MWSUG 2016 - Paper HW03

The Joinless Join ~ The Impossible Dream Come True;

Expand the Power of Base SAS® and SAS® Enterprise Guide® in a New Way

Kent ♥ Ronda Team Phelps, The SASketeers, Des Moines, IA
All for SAS and SAS for All!

ABSTRACT

Base SAS and SAS Enterprise Guide can easily combine data from tables or data sets by using a PROC SQL Join to match on like columns or by using a 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 exciting presentation on the Joinless Join where we will teach you how to expand the power of Base SAS and 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 how to design a Joinless Join based upon dependencies, indirect relationships, or no relationships at all between the tables or data sets. In addition, we will highlight how to use a Joinless Join to prepare unrelated joinless data to be utilized by ODS and PROC REPORT in creating a PDF. 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 using Base SAS and SAS Enterprise Guide
- 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
- Prepare unrelated joinless data to be utilized by ODS and PROC REPORT
- 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 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
E X P A N D
the power of Base SAS and 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 combine tables side-by-side horizontally by matching related rows. A like column or the same variable name, with the same attributes and like values, is used to connect the tables and bring together some or all of each table's contents.

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.

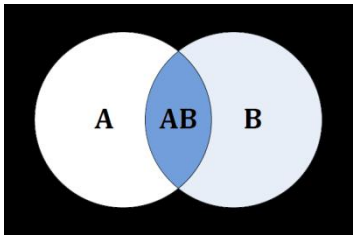
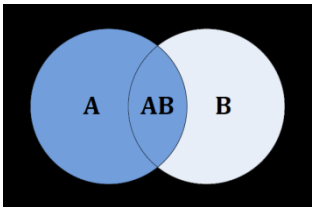


Figure 1. Venn Diagram – Inner Join or Merge

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 1**.

An **Outer Join or Merge** is an asymmetrical process of matching related rows in **2** tables. The resulting set of data also contains **unmatched** rows from the left, right, or both tables.



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 2**.

Figure 2. Venn Diagram – Left Outer Join or Merge

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 3**.

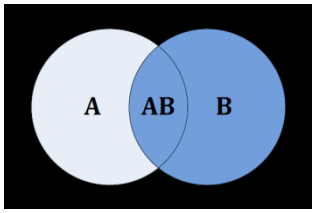
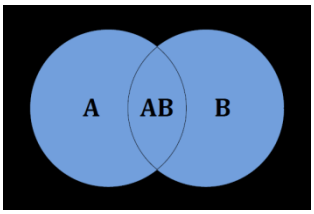


Figure 3. Venn Diagram – Right Outer Join or Merge



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

Figure 4. Venn Diagram – Full Outer Join or Merge

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

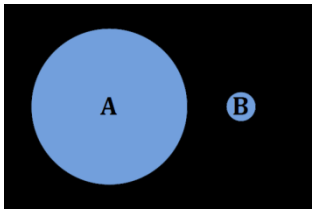


Figure 5. Venn Diagram – Tables Without Like Columns or the Same Variable Name

What do you do when the tables you want to analyze do not contain like columns or the same variable name (**Figure 5**) 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

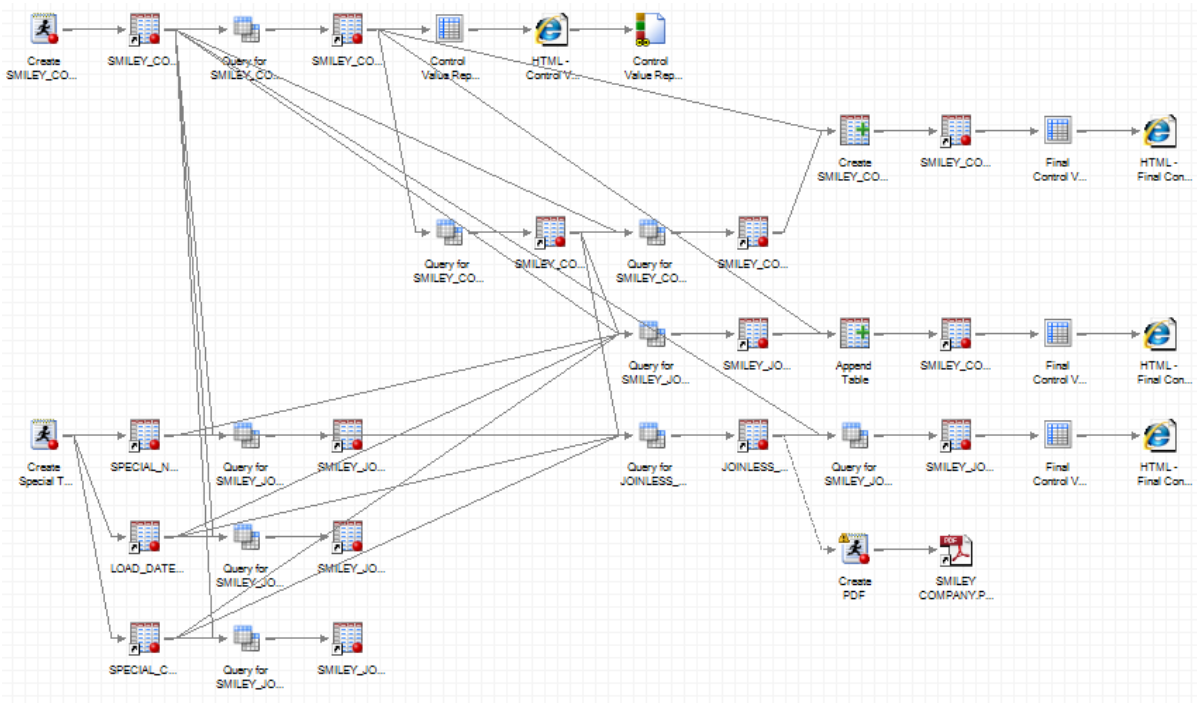
Sometimes success is seeing
what we already have
in a new light.
Dan Miller

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

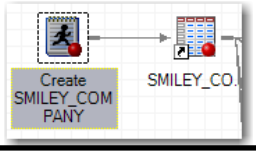
A One-Way Frequency contains a distribution list of values, counts, and percentages for a column.

Here is our SAS Enterprise Guide project example:



- **Our project example demonstrates 8 ways to use a Joinless Join.**

We design a Program Node to create a source table:




```
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, ,20090
Smiley's Son,10173341,K,20090
Smiley's Twin,10376606,B,20090
Smiley's Wife,10927911,A,20090
Smiley's Son,11471884,E,20090
Smiley's Twin,11573691,G,20090
Smiley's Daughter,11975386,C,20090
Smiley's Son,12071884,J,20090
Smiley's Son,12871884,D,20090
Smiley's Twin,13173691,A,20090
Smiley's Wife,13771202,D,20090
Smiley's Daughter,13775498,H,20090
Smiley's Son,14171884,I,20090
Smiley's Twin,15373691,F,20090
Smiley's Son,15471884,C,20090
Smiley's Son,16074330,H,20090
Smiley's Daughter,16175498,B,20090
Smiley's Wife,16176964,I,20088
Smiley,16279111,E,20090
Smiley's Twin,16573691,K,20090
RUN;
```

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

The Program Node creates the SMILEY_COMPANY source table:

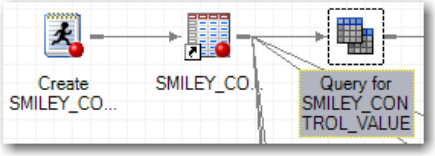
SMILEY_COMPANY

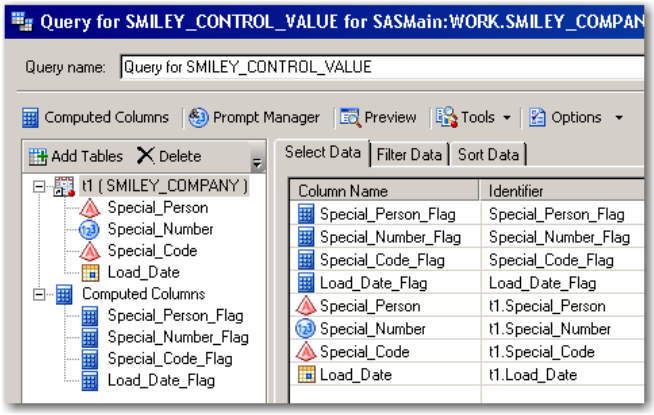
Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Exp

	Special_Person	Special_Number	Special_Code	Load_Date
1	Smiley	10127911		02JAN2015
2	Smiley's Son	10173341	K	02JAN2015
3	Smiley's Twin	10376606	B	02JAN2015
4	Smiley's Wife	10927911	A	02JAN2015
5	Smiley's Son	11471884	E	02JAN2015
6	Smiley's Twin	11573691	G	02JAN2015
7	Smiley's Daughter	11975386	C	02JAN2015
8	Smiley's Son	12071884	J	02JAN2015
9	Smiley's Son	12871884	D	02JAN2015
10	Smiley's Twin	13173691	A	02JAN2015
11	Smiley's Wife	13771202	D	02JAN2015
12	Smiley's Daughter	13775498	H	02JAN2015
13	Smiley's Son	14171884	I	02JAN2015
14	Smiley's Twin	15373691	F	02JAN2015
15	Smiley's Son	15471884	C	02JAN2015
16	Smiley's Son	16074330	H	02JAN2015
17	Smiley's Daughter	16175498	B	02JAN2015
18	Smiley's Wife	16176964	I	31DEC2014
19	Smiley	16279111	E	02JAN2015
20	Smiley's Twin	16573691	K	02JAN2015

- **The SMILEY_COMPANY table is used throughout this presentation.**
- **This table contains each Special Person, Special Number, and Special Code of the ☺ Smiley Company ☺ employees.**
- **Load_Date is the date when each row was created.**

This Query creates the SMILEY_CONTROL_VALUE table:





- **Please see Appendix A to learn how to create Computed Columns and see Appendix B for the Base SAS code which corresponds to each example.**

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
```

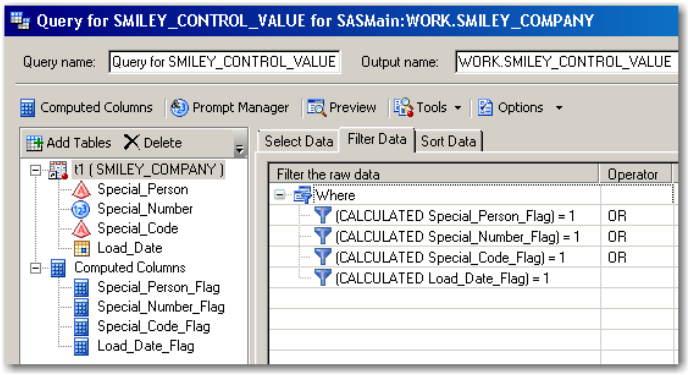
Special_Code_Flag:

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

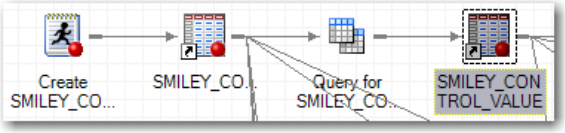
Load_Date_Flag:

