The Joinless Join;
Expand the Power of SAS® Enterprise Guide® in a New Way

Kent Ronda Team Phelps, The SASketeers, Des Moines, Iowa
Kirk Paul Lafier, Software Intelligence Corporation, Spring Valley, California

ABSTRACT

SAS Enterprise Guide can easily combine data from tables or data sets by using a Graphical User Interface (GUI) PROC SQL Join to match on like columns or by using a Base SAS® Program Node DATA Step Merge to match on the same variable name. However, what do you do when tables or data sets do not contain like columns or the same variable name and a Join or Merge cannot be used?

We invite you to attend our presentation on the Joinless Join where we teach you how to expand the power of SAS Enterprise Guide in a new way. We will empower you to creatively overcome the limits of a standard Join or Merge. You will learn when to utilize and how to design a Joinless Join based upon dependencies, indirect relationships, or no relationships at all between the tables or data sets. Come experience the power and the versatility of the Joinless Join to greatly expand your data transformation and analysis toolkit.

We look forward to introducing you to the surprising paradox of the Joinless Join.

INTRODUCTION

The tagline for SAS is The Power To Know®, and your 'power to know' greatly expands with your ability to access, combine, and analyze important data from tables or data sets (referred to as tables going forward). The Power To Know sets off The Power To Create which leads to The Power To Automate – much like an intricate and fluid domino design. However, this power will quickly become disjointed if you do not know how to effectively Join or Merge tables of data – even when the tables do not have a relationship.

Here are 2 questions to ask yourself when analyzing 2 or more tables:

❖ Do the tables contain like columns or the same variable name which can be utilized in a Join or Merge?
❖ If the tables do not contain like columns or the same variable name and a standard Join or Merge cannot be used, have I reached a cavernous and insurmountable ‘woe is me’ research impasse in my data analysis?
There is no need to fear, the Joinless Join is here!

The Joinless Join will bridge your research impasse and empower you to:

- Creatively overcome the limits of a standard Join or Merge
- Access, combine, and analyze tables for the first time based upon dependencies, indirect relationships, or no relationships at all
- Open up new worlds of table creations, calculations, validations, and filtrations
- Increase your ability to detect and resolve errors including hidden errors
- Prevent validation process failure – yea! – and completely... yes, completely automate your projects

The SAS project in this presentation demonstrates:

The Power To Know when to utilize and how to design a Joinless Join
The Power To Create tables based upon dependencies, indirect relationships, or no relationships at all
The Power To Automate projects even when tables cannot be directly joined or merged

We invite you to journey with us as we help you EXPAND the power of SAS Enterprise Guide in a new way.

Brief Overview of Standard PROC SQL Joins and DATA Step Merges

Just traveling along... side-by-side.
Harry Macgregor Woods

A standard Join or Merge enables you to gather and manipulate tables of data for exciting insights into data relationships. The process consists of combining tables side-by-side horizontally (illustrated in Figure 1) and matching related rows to bring together some or all of each table’s contents.

A column from each table is used to connect the tables and needs to have the same attributes and like values because the success of a standard Join or Merge is dependent upon these factors. A powerful feature of the relational model is the ability to define relationships between multiple tables and to retrieve information based on these relationships.
Here are some basic differences between standard Joins and Merges –

**Join Features:**
- Code conforms to ANSI guidelines and is portable to other vendor databases
- Data does not need to be sorted using BY-value
- Does not require the same variable name
- Duplicate matching column is not automatically overlaid
- Results automatically print unless NOPRINT option is specified

**Merge Features:**
- Relevant only to SAS Software and is not portable to other vendor databases
- Data must first be sorted using BY-value
- Requires the same variable name
- Duplicate matching column is automatically overlaid
- Results do not automatically print
- More steps are often needed than with the SQL procedure

There are also Syntax and Operational differences between standard Joins and Merges –

**Inner Join Features:**
- Symmetrical process of relating rows in 2 or more tables
- Maximum number of tables that can be specified is **256**
- Uses the WHERE-clause

**Outer Join and Merge Features:**
- Asymmetrical process of relating rows in 2 tables
- Maximum number of tables that can be specified is **2**
- Uses syntax keywords such as LEFT JOIN, RIGHT JOIN, and FULL JOIN
- Uses the ON-clause

An **Inner Join or Merge** is a symmetrical process of matching related rows in tables – an Inner Join can match related rows in **2 to 256** tables, and a Merge can match related rows in **2** tables.

