ABSTRACT

There’s SuperCE for comparing text files on the mainframe. Diff and sdff are there for the UNIX environment. But what about SAS® data sets? A straight text comparison doesn’t tell us nearly enough. This paper explores some options for the CONTENTS and the COMPARE procedures to make testing easier and the results more understandable.

A brief primer on the physical nature of SAS data sets is also included.

INTRODUCTION

It can be more difficult to test new SAS data sets than to create them in the first place. This is especially true when testing differences between data sets created by copying or updating older versions. Manually validating changes thereafter can be time-consuming and error prone. Fortunately, SAS has some procedures that with the right options, will automate much of this. The CONTENTS procedure lets us quickly see how many variables are in a data set, their names and types. With a few exceptions, we can even see how many observations are there without having to read the entire data set. The COMPARE procedure takes us deeper with several useful options. This paper will list appropriate options for each section and then go more deeply into those the author has found especially useful. We will also discover ways to save the results to a data set and get more detailed results with just a little more effort.

THE PHYSICAL NATURE OF SAS DATA SETS

All SAS data sets consist of data and a descriptor section which has information about the data such as name of the data set, the system path, the variable names and the variable types. In addition, if the data set is on disk and not a view, the number of observations will be listed as well. When the data set is first created, SAS must go back to the descriptor section at the front of the data set and add the number of observations. This backward update is obviously impossible for tape. SAS views are basically stored SAS code used to limit the number of observations or variables retrieved. They can also be used to perform other operations. We can’t know how observations we will return from a view before we use it so here also the observation count will be unknown. If SAS was used to sort the data set, the variables by which it was sorted will also be stored in the descriptor. Any indexes will also be listed. PROC CONTENTS reads the descriptor information, giving us a speedy overview of all these properties. Information about the data set can be stored in another data set for future use. See the Appendix for an example PROC CONTENTS output use in a macro for renaming variables without the developer having to know all their names, utilized in multiple MERGE statements.

Speaking of the MERGE statement, there are many ways to create SAS data sets. We could read in various non-SAS inputs, copy another data set or match data sets together using the afore-mentioned SAS merge or the SQL procedure joins. If using other SAS data sets to create your new data set, there are two things to keep in mind. One is that internally there are only two SAS variables types, character or numeric. If you attempt to combine two variables with the same name but different types, you will get an error. There also cannot be two SAS variables with the same name in the same data set. If you merge or join two data sets together, only one of the variables can survive. SAS will not rename them for you automatically (although SAS Enterprise Guide Query Builder will!). One or the other will be kept depending upon the method. If you use the SQL procedure, SAS will keep the values from the first variable. If you create the data set with a MERGE statement, SAS will keep the last variable.

A TALE OF TWO DATA SETS

Let us examine two data sets developed in an effort to track orders and analyze shipping issues. The current version of our data set has two possible product statuses: ‘A’ (active) and ‘S’ (shipped). We wanted a new product status added, ‘B’ for back-order. The product status would be ‘B’ if the expected ship date was 60 days or more out from the previous expected ship date. The item on a particular order is identified by the the order number (ORD_NO) and the product code (PROD_CDE). We will look at these two variables, order date (ORDER_DT), expected shipping date (EXPECTED_SHIP_DT) and product status (PROD_STATUS). First we’ll do PROC CONTENTS for both the new and old versions. Figure 1 shows the PROC CONTENTS for the old version, ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP, hereafter referred to as BASE and figure 2 for the new version, ORDERS.ORDER_ANALYSIS_SEPT2012 now referred to as COMPARE.
Note five items about both data sets:

1. Both have a member type of DATA and engine V9. The backup was created just before the current version.
2. Each has five variables but the BASE actually has more observations, 486,961 vs. 486,439 – not what we were expecting.
3. The encoding is the same and the directory location is the same for both.
4. Both data sets are sorted by SAS. But...
5. They aren't sorted on the same variables.
READING THE OUTPUT FROM PROC COMPARE

The proc CONTENTS tells us a lot about our two data sets. It’s nice to know how the data sets are sorted. But if you ignore that they are sorted in different ways, you will get a comparison result that looks like the one shown in Figures 3 through 6. This would be the simplest form of proc COMPARE:

```
PROC COMPARE BASE=ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP
    COMPARE=ORDERS.ORDER_ANALYSIS_SEPT2012;
RUN;
```

Let’s take a look at this report. First we’ll look at the Data Set Summary in figure 3.