```
CASE
    WHEN t1.Load_Date = . THEN 1
    ELSE 0
END
```

The output is filtered to include only rows where a flag is set to 1:



The output table contains 1 row:

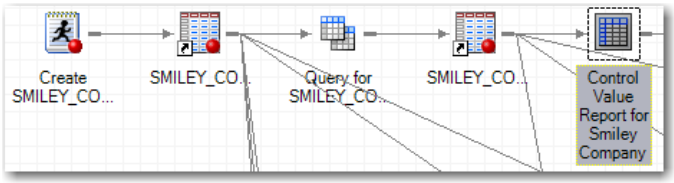


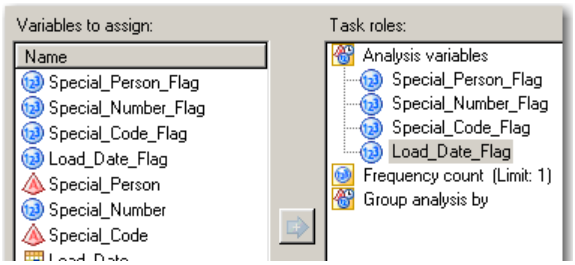
	Special_Person_Flag	Special_Number_Flag	Special_Code_Flag	Load_Date_Flag
1	0	0	1	0

Special_Person	Special_Number	Special_Code	Load_Date
Smiley	10127911		02JAN2015

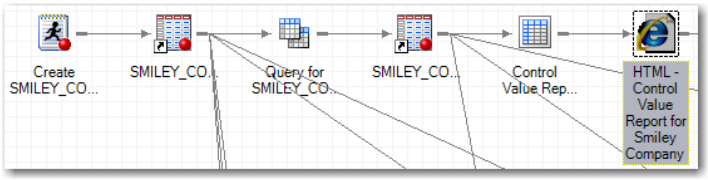
- **The Special_Code_Flag is set to 1 because the Special_Code is missing from this row.**

A One-Way Frequency is run using the 4 flags:





Here is the One-Way Frequency output with the 4 flags:



Control Value Report for Smiley Company

The FREQ Procedure

Special_Person_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Number_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Code_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

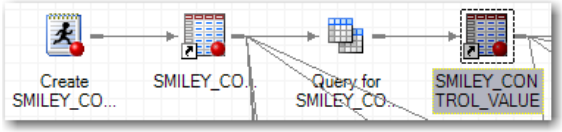
Load_Date_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	1	100.00	1	100.00

- **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…

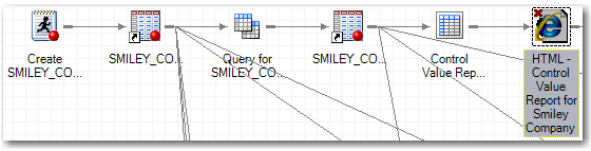
- **To replicate this scenario you will need to perform the following:**
  - **Replace the `Smiley,10127911, ,20090` DATALINE with `Smiley,10127911,A,20090` in the SMILEY_COMPANY Program Node on Page 6 and rerun to have no missing data in the table.**
  - **Rerun the Query for the SMILEY_CONTROL_VALUE table and the Control Value Report One-Way Frequency.**

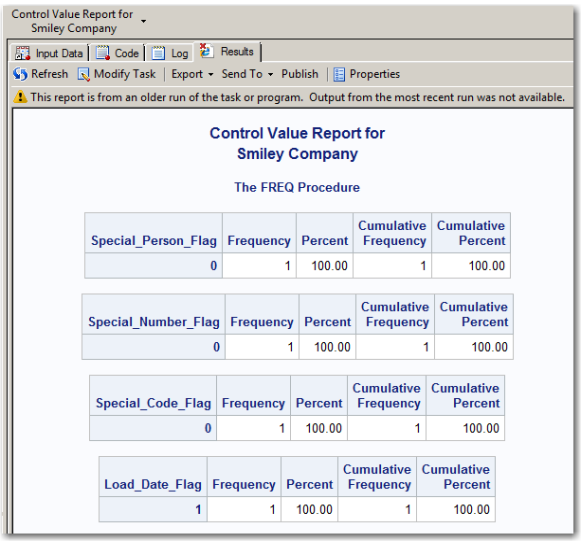
Here is the empty SMILEY_CONTROL_VALUE table:



- **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:





Special_Person_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Number_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Code_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Load_Date_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	1	100.00	1	100.00

- **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:**

- **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.**

When the Smiley_Company table processed error free and no data was missing for the first time, it was ironic 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…

Necessity is the mother of all inventions.

Plato / Einstein

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.**

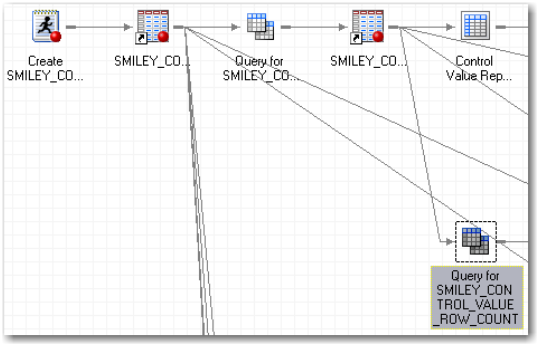
Here is the solution which arose during the quest to resolve this issue:

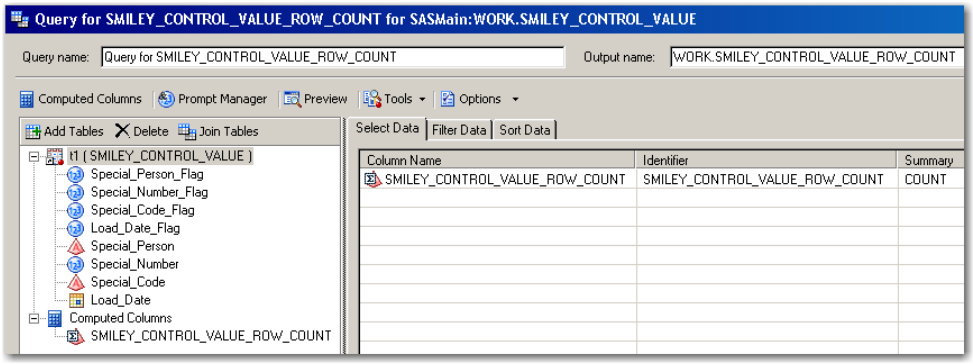
- Create a Smiley_Control_Value_Row_Count table with the row count of the Smiley_Control_Value table.
- Create a Smiley_Control_Value_Mock_Row table based upon an indirect relationship between the Smiley_Control_Value_Row_Count table and the Smiley_Company table.
- When the Smiley_Control_Value table 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.

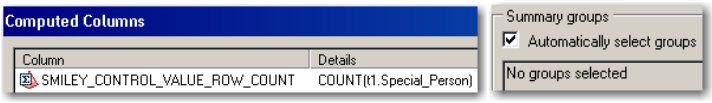
Always Remember, It's Too Soon To Quit!

Bob Wieland (Mr. Inspiration)

This Query creates the SMILEY_CONTROL_VALUE_ROW_COUNT table with the row count of the SMILEY_CONTROL_VALUE table:

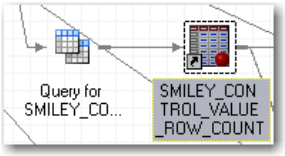






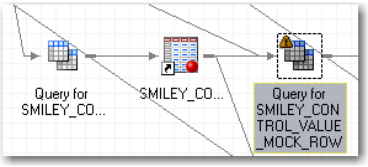
- **A Count of Special_Person is used to create the SMILEY_CONTROL_VALUE_ROW_COUNT.**
- **Automatically Select Groups is selected and no groups are selected to count the rows.**

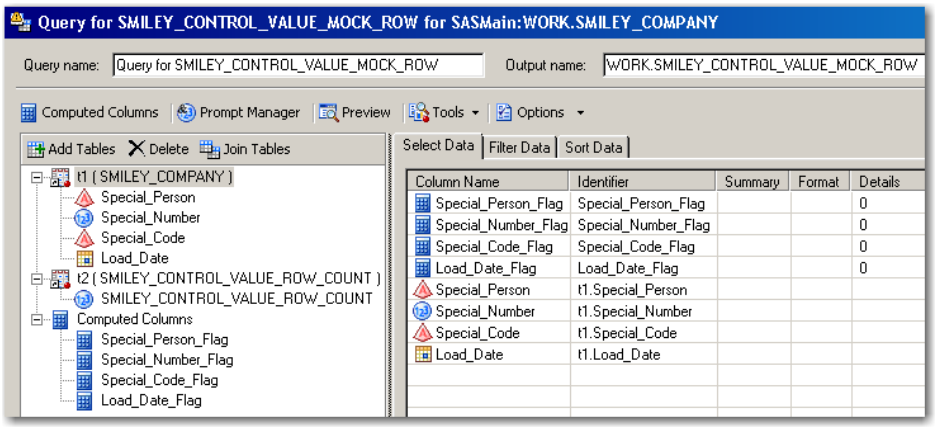
The output table contains 1 row with 1 column:

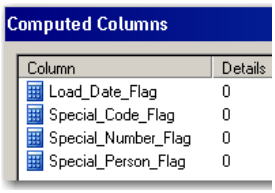


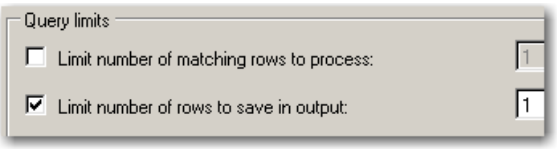
	SMILEY_CONTROL_VALUE_ROW_COUNT
1	0

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:

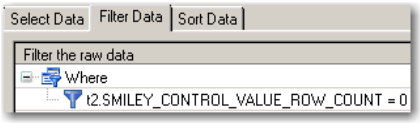






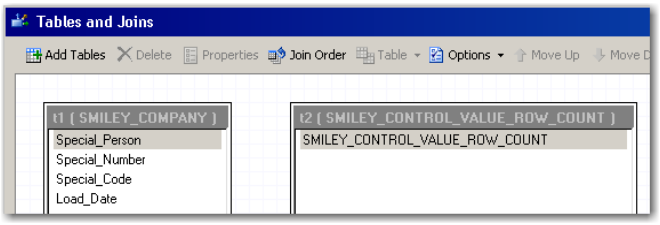


- **As the mock row is created, all 4 flags are set to a 0 value meaning nothing is missing.**
- **Since only 1 mock row is needed, Query limits are set to create 1 output row via the Options.**



- **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:

SMILEY_COMPANY

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Exp

	Special_Person	Special_Number	Special_Code	Load_Date
1	Smiley	10127911	A	02JAN2015
2	Smiley's Son	10173341	K	02JAN2015
3	Smiley's Twin	10376606	B	02JAN2015
4	Smiley's Wife	10927911	A	02JAN2015
5	Smiley's Son	11471884	E	02JAN2015
6	Smiley's Twin	11573691	G	02JAN2015
7	Smiley's Daughter	11975386	C	02JAN2015
8	Smiley's Son	12071884	J	02JAN2015
9	Smiley's Son	12871884	D	02JAN2015
10	Smiley's Twin	13173691	A	02JAN2015
11	Smiley's Wife	13771202	D	02JAN2015
12	Smiley's Daughter	13775498	H	02JAN2015
13	Smiley's Son	14171884	I	02JAN2015
14	Smiley's Twin	15373691	F	02JAN2015
15	Smiley's Son	15471884	C	02JAN2015
16	Smiley's Son	16074330	H	02JAN2015
17	Smiley's Daughter	16175498	B	02JAN2015
18	Smiley's Wife	16176964	I	31DEC2014
19	Smiley	16279111	E	02JAN2015
20	Smiley's Twin	16573691	K	02JAN2015

SMILEY_CONTROL_VALUE_ROW_COUNT

Filter and Sort | Query Builder | Data | Describe

	SMILEY_CONTROL_VALUE_ROW_COUNT
1	0

- **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 of SMILEY_COMPANY and SMILEY_CONTROL_VALUE_ROW_COUNT

Input Data | Code | Log | Output Data

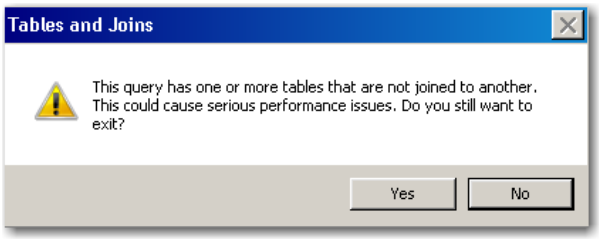
Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Special_Person	Special_Number	Special_Code	Load_Date	SMILEY_CONTROL_VALUE_ROW_COUNT
1	Smiley	10127911	A	02JAN2015	0
2	Smiley's Son	10173341	K	02JAN2015	0
3	Smiley's Twin	10376606	B	02JAN2015	0
4	Smiley's Wife	10927911	A	02JAN2015	0
5	Smiley's Son	11471884	E	02JAN2015	0
6	Smiley's Twin	11573691	G	02JAN2015	0
7	Smiley's Daughter	11975386	C	02JAN2015	0
8	Smiley's Son	12071884	J	02JAN2015	0
9	Smiley's Son	12871884	D	02JAN2015	0
10	Smiley's Twin	13173691	A	02JAN2015	0
11	Smiley's Wife	13771202	D	02JAN2015	0
12	Smiley's Daughter	13775498	H	02JAN2015	0
13	Smiley's Son	14171884	I	02JAN2015	0
14	Smiley's Twin	15373691	F	02JAN2015	0
15	Smiley's Son	15471884	C	02JAN2015	0
16	Smiley's Son	16074330	H	02JAN2015	0
17	Smiley's Daughter	16175498	B	02JAN2015	0
18	Smiley's Wife	16176964	I	31DEC2014	0
19	Smiley	16279111	E	02JAN2015	0
20	Smiley's Twin	16573691	K	02JAN2015	0

- **The Joinless Join automatically creates a Cartesian Product which places the 1 row and 1 column of the SMILEY_CONTROL_VALUE_ROW_COUNT table to the right of each of the 20 rows and 4 columns 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.

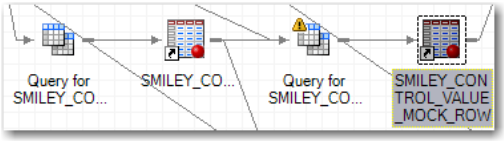


- **This Warning Message always appears whenever 2 tables are joined with a Joinless Join because SAS knows it will create a Cartesian Product which can take a lot of extra resources.**

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:

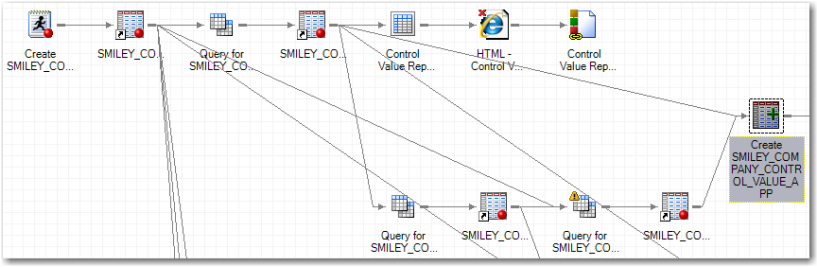


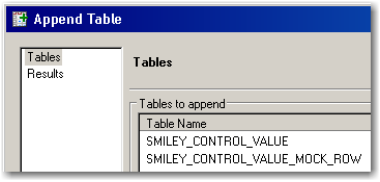
	Special_Person_Flag	Special_Number_Flag	Special_Code_Flag	Load_Date_Flag
1	0	0	0	0

Special_Person	Special_Number	Special_Code	Load_Date
Smiley	10127911	A	02JAN2015

- **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:



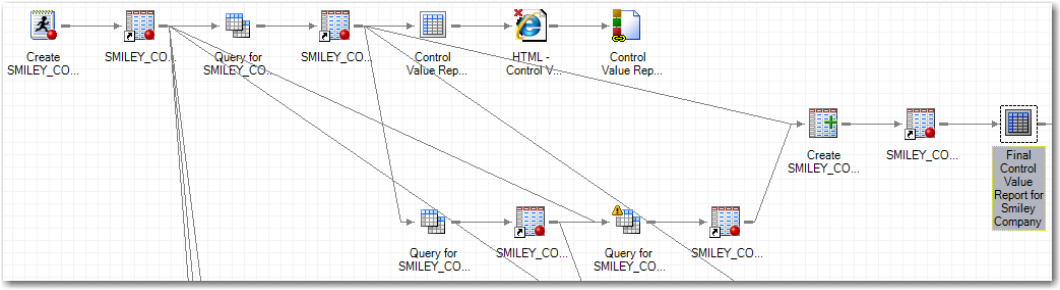


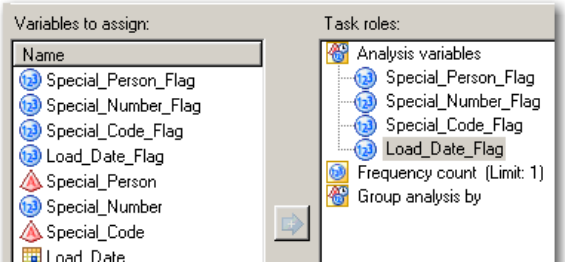
	Special_Person_Flag	Special_Number_Flag	Special_Code_Flag	Load_Date_Flag
1	0	0	0	0

Special_Person	Special_Number	Special_Code	Load_Date
Smiley	10127911	A	02JAN2015

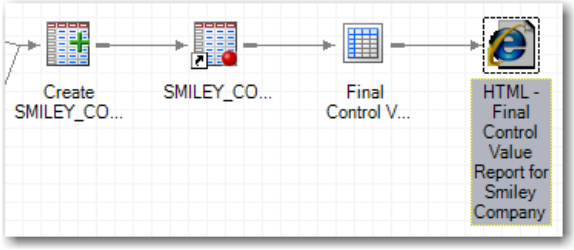
- **Notice the Append result matches the Smiley_Control_Value_Mock_Row table – Done & Done!**
- **We have achieved our desired results and we have a new input to the One-Way Frequency.**

The One-Way Frequency is recreated using the appended table:





Here is the One-Way Frequency output with the 4 flags:



Final Control Value Report for Smiley Company

The FREQ Procedure

Special_Person_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Number_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Special_Code_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

Load_Date_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	1	100.00	1	100.00

- **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,

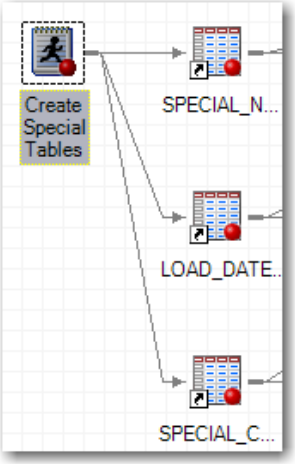
Release the balloons!

Applause… Applause… Applause…

Bring out the treats for everyone!

☺ Oh but wait… your new friend, the Joinless Join, is just getting started! ☺

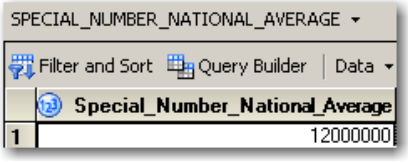
Next we design another Program Node to create 3 additional tables:

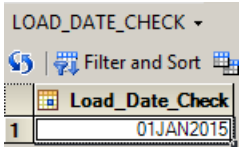


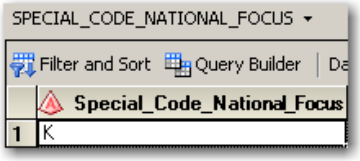
```
DATA Special_Number_National_Average
          (KEEP=Special_Number_National_Average)