The result of an **Inner Join or Merge** produces only matched rows from the tables. The result is illustrated by the shaded area AB in **Figure 2**.

**Figure 2. Venn Diagram – Inner Join or Merge**
An **Outer Join or Merge** is an asymmetrical process of matching related rows in 2 tables. Like an Inner Join or Merge, an Outer Join or Merge can match related rows in tables. However, this is where the similarities end because the resulting set of data from an Outer Join or Merge also contains **unmatched** rows from the left, right, or both tables. The ability to preserve unmatched rows is the major difference in the Outer Join and Merge constructs.

The result of a **Left Outer Join or Merge** produces matched rows from both tables while preserving all unmatched rows from the left table. The result is illustrated by the shaded areas A and AB in **Figure 3**.

The result of a **Right Outer Join or Merge** produces matched rows from both tables while preserving all unmatched rows from the right table. The result is illustrated by the shaded areas B and AB in **Figure 4**.

All of these Joins and Merges have an important common denominator – each of them requires a like column or the same variable name to match on. Thus, we now return to the core focus of this presentation...

**Figure 3. Venn Diagram – Left Outer Join or Merge**

**Figure 4. Venn Diagram – Right Outer Join or Merge**

**Figure 5. Venn Diagram – Tables Without Like Columns or the Same Variable Name**

What do you do when the tables you want to analyze do not contain like columns or the same variable name and a standard Join or Merge cannot be used?

**In the next section we will continue to follow The Power To Know dominoes to find the answer.**

Professor Domino will be our guide 😊
Illuminating the Paradox of the Joinless Join

Sometimes success is seeing what we already have in a new light.  
Dan Miller

The development of the Joinless Join came about during a recent project when the need arose to overcome the limitations of a standard Join and to resolve unforeseen issues which occurred with a One-Way Frequency.

SAS Highlight

A One-Way Frequency contains a distribution list of values, counts, and percentages for a column.

Here is our SAS Enterprise Guide project example:

- Our project example demonstrates 7 ways to use a Joinless Join.
We design a Program Node to create a source table:

```plaintext
DATA SMILEY_COMPANY;
   LENGTH Special_Person $20 Special_Number 8 Special_Code $1 Load_Date 8;
   FORMAT Load_Date date9.;
   INFILE DATALINES DELIMITER=',';
   INPUT Special_Person $ Special_Number Special_Code $ Load_Date;
DATALINES;
Smiley,10127911,E,19362
Smiley's Son,10173341,K,19362
Smiley's Twin,10376606,B,19362
Smiley's Wife,10927911,A,19362
Smiley's Son,11471884,E,19362
Smiley's Twin,11573691,G,19362
Smiley's Daughter,11975386,C,19362
Smiley's Son,12071884,J,19362
Smiley's Son,12871884,D,19362
Smiley's Twin,13173691,A,19362
Smiley's Wife,13771202,D,19362
Smiley's Daughter,13775498,H,19362
Smiley's Son,14171884,I,19362
Smiley's Twin,15373691,F,19362
Smiley's Son,15573691,D,19362
Smiley's Son,16074330,H,19362
Smiley's Daughter,16175498,B,19362
Smiley's Wife,16176964,I,19358
Smiley,16279111,E,19362
Smiley's Twin,16573691,K,19362;
RUN;
```

♦ This is the code you will need to recreate this table.

The Program Node creates the SMILEY_COMPANY source table:

- The SMILEY_COMPANY table is used throughout this presentation.
- This table contains each Special Person, Special Number, and Special Code of the ☹ Smile Company ☹ employees.
- Load_Date is the date when each row was created.
This Query creates the SMILEY_CONTROL_VALUE table:

A Control Value table is created in which
Computed Columns are set to 1 if any data
is missing in the SMILEY_COMPANY table:

Special_Person_Flag:
CASE
WHEN t1.Special_Code = ' ' THEN 1
ELSE 0
END

Special_Number_Flag:
CASE
WHEN t1.Special_Number = 0 THEN 1
WHEN t1.Special_Number is missing
THEN 1
ELSE 0
END

Special_Code_Flag:
CASE
WHEN t1.Special_Code = ' ' THEN 1
ELSE 0
END

Load_Date_Flag:
CASE
WHEN t1.Load_Date = . THEN 1
ELSE 0
END

The output table contains 1 row:

Notice how the Special_Code_Flag is set to 1 because the Special_Code is missing from this row.
A One-Way Frequency is run using the 4 flags:

This One-Way Frequency is setup to automatically send an email when this project is run.