![Figure 3: the Data Set Summary](image)

1. The comparison method is EXACT, the default – other comparison methods will be discussed later
2. Created Date is equal to the Modified Date for both data sets
3. Both have the same number of variables
4. As we saw with the proc CONTENTS, there are actually more observations in the BASE than in the COMPARE

Then there is the Variables Summary:

![Figure 4: The Variables Summary](image)

1. One variable has differing attribute
2. That attribute is the format which is probably not a problem.

The Observation Summary in figure 5 shows how just how much we needed to sort both BASE and COMPARE on the same variables.

![Figure 5: the Observation Summary](image)

1. The first inequality was found at the first observation
2. BASE has 522 fewer observations than COMPARE
3. Only four observations had all variables compared equally?
Figure 6 shows us the Values Comparison Summary. This looks just as much off as the Observation Comparison Summary:

### Values Comparison Summary

- Number of Variables Compared with All Observations Equal: 0
- Number of Variables Compared with Some Observations Unequal: 0
- Total Number of Values which Compare Unequal: 2037708
- Maximum Difference: 25693745

The COMPARE Procedure

**Comparison of ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP with ORDERS.ORDER_ANALYSIS_SEPT2012 (Method=EXACT)**

All Variables Compared have Unequal Values

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Ndif</th>
<th>MaxDiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD_CDE</td>
<td>CHR</td>
<td>3416</td>
<td>245</td>
<td>245</td>
</tr>
<tr>
<td>ORD_NO</td>
<td>NUM</td>
<td>8486435</td>
<td>2.669E7</td>
<td>2.669E7</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>NUM</td>
<td>8493527</td>
<td>577</td>
<td>577</td>
</tr>
<tr>
<td>EXPECTED_SHIP_DT</td>
<td>NUM</td>
<td>8483646</td>
<td>577</td>
<td>577</td>
</tr>
<tr>
<td>PROD_STATUS</td>
<td>CHR</td>
<td>1167875</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: the Values Comparison Summary
1. There were no observations that compared equal.
2. The Ndif shows how many had different values but the amount is more than the observations in both data sets added together.
3. The MaxDiff is the maximum difference between the values for BASE and COMPARE, populated for numeric variables only.

Figure 7 shows the Value Comparison Results for Variables. Does this comparison tell us anything?

<table>
<thead>
<tr>
<th>Obs</th>
<th>Base Value</th>
<th>Compare Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PROD_CDE</td>
<td>PROD_CDE</td>
</tr>
<tr>
<td>1</td>
<td>VJC</td>
<td>AEC</td>
</tr>
<tr>
<td>2</td>
<td>DRB</td>
<td>AEC</td>
</tr>
<tr>
<td>3</td>
<td>DRB</td>
<td>AEC</td>
</tr>
</tbody>
</table>

Figure 7: Value Comparison Results for Variable

Clearly, we need to sort these data sets and then match on something meaningful like order number and product code. We will sort them by these variables, creating work data set views OLD_ORDERS and NEW_ORDERS. See the appendix for the PROC SQL code to do this.

### THE BY AND ID OPTIONS

We can add the statements **BY ORD_NO PROD_CDE** or **ID ORD_NO PROD_CDE** to our PROC COMPARE for a comparison by order number and product code. The **BY** option performs a separate comparison for each ORD_NO and PROD_CDE match while the **ID** option simply identifies which variables on which to match your observations. Either option requires both data sets to be sorted in the same order as in our views.

The author has found the **ID** option far more useful than **BY** due to **BY**'s unfortunate habit of running out of memory when data sets are of any size. Therefore we will use the **ID** statement:

```
PROC COMPARE BASE=OLD_ORDERS COMPARE=NEW_ORDERS;
ID ORD_NO PROD_CDE;
```

The Data Summary section reflects that we are comparing views – the creation dates are the same for both views. The number of observations is missing because we won’t know how many observations are actually there until the view is used.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Created</th>
<th>Modified</th>
<th>NVar</th>
<th>NObs</th>
</tr>
</thead>
</table>

Figure 8: Data Summary for ID Variables ORD_NO and PROD_CDE set up with views
The Observation Summary looks different too:

1. The first observation in the BASE does not have one in COMPARE. The first observations to match are 526 for BASE and 1 for COMPARE. The first inequality found is on order no 90234, product code DRV.
2. There are 525 observations in BASE that are not in COMPARE and 3 in COMPARE not in BASE.
3. Many fewer compare as unequal.