      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 = '01JAN2015'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:

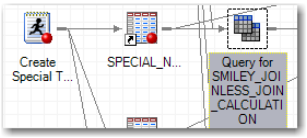


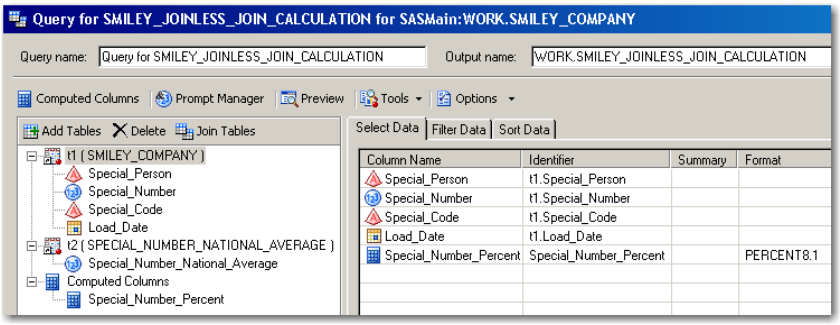




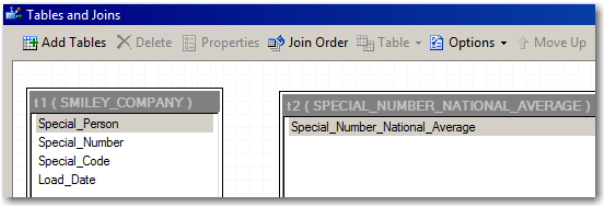
- **The Special_Number_National_Average table contains the average of all the Special_Number columns from each Smiley Company nationwide which we will use in a Joinless Join to calculate a percentage of the Special_Number column in our SMILEY_COMPANY table.**
- **The Load_Date_Check table contains a Load Date which we will use in a Joinless Join to validate that all of our SMILEY_COMPANY table rows were created in 2015.**
- **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 the 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 which indirectly relates to the SMILEY_COMPANY table because it contains the average of all the Special_Number columns from each SMILEY_COMPANY table nationwide.**

The Cartesian product of SMILEY_COMPANY and SPECIAL_NUMBER_NATIONAL_AVERAGE

Input Data (2) | Code | Log | Output Data

Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Special_Person	Special_Number	Special_Code	Load_Date	Special_Number_National_Average
1	Smiley	10127911	A	02JAN2015	12000000
2	Smiley's Son	10173341	K	02JAN2015	12000000
3	Smiley's Twin	10376606	B	02JAN2015	12000000
4	Smiley's Wife	10927911	A	02JAN2015	12000000
5	Smiley's Son	11471884	E	02JAN2015	12000000
6	Smiley's Twin	11573691	G	02JAN2015	12000000
7	Smiley's Daughter	11975386	C	02JAN2015	12000000
8	Smiley's Son	12071884	J	02JAN2015	12000000
9	Smiley's Son	12871884	D	02JAN2015	12000000
10	Smiley's Twin	13173691	A	02JAN2015	12000000
11	Smiley's Wife	13771202	D	02JAN2015	12000000
12	Smiley's Daughter	13775498	H	02JAN2015	12000000
13	Smiley's Son	14171884	I	02JAN2015	12000000
14	Smiley's Twin	15373691	F	02JAN2015	12000000
15	Smiley's Son	15471884	C	02JAN2015	12000000
16	Smiley's Son	16074330	H	02JAN2015	12000000
17	Smiley's Daughter	16175498	B	02JAN2015	12000000
18	Smiley's Wife	16176964	I	31DEC2014	12000000
19	Smiley	16279111	E	02JAN2015	12000000
20	Smiley's Twin	16573691	K	02JAN2015	12000000

- **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 the SMILEY_COMPANY table.**

Computed Columns

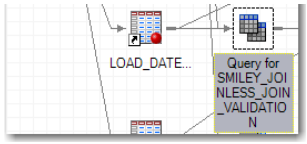
Column	Details
Special_Number_Percent	t1.Special_Number/t2.Special_Number_National_Average

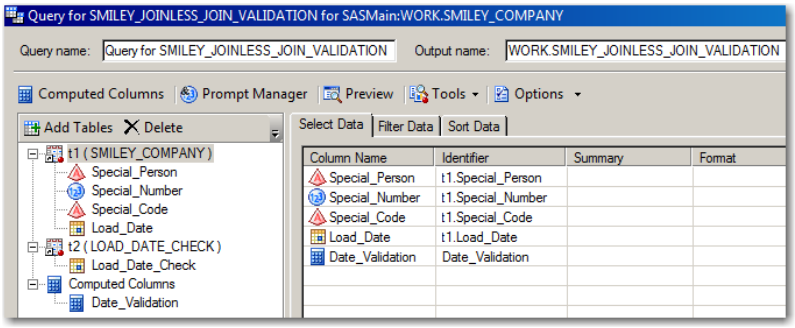
- **Calculate a Special_Number_Percent Computed Column using the Special_Number column from the SMILEY_COMPANY table and the Special_Number_National_Average column from the Cartesian Product results.**

	Special_Person	Special_Number	Special_Code	Load_Date	Special_Number_Percent
1	Smiley	10127911	A	02JAN2015	84.4%
2	Smiley's Son	10173341	K	02JAN2015	84.8%
3	Smiley's Twin	10376606	B	02JAN2015	86.5%
4	Smiley's Wife	10927911	A	02JAN2015	91.1%
5	Smiley's Son	11471884	E	02JAN2015	95.6%
6	Smiley's Twin	11573691	G	02JAN2015	96.4%
7	Smiley's Daughter	11975386	C	02JAN2015	99.8%
8	Smiley's Son	12071884	J	02JAN2015	100.6%
9	Smiley's Son	12871884	D	02JAN2015	107.3%
10	Smiley's Twin	13173691	A	02JAN2015	109.8%
11	Smiley's Wife	13771202	D	02JAN2015	114.8%
12	Smiley's Daughter	13775498	H	02JAN2015	114.8%
13	Smiley's Son	14171884	I	02JAN2015	118.1%
14	Smiley's Twin	15373691	F	02JAN2015	128.1%
15	Smiley's Son	15471884	C	02JAN2015	128.9%
16	Smiley's Son	16074330	H	02JAN2015	134.0%
17	Smiley's Daughter	16175498	B	02JAN2015	134.8%
18	Smiley's Wife	16176964	I	31DEC2014	134.8%
19	Smiley	16279111	E	02JAN2015	135.7%
20	Smiley's Twin	16573691	K	02JAN2015	138.1%

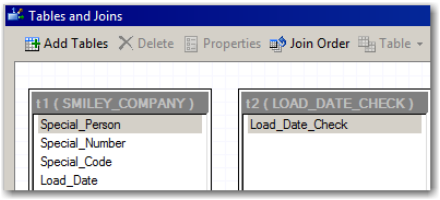
- **Here is the final result of the SMILEY_COMPANY table with the Special_Number_Percent column to the right of each of the 20 rows and 4 columns.**

Designing a Joinless Join to perform a Validation:





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



- **The Joinless Join is based upon the LOAD_DATE_CHECK table which indirectly relates to the 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 Cartesian product of SMILEY_COMPANY and LOAD_DATE_CHECK

Input Data (2) | Code | Log | Output Data

Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Exp

	Special_Person	Special_Number	Special_Code	Load_Date	Load_Date_Check
1	Smiley	10127911	A	02JAN2015	01JAN2015
2	Smiley's Son	10173341	K	02JAN2015	01JAN2015
3	Smiley's Twin	10376606	B	02JAN2015	01JAN2015
4	Smiley's Wife	10927911	A	02JAN2015	01JAN2015
5	Smiley's Son	11471884	E	02JAN2015	01JAN2015
6	Smiley's Twin	11573691	G	02JAN2015	01JAN2015
7	Smiley's Daughter	11975386	C	02JAN2015	01JAN2015
8	Smiley's Son	12071884	J	02JAN2015	01JAN2015
9	Smiley's Son	12871884	D	02JAN2015	01JAN2015
10	Smiley's Twin	13173691	A	02JAN2015	01JAN2015
11	Smiley's Wife	13771202	D	02JAN2015	01JAN2015
12	Smiley's Daughter	13775498	H	02JAN2015	01JAN2015
13	Smiley's Son	14171884	I	02JAN2015	01JAN2015
14	Smiley's Twin	15373691	F	02JAN2015	01JAN2015
15	Smiley's Son	15471884	C	02JAN2015	01JAN2015
16	Smiley's Son	16074330	H	02JAN2015	01JAN2015
17	Smiley's Daughter	16175498	B	02JAN2015	01JAN2015
18	Smiley's Wife	16176964	I	31DEC2014	01JAN2015
19	Smiley	16279111	E	02JAN2015	01JAN2015
20	Smiley's Twin	16573691	K	02JAN2015	01JAN2015

- **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 the SMILEY_COMPANY table.**

Computed Columns

Column	Details
Date_Validation	case when t1.Load_Date ge t2.Load_Date_Check then 'AOK' else 'NOT_AOK' end

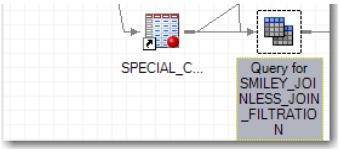
```
case
   when t1.Load_Date ge t2.Load_Date_Check
      then 'AOK'
   else 'NOT_AOK'
end
```

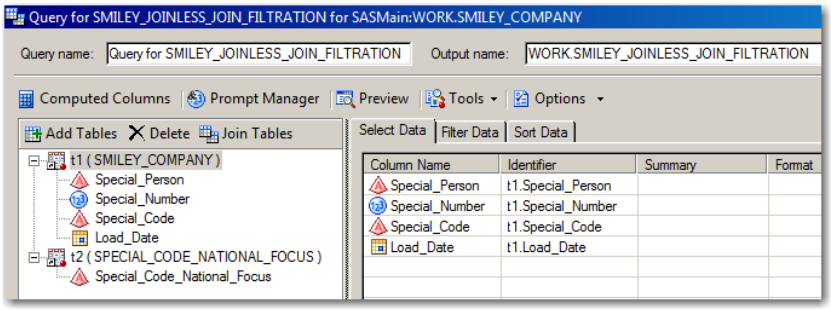
- **Validate a Date_Validation Computed Column using the Load_Date column from the SMILEY_COMPANY table and the Load_Date_Check column from the Cartesian Product results.**