Then one day NOTHING was missing from the SMILEY_COMPANY table...

To replicate this scenario you will need to perform the following:

- Replace the Smiley,10127911, ,19362 DATALINE with Smiley,10127911,A,19362 in the SMILEY_COMPANY Program Node on Page 6 and rerun to have no missing data in the table.
- Rerun the Query for the SMILEY_CONTROL_VALUE table and the Control Value Report One-Way Frequency.
Here is the empty SMILEY_CONTROL_VALUE table:

- Since nothing is missing from the SMILEY_COMPANY table, all of the flags are set to 0 which filters out all of the rows causing the SMILEY_CONTROL_VALUE table to be created empty.
- Do you know what happens when the SMILEY_CONTROL_VALUE table is created empty?

Note the Red X in the upper left corner of the One-Way Frequency output:

- At first glance, it appears the report ran correctly - but remember, the input to the Control Value Report was created empty.
- If the input is empty, then what are we seeing? Notice the Warning Message which appeared:
  
  ![Warning Message]

  This report is from an older run of the task or program. Output from the most recent run was not available.

- This warning message unfortunately means that we are looking at the previous successful run of this One-Way Frequency instead of the current results which we are seeking.
It was ironic when the Smiley_Company table processed error free, no data was missing for the first time, that the resulting empty Smiley_Control_Value table caused the One-Way Frequency to not run! Consequently, the previous results were generated on the monthly report instead of the current results.

Here is a review of the One-Way Frequency issue before we explore the solution:

- When data is missing in the Smiley_Company table a row is created in the Smiley_Control_Value table with the column flags set to 1.
- When the Smiley_Control_Value table is populated with at least 1 row the One-Way Frequency runs correctly and generates current results.
- However, when data is not missing from the Smiley_Company table no rows are created in the Smiley_Control_Value table.
- When the Smiley_Control_Value table is created empty the One-Way Frequency does not run correctly and does not generate current results but instead displays the previous results.
- In summary, the One-Way Frequency runs correctly and generates current results only when the Smiley_Control_Value table is populated with at least 1 row created by missing data detected in the Smiley_Company table.

What to do, what to do...

In response to this dilemma, SAS Intuition kicked in and a quest was undertaken to find a permanent workaround solution that would enable the project to run successfully – even if all the tables were empty.

Here is the solution which arose during the quest to resolve this issue:

- Create a Smiley_Control_Value_Row_Count table with the row count of the Smiley_Control_Value table.
- Create a Smiley_Control_Value_Mock_Row table based upon an indirect relationship between the Smiley_Control_Value_Row_Count table and the Smiley_Company table.
- When the Smiley_Control_Value table is populated with rows, the Smiley_Control_Value_Row_Count table will contain a non-zero row count, and the Smiley_Control_Value_Mock_Row table will be created empty.
- When the Smiley_Control_Value table is empty, the Smiley_Control_Value_Row_Count table will contain a zero row count, and the Smiley_Control_Value_Mock_Row table will be created with 1 mock row of column flags set to 0.
- Append the Smiley_Control_Value table and the Smiley_Control_Value_Mock_Row table to ensure that the appended output is always populated with either real data or mock data instead of being created empty.
- Use this appended output as the input to the One-Way Frequency to enable it to always run correctly and to generate current results.

Always Remember, It's Too Soon To Quit!

Bob Wieland (Mr. Inspiration)
This Query creates the SMILEY_CONTROL_VALUE_ROW_COUNT table with the row count of the SMILEY_CONTROL_VALUE table:

- A Count of Special_Person is used to create the SMILEY_CONTROL_VALUE_ROW_COUNT.
- Automatically Select Groups is selected and no groups are selected to count the rows.