Our Values Comparison Summary in figure 10 confirms this:

1. One variable is actually equal for all observations.
2. The maximum difference for any pair of variables is 250.
3. The two variables with differences are EXPECTED_SHIP_DT and PROD_STATUS. It is the EXPECTED_SHIP_DT that has the differences calculated as date variables are numeric.
4. PROD_STATUS has the count of differences (62368) but not the maximum difference because it is character.

Our Variables with Unequal Values listing reflects our ID variables ORD_NO and PROD_CODE and more reasonable results:

1. The ID variables ORD_NO and PROD.CODE appear on the left side, separated by ||.
2. We see the EXPECTED_SHIP_DT for both the BASE and COMPARE side by side.
3. The difference for these examples is 250 (same as maximum difference) and percent difference is 1.3104.
4. PROD_STATUS is character so the difference and percent difference calculations don’t apply.

### Observation Summary

<table>
<thead>
<tr>
<th>Observation</th>
<th>Base</th>
<th>COMPARE</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Obs</td>
<td>1</td>
<td>-</td>
<td>ORD_NO=00000000000087652 PROD_CODE=VJC</td>
</tr>
<tr>
<td>First Match</td>
<td>526</td>
<td>1</td>
<td>ORD_NO=00000000000087652 PROD_CODE=VJC</td>
</tr>
<tr>
<td>First Unequal</td>
<td>14</td>
<td>ORD_NO=00000000000090234 PROD_CODE=DRV</td>
<td></td>
</tr>
<tr>
<td>Last Unequal</td>
<td>486961</td>
<td>486436</td>
<td>ORD_NO=0000000000009346661 PROD_CODE=DRV</td>
</tr>
<tr>
<td>Last Match</td>
<td>486961</td>
<td>486436</td>
<td>ORD_NO=0000000000009346661 PROD_CODE=DRV</td>
</tr>
<tr>
<td>Last Obs</td>
<td>486439</td>
<td>-</td>
<td>ORD_NO=0000000000009346661 PROD_CODE=EAL</td>
</tr>
</tbody>
</table>

Number of Observations in Common: 406436.
Number of Observations in WORK.OLD_ORDERS but not in WORK.NEW_ORDERS: 525.
Number of Observations in WORK.NEW_ORDERS but not in WORK.OLD_ORDERS: 3.
Total Number of Observations Read from WORK.OLD_ORDERS: 406961.
Total Number of Observations Read from WORK.NEW_ORDERS: 406439.

Number of Observations with Some Compared Variables Unequal: 62368.
Number of Observations with All Compared Variables Unequal: 424068.

### Values Comparison Summary for Matched Observations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Ndif</th>
<th>MaxDif</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPECTED_SHIP_DT</td>
<td>NUM</td>
<td>8</td>
<td>62368</td>
<td>250</td>
</tr>
<tr>
<td>PROD_STATUS</td>
<td>CHAR</td>
<td>1</td>
<td>62368</td>
<td>4</td>
</tr>
</tbody>
</table>

### Variables with Unequal Values

| ORD_NO | PROD_CODE || EXPECTED_SHIP_DT | Diff. | % Diff |
|--------|-----------||------------------|-------|-------|
| 000000000000090234 DRV | 03/26/12 | 12/01/12 | 250.0000 | 1.3104 |
| 000000000000090655 DRV | 03/26/12 | 12/01/12 | 250.0000 | 1.3104 |
| 000000000000091926 DRV | 03/26/12 | 12/01/12 | 250.0000 | 1.3104 |

| ORD_NO | PROD_CODE || PROD_STATUS | Diff. | % Diff |
|--------|-----------||-------------|-------|-------|
| 000000000000090234 DRV | A | B |
| 000000000000090655 DRV | A | B |
| 000000000000091926 DRV | A | B |
A MATTER OF METHODS

Can we avoid a comparison result like this?