	Special_Person	Special_Number	Special_Code	Load_Date	Date_Validation
1	Smiley	10127911	A	02JAN2015	AOK
2	Smiley's Son	10173341	K	02JAN2015	AOK
3	Smiley's Twin	10376606	B	02JAN2015	AOK
4	Smiley's Wife	10927911	A	02JAN2015	AOK
5	Smiley's Son	11471884	E	02JAN2015	AOK
6	Smiley's Twin	11573691	G	02JAN2015	AOK
7	Smiley's Daughter	11975386	C	02JAN2015	AOK
8	Smiley's Son	12071884	J	02JAN2015	AOK
9	Smiley's Son	12871884	D	02JAN2015	AOK
10	Smiley's Twin	13173691	A	02JAN2015	AOK
11	Smiley's Wife	13771202	D	02JAN2015	AOK
12	Smiley's Daughter	13775498	H	02JAN2015	AOK
13	Smiley's Son	14171884	I	02JAN2015	AOK
14	Smiley's Twin	15373691	F	02JAN2015	AOK
15	Smiley's Son	15471884	C	02JAN2015	AOK
16	Smiley's Son	16074330	H	02JAN2015	AOK
17	Smiley's Daughter	16175498	B	02JAN2015	AOK
18	Smiley's Wife	16176964	I	31DEC2014	NOT_AOK
19	Smiley	16279111	E	02JAN2015	AOK
20	Smiley's Twin	16573691	K	02JAN2015	AOK

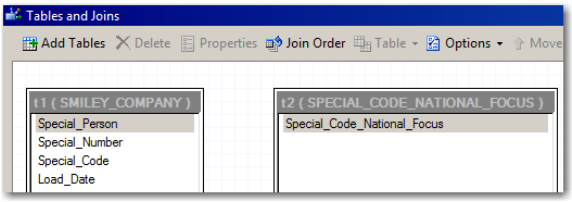
- **Here is the final result of the SMILEY_COMPANY table with the Special_Number_Percent column to the right of each of the 20 rows and 4 columns.**

Designing a Joinless Join to perform a Filtration:

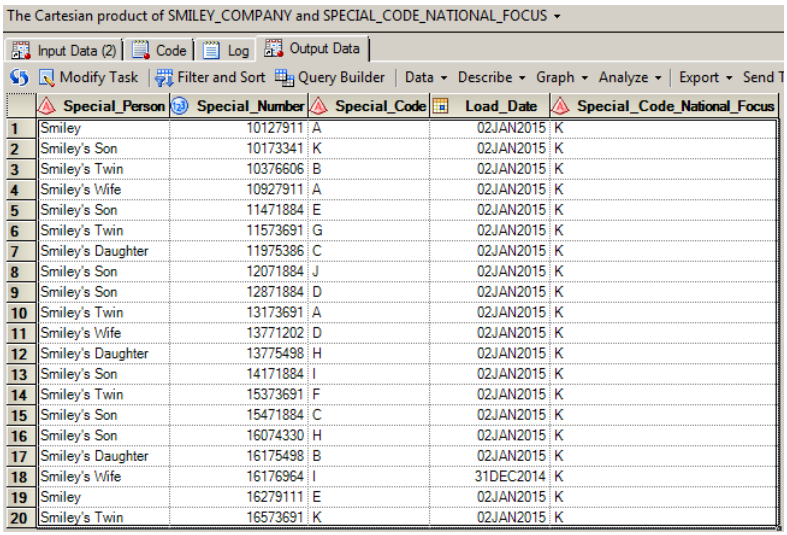




- **Build a Query with the SMILEY_COMPANY table and the Smiley Company National Headquarters SPECIAL_CODE_NATIONAL_FOCUS table.**

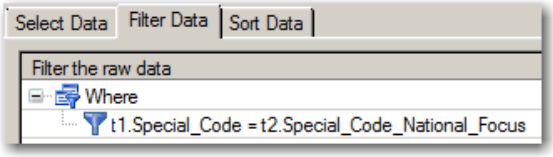


- **The Joinless Join is based upon the SPECIAL_CODE_NATIONAL_FOCUS table which indirectly relates to the SMILEY_COMPANY table because it contains the Special Code to be focused upon nationwide within the Special_Code column in the SMILEY_COMPANY table.**



	Special_Person	Special_Number	Special_Code	Load_Date	Special_Code_National_Focus
1	Smiley	10127911	A	02JAN2015	K
2	Smiley's Son	10173341	K	02JAN2015	K
3	Smiley's Twin	10376606	B	02JAN2015	K
4	Smiley's Wife	10927911	A	02JAN2015	K
5	Smiley's Son	11471884	E	02JAN2015	K
6	Smiley's Twin	11573691	G	02JAN2015	K
7	Smiley's Daughter	11975386	C	02JAN2015	K
8	Smiley's Son	12071884	J	02JAN2015	K
9	Smiley's Son	12871884	D	02JAN2015	K
10	Smiley's Twin	13173691	A	02JAN2015	K
11	Smiley's Wife	13771202	D	02JAN2015	K
12	Smiley's Daughter	13775498	H	02JAN2015	K
13	Smiley's Son	14171884	I	02JAN2015	K
14	Smiley's Twin	15373691	F	02JAN2015	K
15	Smiley's Son	15471884	C	02JAN2015	K
16	Smiley's Son	16074330	H	02JAN2015	K
17	Smiley's Daughter	16175498	B	02JAN2015	K
18	Smiley's Wife	16176964	I	31DEC2014	K
19	Smiley	16279111	E	02JAN2015	K
20	Smiley's Twin	16573691	K	02JAN2015	K

- **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 the SMILEY_COMPANY table.**

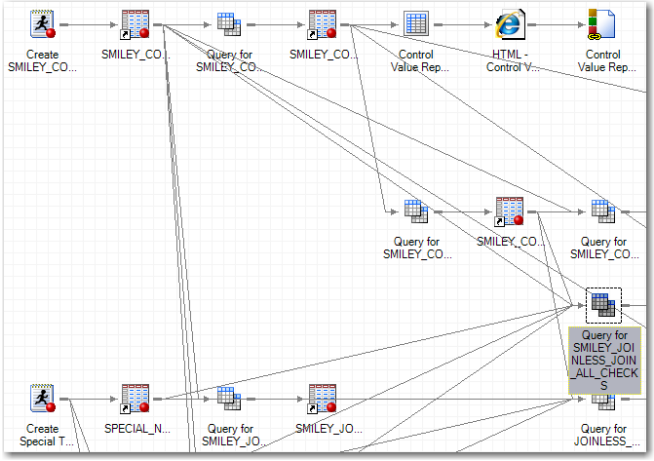


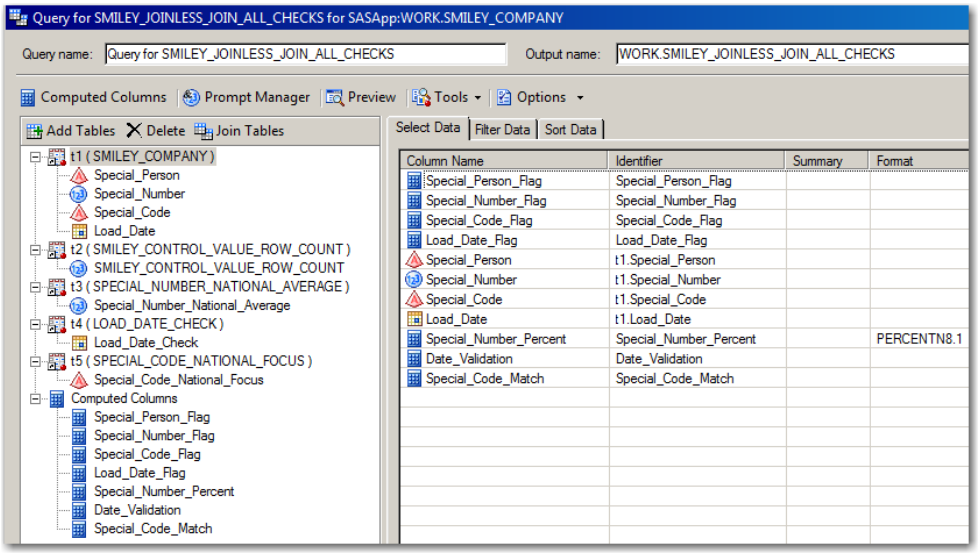
- **Filter the raw data to include the rows where the value of the Special_Code column from the SMILEY_COMPANY table is equal to the value of the Special_Code_National_Focus column from the Cartesian Product results.**

	Special_Person	Special_Number	Special_Code	Load_Date
1	Smiley's Son	10173341	K	02JAN2015
2	Smiley's Twin	16573691	K	02JAN2015

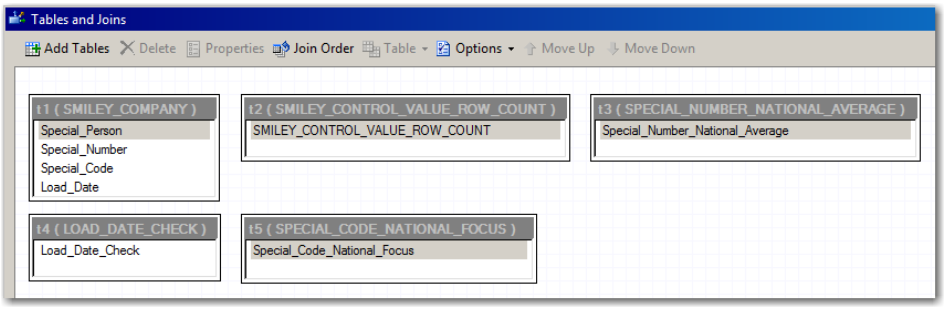
- **Here is the final result of the SMILEY_COMPANY table with the Special_Code column filtered by the Special_Code_National_Focus column.**

Designing a Joinless Join to perform a Mock Row Creation, Calculation, Validation, and Filtration:





- **Build a Query with the 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 which indirectly relate to the SMILEY_COMPANY table as shown in the previous examples.**

The Cartesian product of SMILEY_COMPANY and 4 Tables with 1 Row and 1 Column in each Table

Input Data (5) | Code | Log | Output Data

Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Special_Person	Special_Number	Special_Code	Load_Date	SMILEY_CONTROL_VALUE_ROW_COUNT	Special_Number_National_Average	Load_Date_Check	Special_Code_National_Focus
1	Smiley	10127911	A	02JAN2015	0	12000000	01JAN2015	K
2	Smiley's Son	10173341	K	02JAN2015	0	12000000	01JAN2015	K
3	Smiley's Twin	10376606	B	02JAN2015	0	12000000	01JAN2015	K
4	Smiley's Wife	10927911	A	02JAN2015	0	12000000	01JAN2015	K
5	Smiley's Son	11471884	E	02JAN2015	0	12000000	01JAN2015	K
6	Smiley's Twin	11573691	G	02JAN2015	0	12000000	01JAN2015	K
7	Smiley's Daughter	11975386	C	02JAN2015	0	12000000	01JAN2015	K
8	Smiley's Son	12071884	J	02JAN2015	0	12000000	01JAN2015	K
9	Smiley's Son	12871884	D	02JAN2015	0	12000000	01JAN2015	K
10	Smiley's Twin	13173691	A	02JAN2015	0	12000000	01JAN2015	K
11	Smiley's Wife	13771202	D	02JAN2015	0	12000000	01JAN2015	K
12	Smiley's Daughter	13775498	H	02JAN2015	0	12000000	01JAN2015	K
13	Smiley's Son	14171884	I	02JAN2015	0	12000000	01JAN2015	K
14	Smiley's Twin	15373691	F	02JAN2015	0	12000000	01JAN2015	K
15	Smiley's Son	15471884	C	02JAN2015	0	12000000	01JAN2015	K
16	Smiley's Son	16074330	H	02JAN2015	0	12000000	01JAN2015	K
17	Smiley's Daughter	16175498	B	02JAN2015	0	12000000	01JAN2015	K
18	Smiley's Wife	16176964	I	31DEC2014	0	12000000	01JAN2015	K
19	Smiley	16279111	E	02JAN2015	0	12000000	01JAN2015	K
20	Smiley's Twin	16573691	K	02JAN2015	0	12000000	01JAN2015	K

- **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 the SMILEY_COMPANY table.**

Computed Columns

Column	Details
Date_Validation	case when t1.Load_Date ge t4.Load_Date_Check then 'AOK' else 'NOT_AOK' end
Load_Date_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Code_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Code_Match	case when t1.Special_Code = t5.Special_Code_National_Focus then 'MATCH' else 'NO MATCH' end
Special_Number_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Number_Percent	t1.Special_Number/t3.Special_Number_National_Average
Special_Person_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end