The output table contains 1 row with 1 column:
Create a Smiley_Control_Value_Mock_Row table based upon an indirect relationship between the Smiley_Control_Value_Row_Count table and the Smiley_Company table:

- As the mock row is created, all 4 flags are set to a 0 value meaning nothing is missing.
- Since only 1 mock row is needed, Query limits are set to create 1 output row via the Options.
- A filter is set to create a mock row only if the SMILEY_CONTROL_VALUE table is empty.

Notice there are no columns to Join between the two tables:
No Problem –
We will use a Joinless Join based upon an indirect relationship between the tables.

How the Joinless Join works:

- The SMILEY_CONTROL_VALUE_ROW_COUNT table indirectly relates to the SMILEY_COMPANY table because it contains the row count of the error rows in the SMILEY_COMPANY table.
- We utilize a Joinless Join to create a Cartesian Product based upon this indirect relationship.

- The Cartesian Product places the SMILEY_CONTROL_VALUE_ROW_COUNT from the SMILEY_CONTROL_VALUE_ROW_COUNT table to the right of each of the 20 rows in the SMILEY_COMPANY table.
SAS Highlight

A Cartesian Product is a result set of all the possible rows and columns contained in 2 or more tables. The resulting set of data can be extremely large and unwieldy. The DATA Step does not easily lend itself to creating a Cartesian Product thus PROC SQL is the desired approach. Its most noticeable coding characteristic is the absence of a WHERE-clause. Although rarely produced, a Cartesian Product Join nicely illustrates a base (or internal representation) for all Joins.

This Warning Message always appears whenever 2 tables are joined with a Joinless Join because SAS knows it will create a Cartesian Product which can take a lot of extra resources.

Caution:

When you design your Joinless Join make sure that one of the tables has only ONE row!

Here is the complete result of the Joinless Join:

Notice that all 4 flags are set to 0 because no data is missing from the SMILEY_COMPANY table.
Append the Smiley_Control_Value table and the Smiley_Control_Value_Mock_Row table to ensure the appended output is always populated with either real data or mock data instead of being created empty:

Notice the Append result matches the Smiley_Control_Value_Mock_Row table – Done & Done!

We have achieved our desired results and we have a new input to the One-Way Frequency.

The One-Way Frequency is recreated using the appended table:
Here is the One-Way Frequency output with the 4 flags:

The One-Way Frequency correctly displays that all 4 flags are set to 0 and therefore no data is missing – thanks to the Joinless Join 😊.

Yea!!!

*Strike up the band,*

*Toss the confetti,*

*Release the balloons!*

*Applause... Applause... Applause...*

*Bring out the treats for everyone!*

😊 Oh but wait... your new friend, the Joinless Join, is just getting started! 😊
Next we design another Program Node to create 3 additional tables:

``` SAS
DATA Special_Number_National_Average
    (KEEP=Special_Number_National_Average)
   Load_Date_Check (KEEP=Load_Date_Check)
   Special_Code_National_Focus
    (KEEP=Special_Code_National_Focus);
LENGTH Load_Date_Check 8;
FORMAT Load_Date_Check date9. ;

Special_Number_National_Average = 12000000;
OUTPUT Special_Number_National_Average;

Load_Date_Check = '01JAN2013'd;
OUTPUT Load_Date_Check;

Special_Code_National_Focus = 'K';
OUTPUT Special_Code_National_Focus;
RUN;
```

- This is the code you will need to recreate these tables.

Here are the 3 additional tables the Program Node creates:

- The Special_Number_National_Average table contains the average of all the Special_Number columns from each Smiley Company nationwide which we will use in a Joinless Join to calculate a percentage of the Special_Number column in our SMILEY_COMPANY table.
- The Load_Date_Check table contains a Load Date which we will use in a Joinless Join to validate that all of our SMILEY_COMPANY table rows were created in 2013.
- The Special_Code_National_Focus table contains a Special Code from the Smiley Company National Headquarters which we will use in a Joinless Join to filter our SMILEY_COMPANY table output.
Designing a Joinless Join to perform a Calculation:

- Build a Query with our SMILEY_COMPANY table and the Smiley Company National Headquarters SPECIAL_NUMBER_NATIONAL_AVERAGE table.

