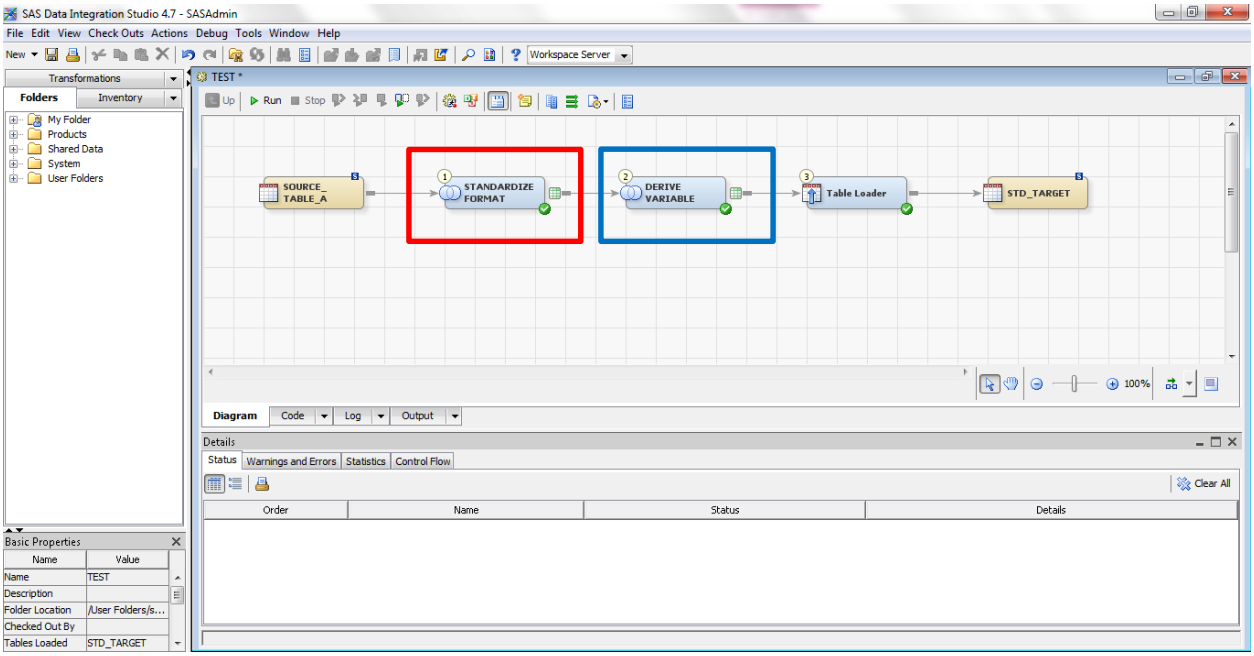
In Data Integration Studio, you can define the attributes and variable names in tables consistent across the code, by defining metadata objects with the attributes built into it for permanent tables. As you build the transformations from step to step DI Studio will automatically propagate the variables from the input tables of the step into the target tables. They can also be modified, added or deleted as your process moves through all of the steps. As with libraries, any changes to the metadata objects will alter the code within all programs that reference the metadata. Be careful though, it can cause errors if not coordinated across all developers using the object.