- **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.**

	Special_Person_Flag	Special_Number_Flag	Special_Code_Flag	Load_Date_Flag	Special_Person	Special_Number	Special_Code	Load_Date	Special_Number_Percent	Date_Validation	Special_Code_Match
1	0	0	0	0	Smiley	10127911	A	02JAN2015	84.4%	AOK	NO MATCH
2	0	0	0	0	Smiley's Son	10173341	K	02JAN2015	84.8%	AOK	MATCH
3	0	0	0	0	Smiley's Twin	10376606	B	02JAN2015	86.5%	AOK	NO MATCH
4	0	0	0	0	Smiley's Wife	10927911	A	02JAN2015	91.1%	AOK	NO MATCH
5	0	0	0	0	Smiley's Son	11471884	E	02JAN2015	95.6%	AOK	NO MATCH
6	0	0	0	0	Smiley's Twin	11573691	G	02JAN2015	96.4%	AOK	NO MATCH
7	0	0	0	0	Smiley's Daughter	11975386	C	02JAN2015	99.8%	AOK	NO MATCH
8	0	0	0	0	Smiley's Son	12071884	J	02JAN2015	100.6%	AOK	NO MATCH
9	0	0	0	0	Smiley's Son	12871884	D	02JAN2015	107.3%	AOK	NO MATCH
10	0	0	0	0	Smiley's Twin	13173691	A	02JAN2015	109.8%	AOK	NO MATCH
11	0	0	0	0	Smiley's Wife	13771202	D	02JAN2015	114.8%	AOK	NO MATCH
12	0	0	0	0	Smiley's Daughter	13775498	H	02JAN2015	114.8%	AOK	NO MATCH
13	0	0	0	0	Smiley's Son	14171884	I	02JAN2015	118.1%	AOK	NO MATCH
14	0	0	0	0	Smiley's Twin	15373691	F	02JAN2015	128.1%	AOK	NO MATCH
15	0	0	0	0	Smiley's Son	15471884	C	02JAN2015	128.9%	AOK	NO MATCH
16	0	0	0	0	Smiley's Son	16074330	H	02JAN2015	134.0%	AOK	NO MATCH
17	0	0	0	0	Smiley's Daughter	16175498	B	02JAN2015	134.8%	AOK	NO MATCH
18	0	0	0	0	Smiley's Wife	16176964	I	31DEC2014	134.8%	NOT_AOK	NO MATCH
19	0	0	0	0	Smiley	16279111	E	02JAN2015	135.7%	AOK	NO MATCH
20	0	0	0	0	Smiley's Twin	16573691	K	02JAN2015	138.1%	AOK	MATCH

- **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.**

Control Value Report for
Smiley Company All Joinless Joins

The FREQ Procedure

Special_Person_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

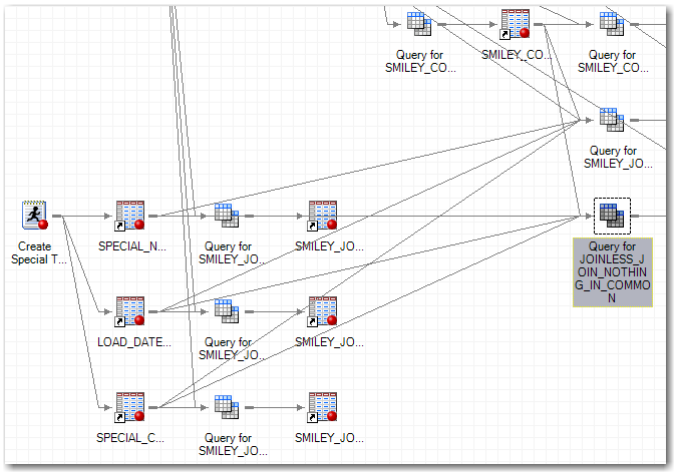
Special_Number_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

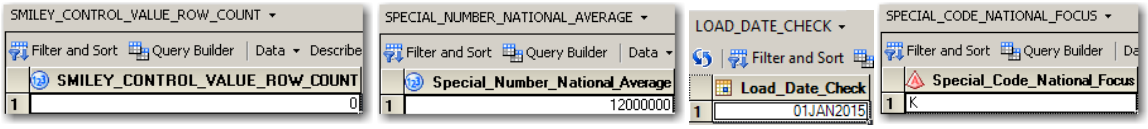
Special_Code_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

Load_Date_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

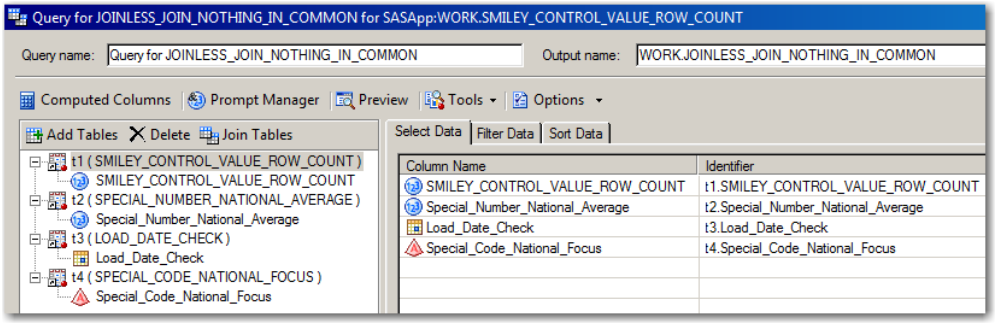
- **The One-Way Frequency correctly displays that all 4 flags are set to 0 and therefore no data is missing ~ thanks to the Joinless Join ☺.**

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:

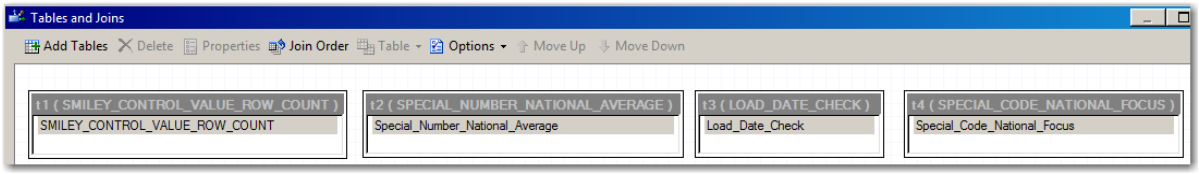




- **Notice how the 4 columns in the 4 tables have No Relationships At All.**



- **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 Cartesian product of 4 Tables with 1 Row and 1 Column in each Table

Input Data (4) | Code | Log | Output Data

Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	SMILEY_CONTROL_VALUE_ROW_COUNT	Special_Number_National_Average	Load_Date_Check	Special_Code_National_Focus
1	0	12000000	01JAN2015	K

- **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.**

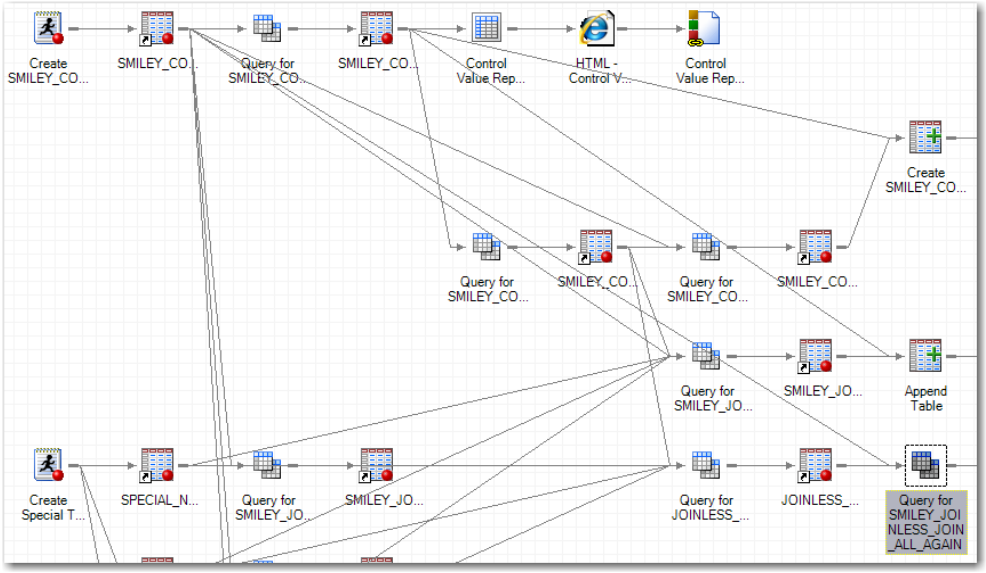
JOINLESS_JOIN_NOTHING_IN_COMMON

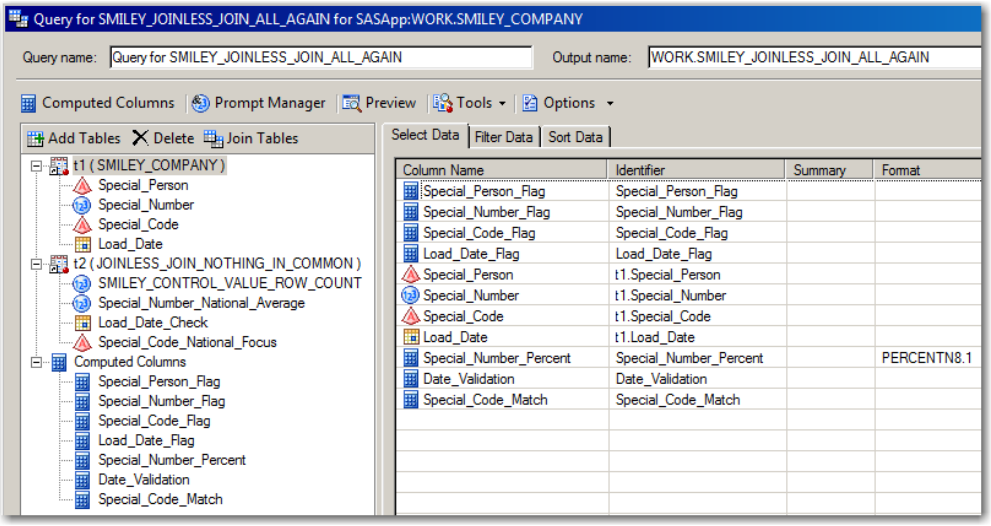
Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	SMILEY_CONTROL_VALUE_ROW_COUNT	Special_Number_National_Average	Load_Date_Check	Special_Code_National_Focus
1	0	12000000	01JAN2015	K

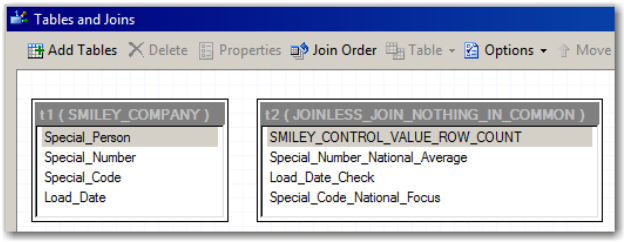
- **Here is the final result from selecting all 4 columns which is equal to the Cartesian Product.**

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 the 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 which indirectly relate to the SMILEY_COMPANY table as shown in the previous examples.**

The Cartesian product of SMILEY_COMPANY and 1 Table with 1 Row and 4 Columns

Input Data (2) | Code | Log | Output Data

Modify Task | Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	Special_Person	Special_Number	Special_Code	Load_Date	SMILEY_CONTROL_VALUE_ROW_COUNT	Special_Number_National_Average	Load_Date_Check	Special_Code_National_Focus
1	Smiley	10127911	A	02JAN2015	0	12000000	01JAN2015	K
2	Smiley's Son	10173341	K	02JAN2015	0	12000000	01JAN2015	K
3	Smiley's Twin	10376606	B	02JAN2015	0	12000000	01JAN2015	K
4	Smiley's Wife	10927911	A	02JAN2015	0	12000000	01JAN2015	K
5	Smiley's Son	11471884	E	02JAN2015	0	12000000	01JAN2015	K
6	Smiley's Twin	11573691	G	02JAN2015	0	12000000	01JAN2015	K
7	Smiley's Daughter	11975386	C	02JAN2015	0	12000000	01JAN2015	K
8	Smiley's Son	12071884	J	02JAN2015	0	12000000	01JAN2015	K
9	Smiley's Son	12871884	D	02JAN2015	0	12000000	01JAN2015	K
10	Smiley's Twin	13173691	A	02JAN2015	0	12000000	01JAN2015	K
11	Smiley's Wife	13771202	D	02JAN2015	0	12000000	01JAN2015	K
12	Smiley's Daughter	13775498	H	02JAN2015	0	12000000	01JAN2015	K
13	Smiley's Son	14171884	I	02JAN2015	0	12000000	01JAN2015	K
14	Smiley's Twin	15373691	F	02JAN2015	0	12000000	01JAN2015	K
15	Smiley's Son	15471884	C	02JAN2015	0	12000000	01JAN2015	K
16	Smiley's Son	16074330	H	02JAN2015	0	12000000	01JAN2015	K
17	Smiley's Daughter	16175498	B	02JAN2015	0	12000000	01JAN2015	K
18	Smiley's Wife	16176964	I	31DEC2014	0	12000000	01JAN2015	K
19	Smiley	16279111	E	02JAN2015	0	12000000	01JAN2015	K
20	Smiley's Twin	16573691	K	02JAN2015	0	12000000	01JAN2015	K

- **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 the SMILEY_COMPANY table.**

Computed Columns

Column	Details
Date_Validation	case when t1.Load_Date ge t2.Load_Date_Check then 'AOK' else 'NOT_AOK' end
Load_Date_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Code_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Code_Match	case when t1.Special_Code = t2.Special_Code_National_Focus then 'MATCH' else 'NO MATCH' end
Special_Number_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end
Special_Number_Percent	t1.Special_Number/t2.Special_Number_National_Average
Special_Person_Flag	case t2.SMILEY_CONTROL_VALUE_ROW_COUNT when 0 then 0 else . end