- The Joinless Join is based upon the SPECIAL_NUMBER_NATIONAL_AVERAGE table indirectly relating to our SMILEY_COMPANY table because it contains the average of all the Special_Number columns from each SMILEY_COMPANY table nationwide.

- The Joinless Join automatically creates a Cartesian Product which places the 1 row and 1 column of the SPECIAL_NUMBER_NATIONAL_AVERAGE table to the right of each of the 20 rows and 4 columns in our SMILEY_COMPANY table.
Calculate a Special_Number_Percent Computed Column using the Special_Number column from our SMILEY_COMPANY table and the Special_Number_National_Average column from the Cartesian Product results.

Here is the final result of our SMILEY_COMPANY table with the Special_Number_Percent column to the right of each of the 20 rows and 4 columns.
Designing a Joinless Join to perform a Validation:

- Build a Query with our SMILEY_COMPANY table and the LOAD_DATE_CHECK table.

- The Joinless Join is based upon the LOAD_DATE_CHECK table indirectly relating to our SMILEY_COMPANY table because it contains the valid Load Date that should be found in the Load_Date column in the SMILEY_COMPANY table.

- The Joinless Join automatically creates a Cartesian Product which places the 1 row and 1 column of the LOAD_DATE_CHECK table to the right of each of the 20 rows and 4 columns in our SMILEY_COMPANY table.
Validate a Date_Verification Computed Column using the Load_Date column from our SMILEY_COMPANY table and the Load_Date_Check column from the Cartesian Product results.

Here is the final result of our SMILEY_COMPANY table with the Special_Number_Percent column to the right of each of the 20 rows and 4 columns.
Designing a Joinless Join to perform a Filtration:

- Build a Query with our SMILEY_COMPANY table and the Smiley Company National Headquarters SPECIAL_CODE_NATIONAL_FOCUS table.

- The Joinless Join is based upon the SPECIAL_CODE_NATIONAL_FOCUS table indirectly relating to our SMILEY_COMPANY table because it contains the Special Code to be focused upon nationwide within the Special_Code column in the SMILEY_COMPANY table.

- The Joinless join automatically creates a Cartesian Product which places the 1 row and 1 column of the SPECIAL_CODE_NATIONAL_FOCUS table to the right of each of the 20 rows and 4 columns in our SMILEY_COMPANY table.
Filter the raw data to include the rows where the value of the Special_Code column from our SMILEY_COMPANY table is equal to the value of the Special_Code_National_Focus column from the Cartesian Product results.

<table>
<thead>
<tr>
<th>Special_Person</th>
<th>Special_Number</th>
<th>Special_Code</th>
<th>Load_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smiley's Son</td>
<td>10173341 K</td>
<td></td>
<td>04JAN2013</td>
</tr>
<tr>
<td>Smiley's Twin</td>
<td>16573891 K</td>
<td></td>
<td>04JAN2013</td>
</tr>
</tbody>
</table>

Here is the final result of our SMILEY_COMPANY table with the Special_Code column filtered by the Special_Code_National_Focus column.
Designing a Joinless Join to perform a Mock Row Creation, Calculation, Validation, and Filtration:

Build a Query with our SMILEY_COMPANY table and the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables.

The Joinless Join is based upon the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables indirectly relating to our SMILEY_COMPANY table as shown in the previous examples.
The Joinless Join automatically creates a Cartesian Product which places the 1 row and 1 column of the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables to the right of each of the 20 rows and 4 columns in our SMILEY_COMPANY table.

The Mock Row Creation, Calculation, Validation, and Filtration are represented by Computed Columns which are derived in the same way as shown in the previous examples along with one new Special_Code_Match Computed Column representing Filtration.

Here is the final result with the Flags to the left and the Calculation, Validation, and Filtration Computed Columns to the right of each of the 20 rows and 4 columns.
The One-Way Frequency correctly displays that all 4 flags are set to 0 and therefore no data is missing – thanks to the Joinless Join 🎉.
Designing a Joinless Join to combine 4 tables with No Relationships at all using the 3 additional tables that the 2nd Program Node created and the Smiley_Control_Value_Row_Count table:

- Notice how the 4 columns in the 4 tables have No Relationships at all.