Standard Transformation Objects

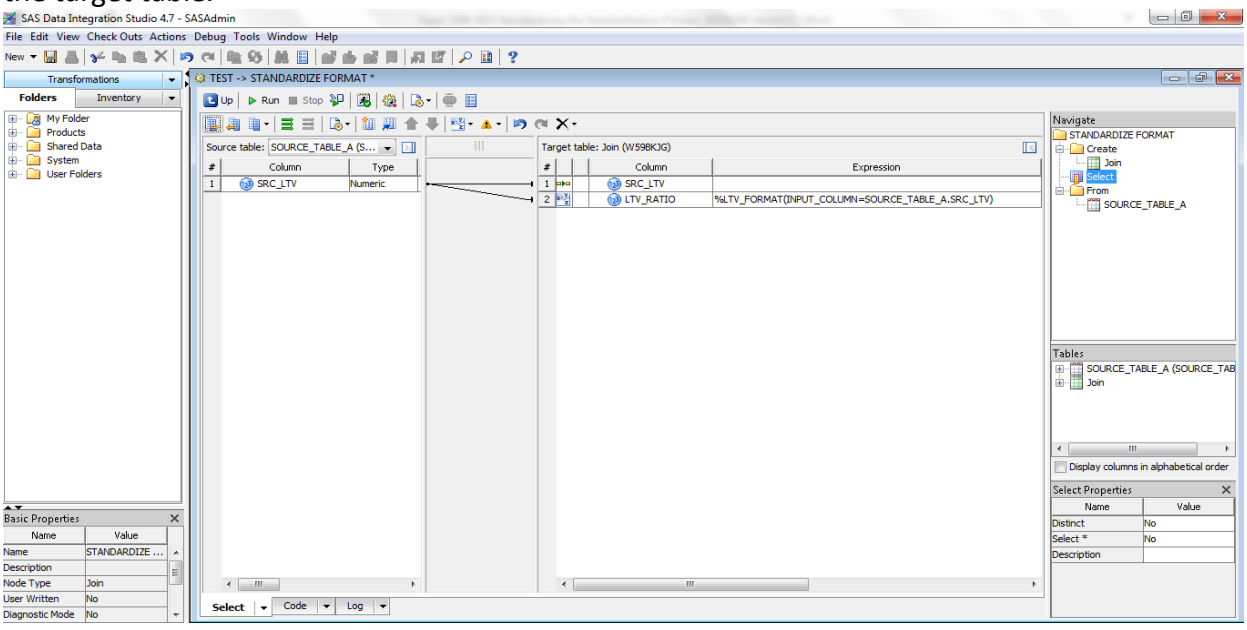
There are many standard transformations within DI Studio which will drop code in and be consistent. Most Transformations have tabs for options that are allowable within the transformation. It is also possible to add user written code to steps, but for standardization purposes, it is usually better to stay within the parameters of the built in transformations and options within them.

We will explore a couple of examples of how to use this to standardize your processes within the confines of the standard transformations.

The following is an example of how the same macro in the previous section can be applied using SAS Data Integration Studio®. On the canvas of the job, the join transformations contain the macro definitions.



Within the “STANDARDIZE FORMAT” join transformation we can see the input column (SRC_LTV) connected to the target column (LTV_RATO). The macro is called in the expression of the target table.