- **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.**

	Special_Person_Flag	Special_Number_Flag	Special_Code_Flag	Load_Date_Flag	Special_Person	Special_Number	Special_Code	Load_Date	Special_Number_Percent	Date_Validation	Special_Code_Match
1	0	0	0	0	Smiley	10127911	A	02JAN2015	84.4%	AOK	NO MATCH
2	0	0	0	0	Smiley's Son	10173341	K	02JAN2015	84.8%	AOK	MATCH
3	0	0	0	0	Smiley's Twin	10376606	B	02JAN2015	86.5%	AOK	NO MATCH
4	0	0	0	0	Smiley's Wife	10927911	A	02JAN2015	91.1%	AOK	NO MATCH
5	0	0	0	0	Smiley's Son	11471884	E	02JAN2015	95.6%	AOK	NO MATCH
6	0	0	0	0	Smiley's Twin	11573691	G	02JAN2015	96.4%	AOK	NO MATCH
7	0	0	0	0	Smiley's Daughter	11975386	C	02JAN2015	99.8%	AOK	NO MATCH
8	0	0	0	0	Smiley's Son	12071884	J	02JAN2015	100.6%	AOK	NO MATCH
9	0	0	0	0	Smiley's Son	12871884	D	02JAN2015	107.3%	AOK	NO MATCH
10	0	0	0	0	Smiley's Twin	13173691	A	02JAN2015	109.8%	AOK	NO MATCH
11	0	0	0	0	Smiley's Wife	13771202	D	02JAN2015	114.8%	AOK	NO MATCH
12	0	0	0	0	Smiley's Daughter	13775498	H	02JAN2015	114.8%	AOK	NO MATCH
13	0	0	0	0	Smiley's Son	14171884	I	02JAN2015	118.1%	AOK	NO MATCH
14	0	0	0	0	Smiley's Twin	15373691	F	02JAN2015	128.1%	AOK	NO MATCH
15	0	0	0	0	Smiley's Son	15471884	C	02JAN2015	128.9%	AOK	NO MATCH
16	0	0	0	0	Smiley's Son	16074330	H	02JAN2015	134.0%	AOK	NO MATCH
17	0	0	0	0	Smiley's Daughter	16175498	B	02JAN2015	134.8%	AOK	NO MATCH
18	0	0	0	0	Smiley's Wife	16176964	I	31DEC2014	134.8%	NOT_AOK	NO MATCH
19	0	0	0	0	Smiley	16279111	E	02JAN2015	135.7%	AOK	NO MATCH
20	0	0	0	0	Smiley's Twin	16573691	K	02JAN2015	138.1%	AOK	MATCH

- **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.**

Control Value Report for
Smiley Company All Joinless Joins Again

The FREQ Procedure

Special_Person_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

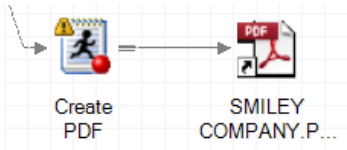
Special_Number_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

Special_Code_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

Load_Date_Flag	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	20	100.00	20	100.00

- **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 ☺.**

Design a Quarterly Report PDF utilizing the results of the Joinless Join of the 4 tables with No Relationships At All:



JOINLESS_JOIN_NOTHING_IN_COMMON

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze | Export | Send To

	SMILEY_CONTROL_VALUE_ROW_COUNT	Special_Number_National_Average	Load_Date_Check	Special_Code_National_Focus
1	0	12000000	01JAN2015	K

- **The Smiley Company has requested a quarterly report of the results of the Joinless Join of the 4 tables with No Relationships At All that we designed on pages 26 and 27.**

```
ODS PDF FILE='/data/MWSUG/JOINLESS_JOIN/SMILEY COMPANY.PDF' NOTOC;
TITLE 'SMILEY COMPANY - QUARTERLY VALIDATION PARAMETERS';
PROC REPORT DATA=JOINLESS_JOIN_NOTHING_IN_COMMON NOWD;
    COLUMNS SMILEY_CONTROL_VALUE_ROW_COUNT Special_Number_National_Average
            Load_Date_Check Special_Code_National_Focus;
    DEFINE SMILEY_CONTROL_VALUE_ROW_COUNT  / STYLE={WIDTH=25mm JUST=CENTER}
        "Missing Values";
    DEFINE Special_Number_National_Average / STYLE={WIDTH=25mm JUST=CENTER}
        "National Average";
    DEFINE Load_Date_Check                 / STYLE={WIDTH=25mm JUST=CENTER}
        "Load Date Check";
    DEFINE Special_Code_National_Focus     / STYLE={WIDTH=25mm JUST=CENTER}
        "Special Code";
RUN;
ODS PDF CLOSE;
```

- **The `ODS PDF FILE` statement opens the `SMILEY COMPANY.PDF` with no table of contents – `NOTOC`.**
- **The `TITLE` statement includes the title shown at the top of the PDF.**
- **`PROC REPORT` is used to report the contents of the `JOINLESS_JOIN_NOTHING_IN_COMMON` table in the PDF with no default report window – `NOWD`.**
- **The `COLUMNS` statement tells `PROC REPORT` which columns to include in the report.**
- **The `DEFINE` statements provide a `WIDTH` and justification along with renaming each column.**
- **The `ODS PDF CLOSE` statement closes the PDF.**

21:33 Monday, October 5, 2015 1

SMILEY COMPANY - QUARTERLY VALIDATION PARAMETERS

Missing Values	National Average	Load Date Check	Special Code
0	12000000	01JAN2015	K

- **Here is the PDF of the results of the Joinless Join of the 4 tables with No Relationships At All.**

CONCLUSION

The **Joinless Join** empowers you to creatively overcome the limits of a standard Join or Merge and enables you to expand the power of Base SAS and SAS Enterprise Guide in a new way. **The Power To Know** 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 tables which do not contain like columns or the same variable name. New worlds of table creations, calculations, validations, filtrations, and PROC REPORTing have opened up to greatly expand your data transformation and analysis toolkit. Begin thinking about how you can benefit from the power and 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.

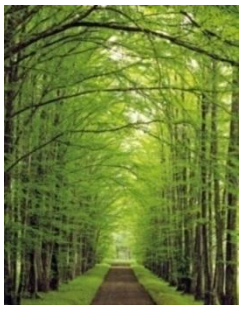
☺ **Roy Rogers Riders Club Rules** ☺

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

Thank You for sharing part of your SAS journey with us ~

☺ **Happy SAS Trails to you… until we meet again** ☺



MEET THE AUTHORS

Writing is a permanent legacy.

John C. Maxwell

Kent Phelps ~ *SAS Certified Professional* **~** *B.S. Electrical Engineering* **~** *Writer* **~** *Teacher* **~** *Coach* **~** has presented at the MWSUG Conference for 3 years, worked in IT and Data Governance since 1990, programmed in SAS since 2007, and specializes in blending the best of Base SAS with SAS Enterprise Guide to engineer automated solutions. He co-created/taught *Intro to SAS EG* classes, offered *SAS News You Can Use*, presented at the Iowa SAS Users Group (IASUG), studied Transformational Leadership, Dynamic Teamwork, and Personal Growth since 1994, and is certified as a *John Maxwell Team* and *48 Days To The Work You Love* Coach. Past highlights include acting for over ten years, co-leading *WOW Drama*, singing a drama solo with a live orchestra, and auditioning in Branson, MO. Kent wants to encourage and equip you to fulfill your life and leadership potential as you build an enduring legacy of inspiration, excellence, and honor.

****

Ronda Phelps ~ *Writer* **~** *Teacher* **~** *Coach* **~** has presented at the MWSUG Conference for 2 years, formerly worked in the Banking and Insurance industries for 19 years, studied Transformational Leadership, Dynamic Teamwork, and Personal Growth since 1994, and is certified as a *John Maxwell Team* and *48 Days To The Work You Love* Coach. Past highlights include speaking in Siberia, acting for over ten years, co-leading *WOW Drama*, and developing life-changing presentations. Ronda believes that YOU are a gift the world is waiting to receive, and she wants to encourage and equip you to pursue your unique destiny as you navigate your life journey with intentionality, fulfilling purpose, and enduring hope.

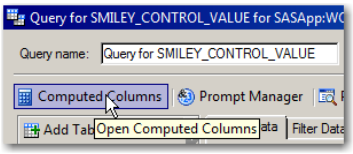
We invite you to share your valued comments with us:

Kent ♥ Ronda Team Phelps
The SASketeers ~ *All for SAS & SAS for All!*
E-mail: SASketeers@q.com

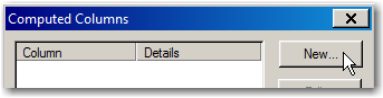
☺ ***We look forward to connecting with you in the future!*** ☺

APPENDIX A
How To Create Computed Columns

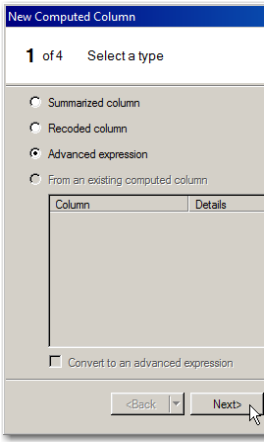
Here is the process to create the 4 Computed Columns in the SMILEY_CONTROL_VALUE table:



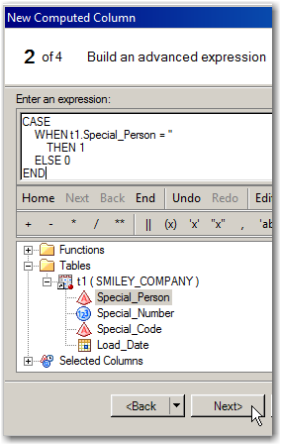
- **From within the Query click Computed Columns to open the list of Computed Columns.**



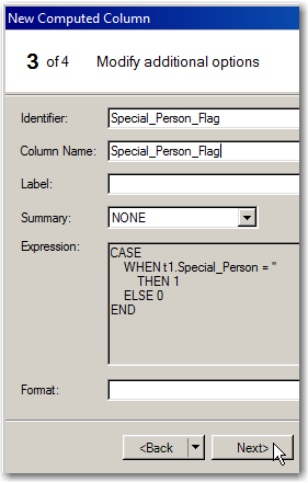
- **Click New to create a New Computed Column.**



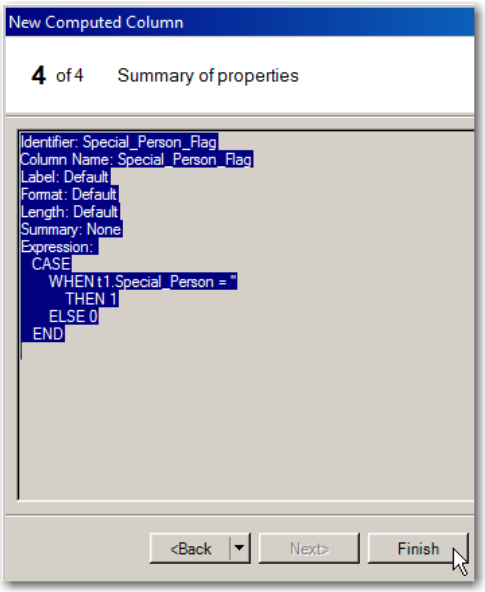
- **To create a flag using a CASE statement, select Advanced expression and click Next.**

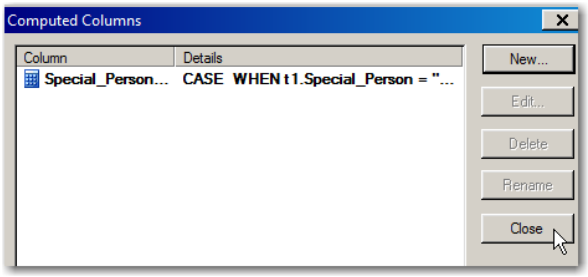


- **Enter the expression while typing or clicking the functions and column names and click Next.**

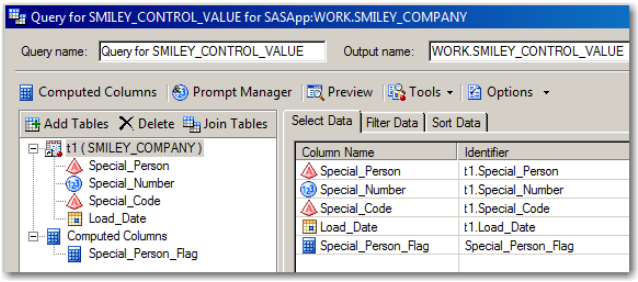


- **Enter the New Computed Column as the Identifier and Column Name and click Next.**





- **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.**

APPENDIX B
How To Code a Joinless Join Using Base SAS

The following Base SAS Code was generated by SAS Enterprise Guide for all examples and creates the same results when copied to and run in Base SAS.