<table>
<thead>
<tr>
<th>BASE</th>
<th>COMPARE</th>
<th>DIFF.</th>
<th>% DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>508.0500</td>
<td>508.0500</td>
<td>-6.79E-15</td>
<td>-2.254E-15</td>
</tr>
</tbody>
</table>

The variable was probably calculated without truncation or rounding. The default method EXACT makes an exact comparison which will compare all eight positions in a standard numeric variable, regardless of how many places are actually of interest. There are three other methods we could use to circumvent this. We will use the OLD_ORDERS and NEW_ORDERS data views to demonstrate.

You might remember that in Figure 10, we saw the maximum difference for the EXPECTED_SHIP_DT was 250. The ABSOLUTE method looks at the absolute value of BASE – COMPARE and uses the value set by the CRITERION statement as the cut-off. If we make the criterion 250 as in the code below, we will see the Values Comparison Summary in Figure 12 change.

```plaintext
PROC COMPARE BASE=OLD_ORDERS METHOD=absolute CRITERION=250
COMPARE=NEW_ORDERS;
ID ORD_NO PROD_CDE;
RUN;
```

This is a good place to mention the other output options: a) LISTBASEOBS - all observations only found in the base data set, b) LISTBASEVAR – all variables only found in the base data set, c) LISTBASE – both variables and observations found only in the base data set, d) LISTCOMPBAS - all observations found only in the comparison data set, e) LISTCOMPVAR – all variables found only in the comparison data set, f) LISTCOMP - both variables and observations found only in the comparison data set, g) LISTBASE – all observations found in only one data set, h) LISTVAR – all variables found only in one data set, i) LISTEQUALVAR – variables whose values are judged equal and j) LISTALL – all of the above. Later, we will see LISTALL’s impact upon our listing.

---

The **COMPARE** Procedure

**Comparison of WORK.OLD_ORDERS with WORK.NEW_ORDERS**

(Method=ABSOLUTE, Criterion=250)

**Values Comparison Summary**

1. Number of Variables Compared with All Observations Equal: 2.
3. Total Number of Values which Compare Unequal: 62368.
4. Total Number of Values not EXACTLY Equal: 124736.

**Variables with Unequal Values**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Ndif</th>
<th>MaxDiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD_STATUS</td>
<td>CHAR</td>
<td>1</td>
<td>62368</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12: ABSOLUTE Method Values Comparison Summary

1. ABSOLUTE Method and Criterion are part of the heading.
2. Down to just one variable with some unequal but...
3. Comparison notes the number comparing not exactly equal. Shows the maximum difference which is equal to the criterion (250 vs. 250)
4. No effect on PROD_STATUS or indeed any character variable.

Another method is **PERCENT**. This is defined as the absolute value of BASE minus COMPARE divided by BASE = ABS((BASE-COMPARE)/BASE). We will specify criterion=1.5 to leave out anything with 1.5% difference or less.

```plaintext
PROC COMPARE BASE=OLD_ORDERS METHOD=percent CRITERION=1.5
COMPARE=NEW_ORDERS MAXPRINT=(100,2000);
ID ORD_NO PROD_CDE;
RUN;
```
Our new results by percent are:

### Figure 13: PERCENT Method Values Comparison Summary
1. PERCENT Method and Criterion are part of the heading
2. Down to just one variable with some unequal but-
3. Comparison notes the number comparing not exactly equal. Shows the maximum percent which is below our criterion (1.3104 vs. 1.5)
4. No effect on PROD_STATUS or indeed any character variable, just like the ABSOLUTE method

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Ndif</th>
<th>MaxCrit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD_STATUS</td>
<td>CHAR</td>
<td>1</td>
<td>62368</td>
<td></td>
</tr>
</tbody>
</table>

The last is the RELATIVE method which compares the absolute relative difference to the value specified by the criterion. The $\delta$ is the numeric precision of the system – the default is 0 which we’ll use here.