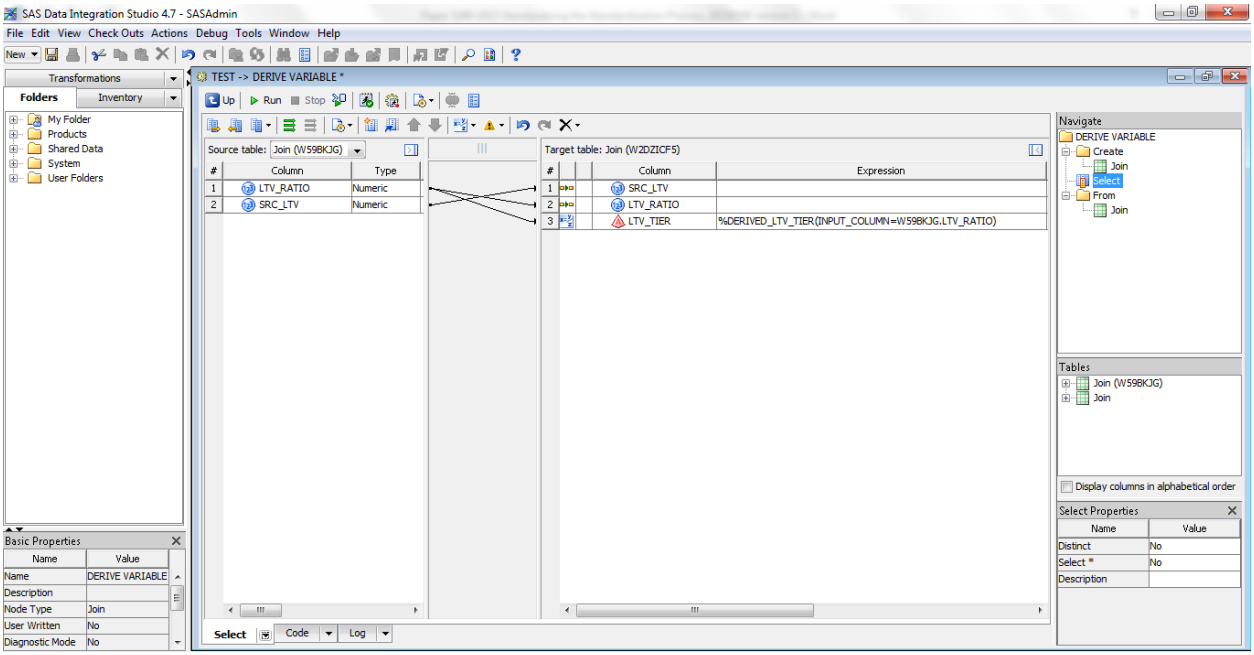


The below code is automatically generated in the code tab of the join transformation.

```
proc sql;
   create table work.W59BKJG as
   select
     SOURCE_TABLE_A.SRC_LTV length = 8,
     %LTV_FORMAT(INPUT_COLUMN=SOURCE_TABLE_A.SRC_LTV) as LTV_RATIO length=8
   from work.SOURCE_TABLE_A as SOURCE_TABLE_A
```

```
    ;
quit;
```

Within the “DERIVE VARIABLE” join transformation we can see the input column (LTV_RATIO) connected to the target column (LTV_TIER). The macro is called in the expression of the target table.



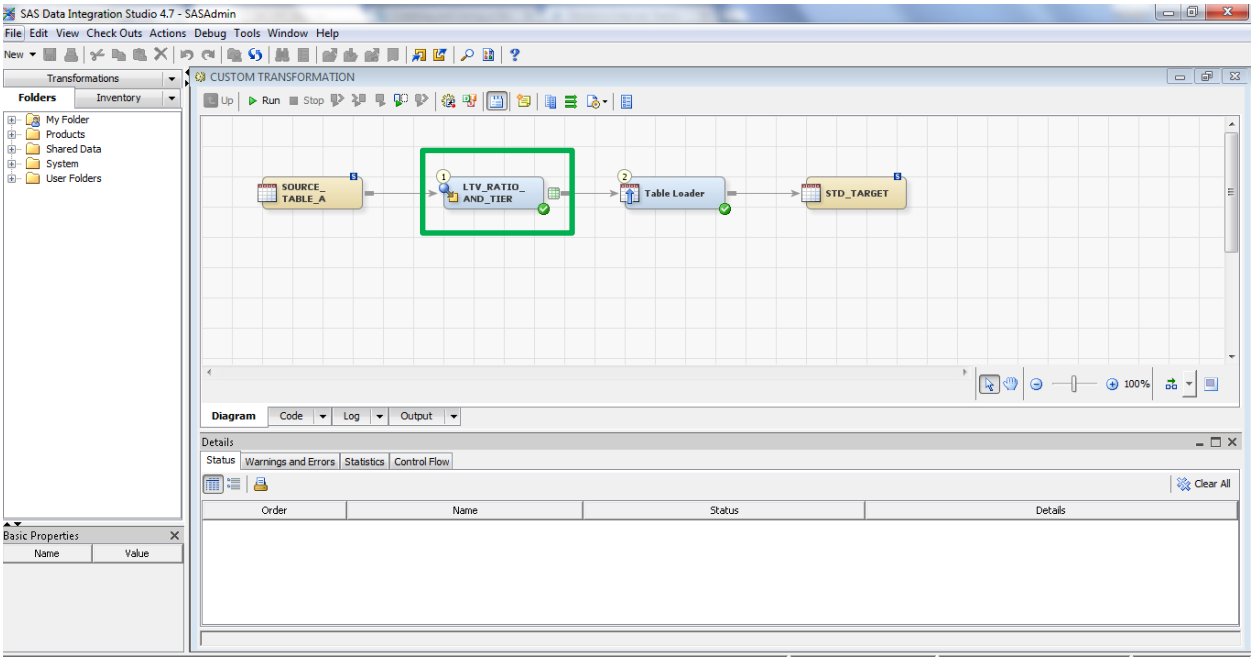
The below code is automatically generated in the code tab of the join transformation.

```
proc sql;
   create table work.W2DZICF5 as
   select
     W59BKJG.SRC_LTV length = 8,
     W59BKJG.LTV_RATIO length = 8,
     %DERIVED_LTV_TIER(INPUT_COLUMN=W59BKJG.LTV_RATIO) as LTV_TIER length=8
   from
      work.W59BKJG as W59BKJG
   ;
quit;
```