This code creates the SMILEY_CONTROL_VALUE table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_CONTROL_VALUE AS
    SELECT /* Special_Person_Flag */
              (CASE
                  WHEN t1.Special_Person = ''
                      THEN 1
                  ELSE 0
               END) AS Special_Person_Flag,
           /* Special_Number_Flag */
              (CASE
                  WHEN t1.Special_Number = 0
                      THEN 1
                  WHEN t1.Special_Number IS MISSING
                      THEN 1
                  ELSE 0
               END) AS Special_Number_Flag,
           /* Special_Code_Flag */
              (CASE
                  WHEN t1.Special_Code = ''
                      THEN 1
                  ELSE 0
               END) AS Special_Code_Flag,
           /* Load_Date_Flag */
              (CASE
                  WHEN t1.Load_Date = .
                      THEN 1
                  ELSE 0
               END) AS Load_Date_Flag,
           t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date
    FROM WORK.SMILEY_COMPANY t1
    WHERE (CALCULATED Special_Person_Flag) = 1 OR
          (CALCULATED Special_Number_Flag) = 1 OR
          (CALCULATED Special_Code_Flag)   = 1 OR
          (CALCULATED Load_Date_Flag)      = 1;
QUIT;
```

- **The `PROC SQL` creates a `TABLE` called `SMILEY_CONTROL_VALUE` by assigning the value `0` (present) or `1` (missing) to `Special_Person_Flag`, `Special_Number_Flag`, `Special_Code_Flag`, and `Load_Date_Flag` and selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.**
- **Each `CASE` statement ends with `AS` and a variable name because the result of each `CASE` statement is stored in a flag variable.**
- **The `WHERE` clause contains the word `CALCULATED` before each variable name because these variables are calculated rather than selected from the table while also limiting the output data set to contain only rows in which 1 or more of the calculated variables are missing (= `1`).**

This code creates the Control Value Report for the Smiley Company One-Way Frequency:

```
PROC SQL;
    CREATE VIEW WORK.SORT AS
           SELECT T.Special_Person_Flag, T.Special_Number_Flag,
                  T.Special_Code_Flag, T.Load_Date_Flag
    FROM WORK.SMILEY_CONTROL_VALUE AS T;
QUIT;

TITLE;
TITLE1 "Control Value Report for";
TITLE2 "Smiley Company";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System on %TRIM(%QSYSFUNC(DATE(),
           NLDATE20.)) at %TRIM(%SYSFUNC(TIME(), TIMEAMNP12.))";

PROC FREQ DATA=WORK.SORT
       ORDER=INTERNAL;
       TABLES Special_Person_Flag /  SCORES=TABLE;
       TABLES Special_Number_Flag /  SCORES=TABLE;
       TABLES Special_Code_Flag   /  SCORES=TABLE;
       TABLES Load_Date_Flag      /  SCORES=TABLE;
RUN;

RUN; QUIT;
TITLE; FOOTNOTE;
```

- The `PROC SQL` creates a `VIEW` from the `SMILEY_CONTROL_VALUE` table containing only the variables which are to be included in the One-Way Frequency.
- The `TITLE` and `FOOTNOTE` statements with no title or footnote clear all titles and footnotes, and the `TITLE1`, `TITLE2`, and `FOOTNOTE1` statements set the titles and footnote.
- `FOOTNOTE1` is an optional default which is always added by SAS Enterprise Guide.
- The `PROC FREQ` creates `TABLES` for each `Flag` containing the `SCORES` or values of each `Flag` listed by `INTERNAL` or numeric/alphabetic `ORDER`.
- Both `ORDER=INTERNAL` and `SCORES=TABLE` are optional defaults for `PROC FREQ` which are always added by SAS Enterprise Guide.

This code creates the SMILEY_CONTROL_VALUE_ROW_COUNT table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_CONTROL_VALUE_ROW_COUNT AS
    SELECT /* SMILEY_CONTROL_VALUE_ROW_COUNT */
             (COUNT(t1.Special_Person)) AS SMILEY_COUNTROL_VALUE_ROW_COUNT
    FROM WORK.SMILEY_CONTROL_VALUE t1;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_CONTROL_VALUE_ROW_COUNT` from the `SMILEY_CONTROL_VALUE` table containing the `COUNT` of the values of the `Special_Person` variable stored in the `SMILEY_COUNTROL_VALUE_ROW_COUNT` variable.

This code creates the SMILEY_CONTROL_VALUE_MOCK_ROW table:

```
PROC SQL OUTOBS=1;
    CREATE TABLE WORK.SMILEY_CONTROL_VALUE_MOCK_ROW AS
    SELECT /* Special_Person_Flag */
             (0) AS Special_Person_Flag,
           /* Special_Number_Flag */
             (0) AS Special_Number_Flag,
           /* Special_Code_Flag */
             (0) AS Special_Code_Flag,
           /* Load_Date_Flag */
             (0) AS Load_Date_Flag,
           t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date
    FROM WORK.SMILEY_COMPANY t1, WORK.SMILEY_CONTROL_VALUE_ROW_COUNT t2
    WHERE t2.SMILEY_COUNTROL_VALUE_ROW_COUNT = 0;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_CONTROL_VALUE_MOCK_ROW` containing one observation (`OUTOBS=1`).
- The value `0` is assigned to `Special_Person_Flag`, `Special_Number_Flag`, `Special_Code_Flag`, and `Load_Date_Flag`, and `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` are selected from the `SMILEY_COMPANY` table.
- Notice the `FROM` does not contain any type of join between the 2 tables thus a Joinless Join.
- The `WHERE` clause causes the output row to be created with all 4 flags set to `0` only when the value of the `SMILEY_CONTROL_VALUE_ROW_COUNT` = `0` and therefore is a 'Mock Row'.

This code creates the Append of the Smiley_Control_Value table and the Smiley_Control_Value_Mock_Row table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_CONTROL_VALUE_FINAL AS
    SELECT * FROM WORK.SMILEY_CONTROL_VALUE
        OUTER UNION CORR
    SELECT * FROM WORK.SMILEY_CONTROL_VALUE_MOCK_ROW;
QUIT;
```

OR

```
DATA WORK.SMILEY_CONTROL_VALUE_FINAL;
    SET WORK.SMILEY_CONTROL_VALUE
        WORK.SMILEY_CONTROL_VALUE_MOCK_ROW;
RUN;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_CONTROL_VALUE_FINAL` by concatenating the results (`OUTER UNION`) of all columns (`SELECT *`) from the `SMILEY_CONTROL_VALUE` and the `SMILEY_CONTROL_VALUE_MOCK_ROW` tables and overlaying all corresponding (`CORR`) columns.
- The `DATA` step creates a data set (or table) called `SMILEY_CONTROL_VALUE_FINAL` by `SET`ing the `SMILEY_CONTROL_VALUE` data set and the `SMILEY_CONTROL_VALUE_MOCK_ROW` data set with all columns (which are the same in this case) from both data sets.
- The `PROC SQL` and the `DATA` step create the same results, thus either can be used.

This code creates the Final Control Value Report for the Smiley Company One-Way Frequency:

```
PROC SQL;
    CREATE VIEW WORK.SORT AS
           SELECT T.Special_Person_Flag, T.Special_Number_Flag,
                  T.Special_Code_Flag, T.Load_Date_Flag
    FROM WORK.SMILEY_CONTROL_VALUE_FINAL AS T;
QUIT;

TITLE;
TITLE1 "Final Control Value Report for";
TITLE2 "Smiley Company";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System on %TRIM(%QSYSFUNC(DATE(),
           NLDATE20.)) at %TRIM(%SYSFUNC(TIME(), TIMEAMNP12.))";

PROC FREQ DATA=WORK.SORT
       ORDER=INTERNAL;
       TABLES Special_Person_Flag /  SCORES=TABLE;
       TABLES Special_Number_Flag /  SCORES=TABLE;
       TABLES Special_Code_Flag   /  SCORES=TABLE;
       TABLES Load_Date_Flag      /  SCORES=TABLE;
RUN;

RUN; QUIT;
TITLE; FOOTNOTE;
```

- The `PROC SQL` creates a `VIEW` from the `SMILEY_CONTROL_VALUE_FINAL` table containing only the variables which are to be included in the One-Way Frequency.
- The `TITLE` and `FOOTNOTE` statements with no title or footnote clear all titles and footnotes, and the `TITLE1`, `TITLE2`, and `FOOTNOTE1` statements set the titles and footnote.
- `FOOTNOTE1` is an optional default which is always added by SAS Enterprise Guide.
- The `PROC FREQ` creates `TABLES` for each `Flag` containing the `SCORES` or values of each `Flag` listed by `INTERNAL` or numeric/alphabetic `ORDER`.
- Both `ORDER=INTERNAL` and `SCORES=TABLE` are optional defaults for `PROC FREQ` which are always added by SAS Enterprise Guide.

This code creates the SMILEY_JOINLESS_JOIN_CALCULATION table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_JOINLESS_JOIN_CALCULATION AS
    SELECT t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date,
           /* Special_Number_Percent */
             (t1.Special_Number/t2.Special_Number_National_Average)
              FORMAT=PERCENT8.1 AS Special_Number_Percent;
    FROM WORK.SMILEY_COMPANY t1, WORK.SPECIAL_NUMBER_NATIONAL_AVERAGE t2;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_JOINLESS_JOIN_CALCULATION` by selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.
- The `Special_Number_Percent` column is calculated by taking the ratio of `Special_Number` from the `SMILEY_COMPANY` table and `Special_Number_National_Average` from the `SPECIAL_NUMBER_NATIONAL_AVERAGE` table and applying the `FORMAT=PERCENT8.1` to obtain the resulting percent instead of the ratio.
- Notice the `FROM` does not contain any type of join between the 2 tables and thus is a Joinless Join.

This code creates the SMILEY_JOINLESS_JOIN_VALIDATION table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_JOINLESS_JOIN_VALIDATION AS
    SELECT t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date,
           /* Date_Validation */
             (CASE
                 WHEN t1.Load_Date GE t2.Load_Date_Check
                     THEN 'AOK'
                 ELSE 'NOT AOK'
              END) AS Date_Validation
    FROM WORK.SMILEY_COMPANY t1, WORK.LOAD_DATE_CHECK t2;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_JOINLESS_JOIN_VALIDATION` by selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.
- The `Date_Validation` column is derived by checking if `Load_Date` from the `SMILEY_COMPANY` table is greater than or equal to (`GE`) `Load_Date_Check` from the `LOAD_DATE_CHECK` table and assigning `'AOK'` or `'NOT AOK'` as a result.
- Notice the `FROM` does not contain any type of join between the 2 tables and thus is a Joinless Join.

This code creates the SMILEY_JOINLESS_JOIN_FILTRATION table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_JOINLESS_JOIN_FILTRATION AS
    SELECT t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date,
    FROM WORK.SMILEY_COMPANY t1, WORK.SPECIAL_CODE_NATIONAL_FOCUS t2
    WHERE t1.Special_Code = t2.Special_Code_National_Focus;
QUIT;
```

- **The `PROC SQL` creates a `TABLE` called `SMILEY_JOINLESS_JOIN_CALCULATION` by selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.**
- **The `WHERE` clause filters the output to include only observations in which `t1.Special_Code` from the `SMILEY_COMPANY` table is equal to `Special_Code_National_Focus` from the `SPECIAL_CODE_NATIONAL_FOCUS` table.**
- **Notice the `FROM` does not contain any type of join between the 2 tables and thus is a Joinless Join.**

This code creates the SMILEY_JOINLESS_JOIN_ALL_CHECKS table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_JOINLESS_JOIN_ALL_CHECKS AS
    SELECT /* Special_Person_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Person_Flag,
           /* Special_Number_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Number_Flag,
           /* Special_Code_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Code_Flag,
           /* Load_Date_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Load_Date_Flag,
           t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date,
           /* Special_Number_Percent */
              (t1.Special_Number/t3.Special_Number_National_Average)
               FORMAT=PERCENT8.1 AS Special_Number_Percent,
           /* Date_Validation */
              (CASE
                  WHEN t1.Load_Date GE t4.Load_Date_Check
                      THEN 'AOK'
                  ELSE 'NOT AOK'
               END) AS Date_Validation,
           /* Special_Code_Match */
              (CASE
                  WHEN t1.Special_Code = t5.Special_Code_National_Focus
                      THEN 'MATCH'
                  ELSE 'NO MATCH'
               END) AS Special_Code_Match
    FROM WORK.SMILEY_COMPANY t1, WORK.SMILEY_CONTROL_VALUE_ROW_COUNT t2,
         WORK.SPECIAL_NUMBER_NATIONAL_AVERAGE t3, WORK.LOAD_DATE_CHECK t4,
         WORK.SPECIAL_CODE_NATIONAL_FOCUS t5;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_JOINLES_JOIN_ALL_CHECKS` by selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.
- If `SMILEY_CONTROL_VALUE_ROW_COUNT` is `0` then a mock row needs to be created with `Special_Person_Flag`, `Special_Number_Flag`, `Special_Code_Flag`, and `Load_Date_Flag` assigned a value `0`; otherwise a mock row is not needed and the flags are set to Null (`.`).
- The `Special_Number_Percent`, `Date_Validation`, and `Special_Code_Match` columns are calculated or derived as in the previous examples; however `Special_Code_Match` is a derived column rather than an applied filter.
- Notice the `FROM` does not contain any type of join between the 5 tables and thus is a Joinless Join.

This code creates the Append of the Smiley_Control_Value table and the Smiley_Joinless_Join_All_Checks table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_CONTROL_VALUE_FINAL_ALL AS
    SELECT * FROM WORK.SMILEY_CONTROL_VALUE
        OUTER UNION CORR
    SELECT * FROM WORK.SMILEY_JOINLESS_JOIN_ALL_CHECKS;
QUIT;
```

OR

```
DATA WORK.SMILEY_CONTROL_VALUE_FINAL_ALL;
    SET WORK.SMILEY_CONTROL_VALUE
        WORK.SMILEY_JOINLESS_JOIN_ALL_CHECKS;
RUN;
```

- **The `PROC SQL` creates a `TABLE` called `SMILEY_CONTROL_VALUE_FINAL_ALL` by concatenating the results (`OUTER UNION`) of all columns (`SELECT *`) from the `SMILEY_CONTROL_VALUE` and the `SMILEY_JOINLESS_JOIN_ALL_CHECKS` tables and overlaying all corresponding (`CORR`) columns.**
- **The `DATA` step creates a data set (or table) called `SMILEY_CONTROL_VALUE_FINAL_ALL` by `SET`ing the `SMILEY_CONTROL_VALUE` data set and the `SMILEY_JOINLESS_JOIN_ALL_CHECKS` data set with all columns (which are the same in this case) from both data sets.**
- **The `PROC SQL` and the `DATA` step create the same results, thus either can be used.**

This code creates the Control Value Report for the Smiley Company All Joinless Joins One-Way Frequency:

```
PROC SQL;
    CREATE VIEW WORK.SORT AS
           SELECT T.Special_Person_Flag, T.Special_Number_Flag,
                  T.Special_Code_Flag, T.Load_Date_Flag
    FROM WORK.SMILEY_CONTROL_VALUE_FINAL_ALL AS T;
QUIT;

TITLE;
TITLE1 "Control Value Report for";
TITLE2 "Smiley Company All Joinless Joins";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System on %TRIM(%QSYSFUNC(DATE(),
           NLDATE20.)) at %TRIM(%SYSFUNC(TIME(), TIMEAMNP12.))";

PROC FREQ DATA=WORK.SORT
       ORDER=INTERNAL;
       TABLES Special_Person_Flag /  SCORES=TABLE;
       TABLES Special_Number_Flag /  SCORES=TABLE;
       TABLES Special_Code_Flag   /  SCORES=TABLE;
       TABLES Load_Date_Flag      /  SCORES=TABLE;
RUN;

RUN; QUIT;
TITLE; FOOTNOTE;
```

- The `PROC SQL` creates a `VIEW` from the `SMILEY_CONTROL_VALUE_FINAL_ALL` table containing only the variables which are to be included in the One-Way Frequency.
- The `TITLE` and `FOOTNOTE` statements with no title or footnote clear all titles and footnotes, and the `TITLE1`, `TITLE2`, and `FOOTNOTE1` statements set the titles and footnote.
- `FOOTNOTE1` is an optional default which is always added by SAS Enterprise Guide.
- The `PROC FREQ` creates `TABLES` for each `Flag` containing the `SCORES` or values of each `Flag` listed by `INTERNAL` or numeric/alphabetic `ORDER`.
- Both `ORDER=INTERNAL` and `SCORES=TABLE` are optional defaults for `PROC FREQ` which are always added by SAS Enterprise Guide.