$$\frac{\text{ABS}(y - x)}{((\text{ABS}(x) + \text{ABS}(y))/2) + \delta}$$

Figure 14 Taken from SAS Online Manual

If we did this equation for one of the maximum differences between our BASE and COMPARE data sets, we would get:

$$\text{Abs}(19078-19328)/((\text{Abs}(19078) + \text{Abs}(19328))/2) = 250/19203 = .013019.$$  Using 0.015 as the criterion we would again remove EXPECTED_SHIP_DT from the results:

```sas
PROC COMPARE BASE=OLD_ORDERS method=relative criterion=0.015
    COMPARE=NEW_ORDERS MAXPRINT=(100,2000)
    ID ORD_NO PROD_CDE;
RUN;
```

### Figure 15: RELATIVE Method Values Comparison Summary
1. RELATIVE Method and Criterion are part of the heading
2. Down to just one variable with some unequal but-
3. Comparison notes the number comparing not exactly equal. Shows the maximum relative value which is below our criterion (0.015 vs. 0.013019)
4. No effect on PROD_STATUS or indeed any character variable

Remember that these methods affect numeric variables only. Character variables will always use the EXACT method.

The trouble with this methods is that they apply to all numeric variables, not just one. Now if we really wanted EXPECTED_SHIP_DT out of the...
comparison we could specify VAR just as we can in many procedures like the PRINT procedure. We could even compare it to a different variable (WITH) as in the following:

```sas
PROC COMPARE BASE=OLD_ORDERS COMPARE=NEW_ORDERS MAXPRINT=(100,2000);
VAR ORDER_DT;
WITH EXPECTED_SHIP_DT;
ID ORD_NO PROD_CDE;
RUN;
```

The above would compare 100% unequally as the comparison would be limited to ORDER_DT in BASE and EXPECTED_SHIP_DT in COMPARE. This would actually be our expected results and would show as in figures 16 and 17:

**Figure 16: proc COMPARE VAR and WITH**
1. Variables Summary shows the number of VAR Statement Variables (1, ORDER_DT)
2. Variables Summary also shows the number of WITH Statement Variables (1,EXPECTED_SHIP_DT)

**Figure 17: Value Comparison Results for Variables ORDER_DT vs. EXPECTED_SHIP_DT**
1. All have unequal values
2. ORDER_DT and EXPECTED_SHIP_DT are listed next to each other
3. Instead of seeing Base ORDER_DT and Compare ORDER_DT in results, we see Base ORDER_DT and Compare EXPECTED_SHIP_DT

**GETTING MORE DETAILS**

Is there any way we can see just which observations were in one data set but not in another? Can we control the number of differences printed for each variable and the number overall? What do we do if the data sets are very large with many differences. As is so often the case with SAS, there are a number of options to control our output. One is to use the LISTALL (already mentioned) and MAXPRINT options. LISTALL will show the observations in one data set not in another, one by one. MAXPRINT will let you control how many differences print out. Its syntax is MAXPRINT(maximum number of lines printed per variable, maximum total lines). For example, we could use these options to list the observations in BASE not in COMPARE and vice versa as well as to limit the number of lines print to one hundred per variable, two thousand total.

```sas
PROC COMPARE BASE=OLD_ORDERS COMPARE=NEW_ORDERS LISTALL MAXPRINT=(100,2000) ;
ID ORD_NO PROD_CDE;
RUN;
```
**Comparison Results for Observations**

Observation 1 in WORK.OLD_ORDERS not found in WORK.NEW_ORDERS:
ORD_NO=0000000000087652 PROD_CDE=VJC.

Observation 2 in WORK.OLD_ORDERS not found in WORK.NEW_ORDERS:
ORD_NO=0000000000087653 PROD_CDE=DBE.

Observation 3 in WORK.OLD_ORDERS not found in WORK.NEW_ORDERS:
ORD_NO=0000000000087654 PROD_CDE=DBE.

- and so on -

Observation 486437 in WORK.NEW_ORDERS not found in WORK.OLD_ORDERS:
ORD_NO=0000000029346662 PROD_CDE=CAS.

Observation 486438 in WORK.NEW_ORDERS not found in WORK.OLD_ORDERS:
ORD_NO=0000000029346662 PROD_CDE=DRV.

Observation 486439 in WORK.NEW_ORDERS not found in WORK.OLD_ORDERS:
ORD_NO=0000000029346663 PROD_CDE=EAL.

---

**Figure 18: Comparison Results for Observations**

Instead of getting a listing of 32,000 or so, depending on your installation's default setting, we'll get only 100 for EXPECTED_SHIP_DT and PROD_STATUS each.