- Build a Query with the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables.

- This time the Joinless Join is based upon the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables having No Relationships at all.
The Joinless Join automatically creates a Cartesian Product which places the 1 row and 1 column of the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables to the right of each other.

Here is the final result from selecting all 4 columns which is equal to the Cartesian Product.
Designing a Joinless Join of the 1 row 4 column table to perform a Mock Row Creation, Calculation, Validation, and Filtration:

- Build a Query with our SMILEY_COMPANY table and the JOINLESS_JOIN NOTHING IN COMMON table.

- The Joinless Join is based upon all 4 columns in the JOINLESS_JOIN NOTHING IN COMMON table indirectly relating to our SMILEY_COMPANY table as shown in the previous examples.
The Joinless Join automatically creates a Cartesian Product which places the 1 row and 4 columns of the SMILEY_CONTROL_VALUE_ROW_COUNT, SPECIAL_NUMBER_NATIONAL_AVERAGE, LOAD_DATE_CHECK, and SPECIAL_CODE_NATIONAL_FOCUS tables to the right of each of the 20 rows and 4 columns in our SMILEY_COMPANY table.

The Mock Row Creation, Calculation, Validation, and Filtration are represented by Computed Columns which are derived in the same way as shown in the previous examples along with one new Special_Code_Match Computed Column representing Filtration.

Here is the final result with the Flags to the left and the Calculation, Validation, and Filtration Computed Columns to the right of each of the 20 rows and 4 columns.
The One-Way Frequency correctly displays that all 4 flags are set to 0 and therefore no data is missing – thanks to the Joinless Join of 1 row with 4 columns 😊.
The **Joinless Join** empowers you to creatively overcome the limits of a standard Join or Merge and enables you to expand the power of SAS Enterprise Guide in a new way. **The Power To Know** when to utilize and how to design a Joinless Join sets off **The Power To Create** tables based upon dependencies, indirect relationships, or no relationships at all which leads to **The Power To Automate** projects even when tables cannot be directly joined or merged. (Try saying that statement really fast for fun 😋.)

The Joinless Join bridges the research impasse you experience when needing to combine data from SAS tables which do not contain like columns or the same variable name. New worlds of table creations, calculations, validations, and filtrations have opened up to greatly expand your data transformation and analysis toolkit. Begin thinking about how you can benefit from the power and the versatility of the Joinless Join.

*How wonderful it is that we need not wait a single minute before starting to improve ourselves and our world!*  
Anne Frank

**SAS Programming** is like a series of intricate and fluid domino designs and you are the **Designer**. Your desire to design a quality program fuels your thoroughness and attention to detail. As a SAS Professional, your inquisitive nature, research oriented mindset, and solution driven focus are among your greatest assets.

*Your life is like a campfire at night -  You never know how many people will see it and be comforted and guided by your light.*  
Claire Draper

*Rule #6: Study hard and learn all you can.*  
♣ Roy Rogers Riders Club Rules ♣

**Always remember** – *It’s not what the SAS world holds for you, it’s what YOU bring to it!* Continue to develop and build on your many skills and talents. Keep looking for different ways to share your God-given abilities and ideas. You will soon discover new and creative ways to design your SAS programs. Plan on coming back to the MWSUG conference next year to shed some light on the exciting things you are learning. All of us are on the SAS journey with you and we look forward to your teaching sessions in the future.

As we conclude our presentation, we want to introduce you to our **SAS Mascot, Smiley**. Smiley represents the **SAS Joy** which each of us experience when we find better ways to accomplish mighty and worthy deeds using SAS. The four of us, along with Professor Domino, hope we have expanded and enriched your SAS knowledge.

*Thank You For Honoring Us With Your Participation*  
😊 Happy SAS Trails To You... Until We Meet Again 😊
MEET THE AUTHORS

Writing is a permanent legacy.