This code creates the JOINLESS_JOIN_NOTHING_IN_COMMON table:

```
PROC SQL;
    CREATE TABLE WORK.JOINLESS_JOIN_NOTHING_IN_COMMON AS
    SELECT t1.SMILEY_CONTROL_VALUE_ROW_COUNT,
           t2.Special_Number_National_Average,
           t3.Load_Date_Check,
           t4.Special_Code_National_Focus,
    FROM WORK.SMILEY_CONTROL_VALUE_ROW_COUNT t1,
         WORK.SPECIAL_NUMBER_NATIONAL_AVERAGE t2, WORK.LOAD_DATE_CHECK t3,
         WORK.SPECIAL_CODE_NATIONAL_FOCUS t4;
QUIT;
```

- **The `PROC SQL` creates a `TABLE` called `JOINLESS_JOIN_NOTHING_IN_COMMON` by selecting `SMILEY_CONTROL_VALUE_ROW_COUNT` from the `SMILEY_CONTROL_VALUE_ROW_COUNT` table, `Special_Number_National_Average` from the `SPECIAL_NUMBER_NATIONAL_AVERAGE` table, `Load_Date_Check` from the `LOAD_DATE_CHECK` table, and `Special_Code_National_Focus` from the `SMILEY_COMPANY` table.**
- **Notice the `FROM` does not contain any type of join between the 4 tables and thus is a Joinless Join.**

This code creates the SMILEY_JOINLESS_JOIN_ALL_AGAIN table:

```
PROC SQL;
    CREATE TABLE WORK.SMILEY_JOINLESS_JOIN_ALL_AGAIN AS
    SELECT /* Special_Person_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Person_Flag,
           /* Special_Number_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Number_Flag,
           /* Special_Code_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Special_Code_Flag,
           /* Load_Date_Flag */
              (CASE  t2.SMILEY_CONTROL_VALUE_ROW_COUNT
                  WHEN 0 THEN 0
                  ELSE .
               END) AS Load_Date_Flag,
           t1.Special_Person,
           t1.Special_Number,
           t1.Special_Code,
           t1.Load_Date,
           /* Special_Number_Percent */
              (t1.Special_Number/t2.Special_Number_National_Average)
               FORMAT=PERCENT8.1 AS Special_Number_Percent,
           /* Date_Validation */
              (CASE
                  WHEN t1.Load_Date GE t2.Load_Date_Check
                      THEN 'AOK'
                  ELSE 'NOT AOK'
               END) AS Date_Validation,
           /* Special_Code_Match */
              (CASE
                  WHEN t1.Special_Code = t2.Special_Code_National_Focus
                      THEN 'MATCH'
                  ELSE 'NO MATCH'
               END) AS Special_Code_Match
    FROM WORK.SMILEY_COMPANY t1, WORK.JOINLESS_JOIN_NOTHING_IN_COMMON t2;
QUIT;
```

- The `PROC SQL` creates a `TABLE` called `SMILEY_JOINLES_JOIN_ALL_CHECKS` by selecting `Special_Person`, `Special_Number`, `Special_Code`, and `Load_Date` from the `SMILEY_COMPANY` table.
- If `SMILEY_CONTROL_VALUE_ROW_COUNT` is `0` then a mock row is needs to be created with `Special_Person_Flag`, `Special_Number_Flag`, `Special_Code_Flag`, and `Load_Date_Flag` assigned a value `0`; otherwise a mock row is not needed and the flags are set to Null (`.`).
- The `Special_Number_Percent`, `Date_Validation`, and `Special_Code_Match` columns are calculated or derived as in the previous examples; however `Special_Code_Match` is a derived column rather than an applied filter.
- Notice the `FROM` does not contain any type of join between the 2 tables and thus is a Joinless Join.

This code creates the Control Value Report for the Smiley Company All Joinless Joins Again One-Way Frequency:

```
PROC SQL;
    CREATE VIEW WORK.SORT AS
           SELECT T.Special_Person_Flag, T.Special_Number_Flag,
                  T.Special_Code_Flag, T.Load_Date_Flag
    FROM WORK.SMILEY_JOINLESS_JOIN_ALL_AGAIN AS T;
QUIT;

TITLE;
TITLE1 "Control Value Report for";
TITLE2 "Smiley Company All Joinless Joins Again";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System on %TRIM(%QSYSFUNC(DATE(),
           NLDATE20.)) at %TRIM(%SYSFUNC(TIME(), TIMEAMNP12.))";

PROC FREQ DATA=WORK.SORT
       ORDER=INTERNAL;
       TABLES Special_Person_Flag /  SCORES=TABLE;
       TABLES Special_Number_Flag /  SCORES=TABLE;
       TABLES Special_Code_Flag   /  SCORES=TABLE;
       TABLES Load_Date_Flag      /  SCORES=TABLE;
RUN;

RUN; QUIT;
TITLE; FOOTNOTE;
```

- The `PROC SQL` creates a `VIEW` from the `SMILEY_JOINLESS_JOIN_ALL_AGAIN` table containing only the variables which are to be included in the One-Way Frequency.
- The `TITLE` and `FOOTNOTE` statements with no title or footnote clear all titles and footnotes, and the `TITLE1`, `TITLE2`, and `FOOTNOTE1` statements set the titles and footnote.
- `FOOTNOTE1` is an optional default which is always added by SAS Enterprise Guide.
- The `PROC FREQ` creates `TABLES` for each `Flag` containing the `SCORES` or values of each `Flag` listed by `INTERNAL` or numeric/alphabetic `ORDER`.
- Both `ORDER=INTERNAL` and `SCORES=TABLE` are optional defaults for `PROC FREQ` which are always added by SAS Enterprise Guide.

ACKNOWLEDGMENTS

We want to thank the 27th Annual MWSUG 2016 Hands-On Workshop Section Co-Chairs, **Dave Foster** and **Chuck Kincaid**, for graciously accepting our abstract and paper. In addition, we want to express our appreciation to the Conference Co-Chairs, **Richann Watson** (Academic Chair) and **Adrian Katschke** (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 our friend, mentor, and fellow SASketeer, **Kirk Paul Lafler**. 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. And in conclusion, we want to give a shout out to our friend, **Charlie Shipp**, for his friendship and faithful service to the SAS World. You both inspire us to share what we are learning and our hope is to be a light of encouragement to you as well **~** ***All for SAS & SAS for All!***

REFERENCES

Carpenter, Art (2012), *PROC REPORT Basics: Getting Started with the Primary Statements*, Proceedings of the 6th Annual SAS Global Forum (SGF) 2012 Conference, California Occidental Consultants, Anchorage, AK, USA.
http://support.sas.com/resources/papers/proceedings12/242-2012.pdf

Celko, Joe (2010), *Joe Celko's SQL for Smarties, Fourth Edition: Advanced SQL Programming (The Morgan Kaufmann Series in Data Management Systems)*; November 10, 2010; ISBN-10: 0123820227; ISBN-13: 978-0123820228.
http://www.accuteach.com/book/joe-celkos-sql-for-smarties-fourth-edition-advanced-sql-programming-the-morgan-kaufmann-series-in-data-management-systems-by-joe-celko/#

Foley, Malachy J. (2005), *Merging vs. Joining: Comparing the DATA Step with SQL*, Proceedings of the 30th Annual SAS Users Group International (SUGI) 2005 Conference, University of North Carolina, Chapel Hill, NC, USA.
http://www.scsug.org/SCSUGProceedings/2005/Foley_Merging%20vs%20Joining%20-%20184.pdf

Kent, Paul, *SQL Joins -- The Long and The Short of It*; SAS Institute Inc., Cary, NC, USA.
http://support.sas.com/techsup/technote/ts553.html

Kerman, Jonathan (2010), *Ordering PROC FREQ Around*, NorthEast SAS Users Group (NESUG) 2010 Conference, Johns Hopkins University, S Baltimore, MD, USA.
http://www.lexjansen.com/nesug/nesug10/cc/cc16.pdf

Lafler, Kirk Paul (2013), *PROC SQL: Beyond the Basics Using SAS, Second Edition*; SAS Press.
http://support.sas.com/publishing/authors/lafler.html

Lafler, Kirk Paul and Mira Shapiro (2013), *Point-and-Click Programming Using SAS® Enterprise Guide®*, NorthEast SAS Users Group (NESUG) 2013 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
http://www.lexjansen.com/nesug/nesug13/63_Final_Paper.pdf

Lafler, Kirk Paul (2012), *Exploring DATA Step Merges and PROC SQL Joins*, Proceedings of the 6th Annual SAS Global Forum (SGF) 2012 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
http://support.sas.com/resources/papers/proceedings12/251-2012.pdf

Lafler, Kirk Paul (2012), *Exploring DATA Step Merges and PROC SQL Joins*, Proceedings of the 14th Annual Pharmaceutical SAS Users Group (PharmaSUG) 2012 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
http://pharmasug.org/proceedings/2012/TA/PharmaSUG-2012-TA02.pdf

Lafler, Kirk Paul (2011), *Output Delivery System (ODS)– Simply the Basics*, Proceedings of the 5th Annual SAS Global Forum (SGF) 2011 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
http://support.sas.com/resources/papers/proceedings11/273-2011.pdf

Phelps, Kent ♥ Ronda Team (2016), *Base SAS® and SAS® Enterprise Guide® ~ Automate Your SAS World With Dynamic Code; Your Newest BFF (Best Friend Forever) in SAS*, Proceedings of the 27th Annual MidWest SAS Users Group (MWSUG) 2016 Conference, The SASketeers, Des Moines, IA, USA.

Phelps, Kent ♥ Ronda Team (2015), *The Joinless Join ~ The Impossible Dream Come True; Expanding the Power of SAS® Enterprise Guide® in a New Way*, Proceedings of the 26th Annual MidWest SAS Users Group (MWSUG) 2015 Conference, The SASketeers, Des Moines, IA, USA.
http://www.mwsug.org/proceedings/2015/BI/MWSUG-2015-BI-11.pdf

Phelps, Kent ♥ Ronda Team (2015), *SAS® Enterprise Guide® Base SAS® Program Nodes ~ Automating Your SAS World With a Dynamic FILENAME Statement, Dynamic Code, and the CALL EXECUTE Command; Your Newest BFF (Best Friends Forever) in SAS*, Proceedings of the 26th Annual MidWest SAS Users Group (MWSUG) 2015 Conference, The SASketeers, Des Moines, IA, USA.
http://www.mwsug.org/proceedings/2015/TT/MWSUG-2015-TT-05.pdf

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2014), *The Joinless Join; Expand the Power of SAS® Enterprise Guide® in a New Way*, Proceedings of the 25th Annual MidWest SAS Users Group (MWSUG) 2014 Conference, The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.
http://www.mwsug.org/proceedings/2014/BI/MWSUG-2014-BI12.pdf

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2014), *SAS® Commands PIPE and CALL EXECUTE; Dynamically Advancing From Strangers to Your Newest BFF (Best Friends Forever)*, Proceedings of the 25th Annual MidWest SAS Users Group (MWSUG) 2014 Conference, The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.
http://www.mwsug.org/proceedings/2014/BI/MWSUG-2014-BI13.pdf

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2013), *The Joinless Join; Expand the Power of SAS® Enterprise Guide® in a New Way*, Presented at Iowa SAS Users Group (IASUG), The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2013), *SAS® Commands PIPE and CALL EXECUTE; Dynamically Advancing From Strangers to Best Friends*, Presented at Iowa SAS Users Group (IASUG), The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2013), *The Joinless Join; Expand the Power of SAS® Enterprise Guide® in a New Way*, Proceedings of the 24th Annual MidWest SAS Users Group (MWSUG) 2013 Conference, The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.
http://www.mwsug.org/proceedings/2013/BB/MWSUG-2013-BB06.pdf

Phelps, Kent ♥ Ronda Team and Kirk Paul Lafler (2013), *SAS® Commands PIPE and CALL EXECUTE; Dynamically Advancing From Strangers to Best Friends*, Proceedings of the 24th Annual MidWest SAS Users Group (MWSUG) 2013 Conference, The SASketeers, Des Moines, IA, and Software Intelligence Corporation, Spring Valley, CA, USA.
http://www.mwsug.org/proceedings/2013/00/MWSUG-2013-0003.pdf

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. Please refer to your specific Operating System (e.g. UNIX, Windows, or z/OS) Manual, Installation Configuration, and/or in-house Technical Support for further guidance in how to create the SAS code presented in this paper.