What if we need even more detail? We know PROD_STATUS changed but we really don't know from what to what for all 62,368 observations. And what about the ones that didn't change? What can we find out about them?

Fortunately like so many SAS procedures we can set our output to a file like this:

```
PROC COMPARE BASE=OLD_ORDERS COMPARE=NEW_ORDERS LISTALL
   MAXPRINT=(100,2000) OUT=compout1;
   ID ORD_NO PROD_CDE;
RUN;
```

**COMPOUT1** will show one observation, _TYPE_ = ‘DIF’ for each ID group which shows the differences for each variable, whether there are actually differences or not.

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>OBS</th>
<th>ORD_NO</th>
<th>PROD_CDE</th>
<th>ORDER_DT</th>
<th>EXPECTED_SHIP_DT</th>
<th>PROD_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIF</td>
<td>13</td>
<td>89933</td>
<td>KU</td>
<td>1/1/1960</td>
<td>1/1/1960</td>
<td>1</td>
</tr>
<tr>
<td>DIF</td>
<td>14</td>
<td>90234</td>
<td>DRV</td>
<td>1/1/1960</td>
<td>9/7/1960</td>
<td>X</td>
</tr>
</tbody>
</table>

---

**Figure 19: Default Data Set Output from Proc COMPARE**

1. This observation had no difference in EXPECTED_SHIP_DT so the value is 0. The date format shows it as 1/1/1960, SAS’s day zero.
2. There were no differences in PROD_STATUS either. A dot (.) shows in each position of a character variable where there is no difference.
3. For this observation, there was a difference in EXPECTED_SHIP_DT. It was 250 which shows as 9/7/1960 in SAS date format.
4. PROD_STATUS had a difference for this observation which is manifested with an ‘X’ value.

There are several other options we could specify for the output like OUTBASE (lists all BASE records), OUTCOMPARE (lists all COMPARE observations), OUTDIFF (lists all matching observations as the previous example and is the default) and OUTPERCENT (lists percent difference from BASE for numeric variables). We could also denote the OUTNOEQUALS option which suppresses output of any variables equal between Base and Compare. The OUTALL option will give us everything, resulting in a data set that has between one and four observations per comparison by the ID variables ORD_NO and PROD_CDE. If BASE, that means the id variables were found in the Base data set. If COMPARE, this means the id variables were found in the Compare data set. The third will be PERCENT which will have the percent difference between the Base and Compare data sets.
Let’s take a look at our output structure when we use OUTALL(Figure 20):

<table>
<thead>
<tr>
<th>TYPE</th>
<th>OBS</th>
<th>ORD_NO</th>
<th>PROD_CDE</th>
<th>ORDER_DT</th>
<th>EXPECTED_SHIP_DT</th>
<th>PROD_STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE</td>
<td>1</td>
<td>87652</td>
<td>VJC</td>
<td>8/2/2010</td>
<td>8/27/2010</td>
<td>S</td>
</tr>
<tr>
<td>BASE</td>
<td>18618</td>
<td>1107184</td>
<td>VJC</td>
<td>3/8/2012</td>
<td>4/2/2012</td>
<td>A</td>
</tr>
<tr>
<td>COMPARE</td>
<td>18093</td>
<td>1107184</td>
<td>VJC</td>
<td>3/8/2012</td>
<td>4/2/2012</td>
<td>A</td>
</tr>
<tr>
<td>DIF</td>
<td>18093</td>
<td>1107184</td>
<td>VJC</td>
<td>1/1/1960</td>
<td>1/1/1960</td>
<td></td>
</tr>
<tr>
<td>PERCENT</td>
<td>18093</td>
<td>1107184</td>
<td>VJC</td>
<td>1/1/1960</td>
<td>1/1/1960</td>
<td></td>
</tr>
<tr>
<td>BASE</td>
<td>539</td>
<td>90234</td>
<td>DRV</td>
<td>3/1/2012</td>
<td>3/26/2012</td>
<td>A</td>
</tr>
<tr>
<td>COMPARE</td>
<td>14</td>
<td>90234</td>
<td>DRV</td>
<td>3/1/2012</td>
<td>12/1/2012</td>
<td>B</td>
</tr>
<tr>
<td>DIF</td>
<td>14</td>
<td>90234</td>
<td>DRV</td>
<td>1/1/1960</td>
<td>9/7/1960</td>
<td>X</td>
</tr>
<tr>
<td>PERCENT</td>
<td>14</td>
<td>90234</td>
<td>DRV</td>
<td>1/1/1960</td>
<td>1/2/1960</td>
<td>X</td>
</tr>
<tr>
<td>COMPARE</td>
<td>486437</td>
<td>25346662</td>
<td>CAS</td>
<td>9/13/2012</td>
<td>9/28/2012</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 20: COMPARE Output Data Set with OUTALL option
1. ORD_NO and PROD_CDE combination are found only in BASE so that is the only observation
2. ORD_NO and PROD_CDE combination are found in both so all four records appear. The BASE and COMPARE observation types show the values for each
3. The DIF observation shows the differences which in this case are none – 0 (1/1/1960) for EXPECTED_SHIP_DT and . for PROD_STATUS. Note that PERCENT for PROD_STATUS is also .
4. ORD_NO and PROD_CDE combination are found in both so all four observation types show here as well but...
5. There is a difference for EXPECTED_SHIP_DT: 250 or 9/7/1960 and
6. A difference for PROD_STATUS as shown by the ‘X’ value in the DIF observation. Note that PERCENT for PROD_STATUS is also X.