John C. Maxwell

Kent Phelps (Senior Data Governance Analyst) has worked in IT and Data Governance since 1990, has programmed in SAS since 2007, is a SAS Certified Professional specializing in combining and automating the best of SAS Enterprise Guide with Base SAS, has Co-Created/Led Intro To SAS EG classes, and presents SAS News You Can Use. Kent has a B.S. in Electrical Engineering, has studied Transformational Leadership, Dynamic Teamwork, and Personal Growth since 1994, and is certified as a John Maxwell Team coach and a 48 Days To The Work You Love coach. His hope is to encourage you to fulfill your life and leadership potential and to equip you in building an enduring legacy of inspiration, excellence, and honor.

****************************

Ronda Phelps (Writer, Teacher, and Coach) formerly worked in the Banking and Insurance industries for 19 years, has studied Transformational Leadership, Dynamic Teamwork, and Personal Growth since 1994, and is certified as a John Maxwell Team coach and a 48 Days To The Work You Love coach. She believes YOU are a gift that the world is waiting to receive! Her hope is to encourage you to pursue your unique destiny and to equip you in navigating your journey with intentionality, fulfilling purpose, and enduring hope.

****************************

Kirk Paul Lafler (Founder/Senior Consultant, Software Intelligence Corporation) has programmed in SAS since 1979, is a SAS Certified Professional, provides IT Consulting Services, trains and mentors SAS users worldwide, and is a SAScommunity.org Emeritus Advisory Board member. Kirk has authored 6 books including Google Search Complete! (Odyssey Press 2014), has written over 500 papers and articles, has been invited to speak and/or train at over 400 SAS international, regional, special-interest, local and in-house user group conferences and meetings, and has received 23 BEST Contributed Paper, Hands-On Workshop (HOW), and Poster Awards. His popular SAS Tips column Kirk’s Korner of Quick and Simple Tips and his funfilled SASword Puzzles appear on various SAS websites and in several SAS User Group newsletters.

We invite you to share your valued comments with us:

Kent ♥ Ronda Team Phelps
Writers, Teachers, and Coaches
E-mail: SASketeers@q.com

Kirk Paul Lafler
Senior Consultant, Application Developer, Trainer, Mentor, and Author
Software Intelligence Corporation
E-mail: KirkLafler@cs.com
LinkedIn: http://www.linkedin.com/in/KirkPaulLafler
Twitter: @sasNerd

😊 We Look Forward To Connecting With You In The Future 😊
Here is the process we used to create the 4 Computed Columns in the SMILEY_CONTROL_VALUE table:

- From within the Query click Computed Columns to open the list of Computed Columns.

- Click New to create a New Computed Column.

- We are creating a flag using a CASE statement, so select Advanced expression and click Next.

- Enter the expression while typing or clicking the functions and column names and click Next.
Enter the New Computed Column as the Identifier and Column Name and click Next.

Click Finish and then click Close to close the Computed Column.

The Special_Person_Flag now appears under Computed Columns and in the Selected Data.
Repeat this process to create the 3 additional Computed Columns that are needed.
ACKNOWLEDGMENTS

We want to thank Dave Foster, the 25th Annual MWSUG 2014 BI/CI Section Chair, and Bruce Lund, the Asst. BI/CI Section Chair, for graciously accepting our abstract and paper. In addition, we want to express our appreciation to the Co-Chairs, Cindy Lee (Academic Chair) and Craig Wildeman (Operations Chair), the Executive Committee and Conference Leaders, and SAS Institute for their diligent efforts in organizing this illuminating and energizing conference.

We also offer our deep gratitude to each of you who empower us through your teaching endeavors. Your heart to continuously share what you are learning, blended with your servant leadership and supportive guidance, is a constant light of encouragement to us. You inspire us to keep sharing what we are learning and our hope is to be a light of encouragement to you as well – All for One & One for All.

REFERENCES


### TRADEMARK CITATIONS

SAS and all other SAS Institute, Inc., product or service names are registered trademarks or trademarks of SAS Institute, Inc., in the USA and other countries. The symbol, ®, indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

### DISCLAIMER

We have endeavored to provide accurate and helpful information in this SAS White Paper. The information is provided in ‘Good Faith’ and ‘As Is’ without any kind of warranty, either expressed or implied. Recipients acknowledge and agree that we and/or our companies are not, and never will be, liable for any problems and/or damages whatsoever which may arise from the recipient’s use of the information in this paper.