Custom Transformation Objects

There is also the option within SAS Data Integration Studio® to build your own custom transformation objects. These transformations allow users to apply standardization practices that go beyond the limitations of standard transformations in a single location. They are used within metadata jobs like any other SAS Data Integration Studio® transformation and are advantageous because they can be used across multiple locations within the ETL process.

The following is an example of how the same macros in the previous section can be applied using a custom transformation within SAS Data Integration Studio®. On the canvas of the job, the custom transformation builds the output intermediate table with the standardization and derivation logic.



The below code is automatically generated in the code tab of the custom transformation.

```
proc sql;
   create table work.std_format as
   select
     &_INPUT..SRC_LTV length = 8,
     %LTV_FORMAT(INPUT_COLUMN=&_INPUT..SRC_LTV) as LTV_RATIO length=8
   from &_INPUT
   ;
quit;

proc sql;
   create table &_OUTPUT as
   select
     &syslast..SRC_LTV length = 8,
     &syslast..LTV_RATIO length = 8,
     %DERIVED_LTV_TIER(INPUT_COLUMN=&syslast..LTV_RATIO) as LTV_TIER
length=8
   from
      &syslast
   ;
quit;
```

With so much emphasis being placed on analytics today, it is easy for the ETL process to be overlooked. Data migration is an essential element to many functionalities within the banking/financial industry. The standardization process is pivotal within data migration and there are many ways to hurdle ETL headaches as long as a strong standardization process is in place. Development becomes more time efficient and data accuracy is maintained throughout the process. Uniform coding standards can be applied in base SAS® and data migration

platforms, such as Data Integration Studio®. These benefits can only be realized with useful design, planning, and road map. Further standardization practices can be utilized with the SAS DataFlux Studio® client and server.

With so much emphasis being placed on analytics today, it is easy for the ETL process to be overlooked. Data migration is an essential element to many functionalities within the banking/financial industry. The standardization process is pivotal within data migration and there are many ways to hurdle ETL headaches as long as a strong standardization process is in place. Development becomes more time efficient and data accuracy is maintained throughout the process. Uniform coding standards can be applied in base SAS® and data migration platforms, such as SAS Data Integration Studio®. These benefits can only be realized with useful design, planning, and road map.

Further standardization practices can be utilized with the SAS DataFlux Studio client and server which is now part of the Data Management Platform within SAS in 9.4.

CONTACT INFORMATION

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

Avery Long

Financial Risk Group

Work Phone: 919-452-8555

E-mail: avery.long@frgrisk.com

Frank Ferriola

Financial Risk Group

Work Phone: 303-548-0278

E-mail: Frank.Ferriola@frgrisk.com

320 S. Academy St.

Cary, NC 27511

Fax: 704-800-7472

Web: http://www.frgrisk.com