It should be noted that only the variables common to both data sets will be stored here. Any variables in one data set not in the other will be dropped.

What would be really helpful is seeing the BASE, COMPARE and DIFF observations side by side. Even better would be a report or frequency that would show our product statuses by the changes in expected ship dates. See the appendix for a macro and data step that accomplishes the data set creation for a frequency like this:

```sql
PROC FREQ DATA=COMPAREALL;
 TABLES MATCH_IND*BASE_PROD_STATUS*COMPARE_PROD_STATUS*DIFF_EXPECTED_SHIP_DT/LIST MISSING;
RUN;
```

<table>
<thead>
<tr>
<th>match_ind</th>
<th>Base Prod Status</th>
<th>Compare Prod Status</th>
<th>Differences in Expected Ship Date</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE ONLY</td>
<td>A</td>
<td></td>
<td>Completely Missing</td>
<td>2</td>
<td>0.00</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>BASE ONLY</td>
<td>B</td>
<td></td>
<td>Completely Missing</td>
<td>1</td>
<td>0.00</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>COMPARE ONLY</td>
<td>S</td>
<td></td>
<td>Completely Missing</td>
<td>525</td>
<td>0.83</td>
<td>528</td>
<td>0.83</td>
</tr>
<tr>
<td>ID MATCHED</td>
<td>A</td>
<td>B</td>
<td>No Change</td>
<td>54819</td>
<td>87.16</td>
<td>55347</td>
<td>87.99</td>
</tr>
<tr>
<td>ID MATCHED</td>
<td>A</td>
<td>B</td>
<td>Over 180 Days</td>
<td>7549</td>
<td>12.01</td>
<td>62896</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Figure 21: Frequency of BASE by COMPARE by Differences in Expected Ship Dates

If we wanted some statistics like min, max, etc. for our numeric variables, we could add one more option. This is OUTSTATS and would go on the same line as OUT:

```sql
PROC COMPAIR BASE=OLD_ORDERS COMPARE=NEW_ORDERS LISTALL MAXPRINT=(100,2000) OUT=compout1 OUTSTATS=compstats;
ID ORD_NO PROD_CDE;
RUN;
```
We would get the statistics in the table for all the observations in common, in this case 486,436.

<table>
<thead>
<tr>
<th><em>VAR</em>-</th>
<th><em>TYPE</em>-</th>
<th><em>BASE</em></th>
<th><em>COMP</em></th>
<th><em>DIF</em></th>
<th><em>PCTDIF</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER_DT</td>
<td>N</td>
<td>486436</td>
<td>486436</td>
<td>486436</td>
<td>486436</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>MEAN</td>
<td>19150</td>
<td>19150.29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>STD</td>
<td>56.17</td>
<td>56.16954</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>MAX</td>
<td>19248</td>
<td>19248</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>MIN</td>
<td>19053</td>
<td>19053</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>STDERR</td>
<td>0.0805</td>
<td>0.080536</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>T</td>
<td>237787</td>
<td>237786.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>PROBT</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>NDIF</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>DIFMEANS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ORDER_DT</td>
<td>R, RSQ</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 22: OUTSTATS for ORDER_DT

CONCLUSION
With just a few options, we've seen how to create some rather sophisticated comparison reports. Saving the output to a data set gives us even more opportunity to really analyze our test results. All this without tedious if-then or array logic. PROC CONTENTS and COMPARE - Don't test without them.

ACKNOWLEDGEMENTS
Thank you, Joe and Paul Butkovitch for reviewing this paper and presentation

CONTACT INFORMATION
Your comments, questions and experiences are valued and encouraged. Contact the author at:

Patricia Hettinger
Oakbrook Terrace, IL 60523
Phone: 331-462-2142, cell 630-309-3431
Email: patricia_hettinger@att.net

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. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

APPENDIX
A. SQL Creating Sorted Views
   
   PROC SQL;
   CREATE VIEW OLD_ORDERS as SELECT *
   FROM ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP
   ORDER BY ORD_NO, PROD_CDE;
   CREATE VIEW NEW_ORDERS as SELECT *
   FROM ORDERS.ORDER_ANALYSIS_SEPT2012
   ORDER BY ORD_NO, PROD_CDE;
   QUIT;
B. Macro to rename all variables except those specified to a given prefix

*parameters are the library (lib) member name (dsn), list of variables to leave out (leaveout), the desired prefix (pref) and the name of the variable to return (varback);

```
%MACRO renamePREF(lib,dsn,leaveout,pref,varback);
%GLOBAL &varback;
%LET leaveout2=%cmpres(&leaveout); *take out spaces;
*put in the proper delimiters;
%LET leaveout3=%upcase(%sysfunc(translate(&leaveout2,"','"," , "")));
```

```
PROC CONTENTS DATA=&lib.&dsn. OUT=listconts;
*proc contents to a work data set;
RUN;
DATA listcont2; *personal preference for building variables like this;
ATTRIB build_rename format=$1000.;
RETAIN build_rename;
NAME=UPCASE(NAME); *Name variable in contents output is the name of the
variable – put in upper case;
SET listconts END=EOF; *mark end of data set observations;
IF _n_ eq 1 THEN
build_rename=COMPBL('RENAME=('||NAME||'='||"&PREF."||NAME);
ELSE build_rename= COMPBL(build_rename||" "||NAME||'='||"&PREF."||NAME);
IF EOF THEN DO;
built_rename =COMPBL(TRIM(build_rename)||")");
*put the rename code into the variable requested;
CALL SYMPUT("&varback",build_rename);
END;
WHERE UPCASE(name) not in (&leaveout3);
*in this case, we want to keep the id variable names as is;
RUN;
%MEND;
```

C. Calling the macro for each desired observation type

```
%renamePREF( work,compout1, 'ORD_NO', PROD_CDE',comp_,rename1);
%renamePREF( work,compout1, 'ORD_NO', PROD_CDE',base_,rename2);
%renamePREF( work,compout1, 'ORD_NO', PROD_CDE',diff_,rename3);
```

Results example: Rename1 macro variable will be set to

RENAME=ORDER_DT=COMP_ORDER_DT PROD_STATUS=COMP_PROD_STATUS
EXPECTED_SHIP_DT=COMP_EXPECTED_SHIP_DT
_OBS_ = COMP__OBS_ _TYPE_ = COMP__TYPE_)

D. The DATA step

```
DATA COMPAREALL;
ATTRIB match_ind FORMAT=$12.;
```

```
MERGE compout1(in=a where=(COMP__type_='COMPARE') & rename1)
	compout1 (in=b where=(BASE__type_='BASE') & rename2)
	compout1 (in=c where=(DIFF__type_='DIFF') & rename3);
   if a and not b then match_ind='COMPARE ONLY';
   else if b and not a then match_ind = 'BASE ONLY';
   ELSE IF A AND B AND C THEN match_ind = 'ID MATCH';
   ELSE IF C AND NOT A AND NOT B THEN match_ind = 'ERROR';
   BY ORD_NO PROD_CDE;
```

RUN;