Paper SS-03

New Vs. Old – Under the Hood with Procs CONTENTS and COMPARE

Patricia Hettinger, SAS Professional, Oakbrook Terrace, IL

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.

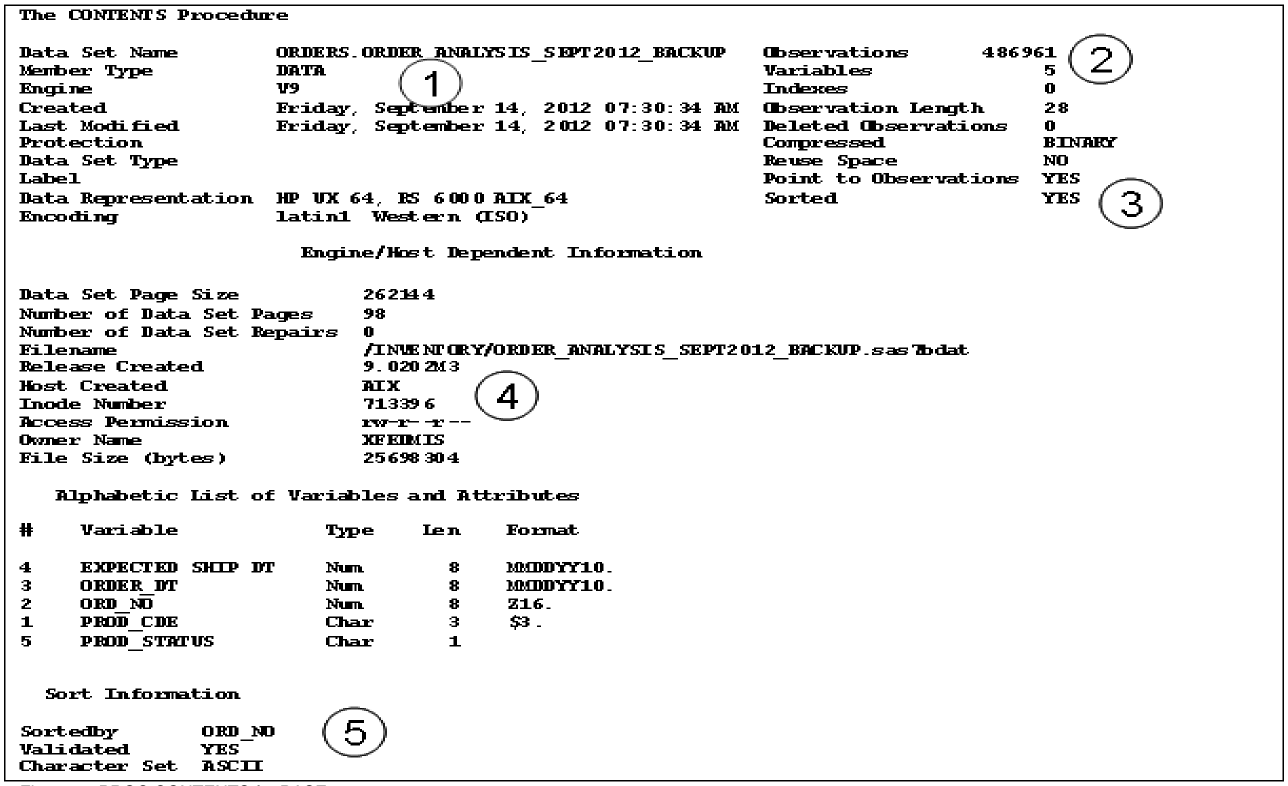
THE PHYSICAL NATURE OF SAS DATA SETS

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.

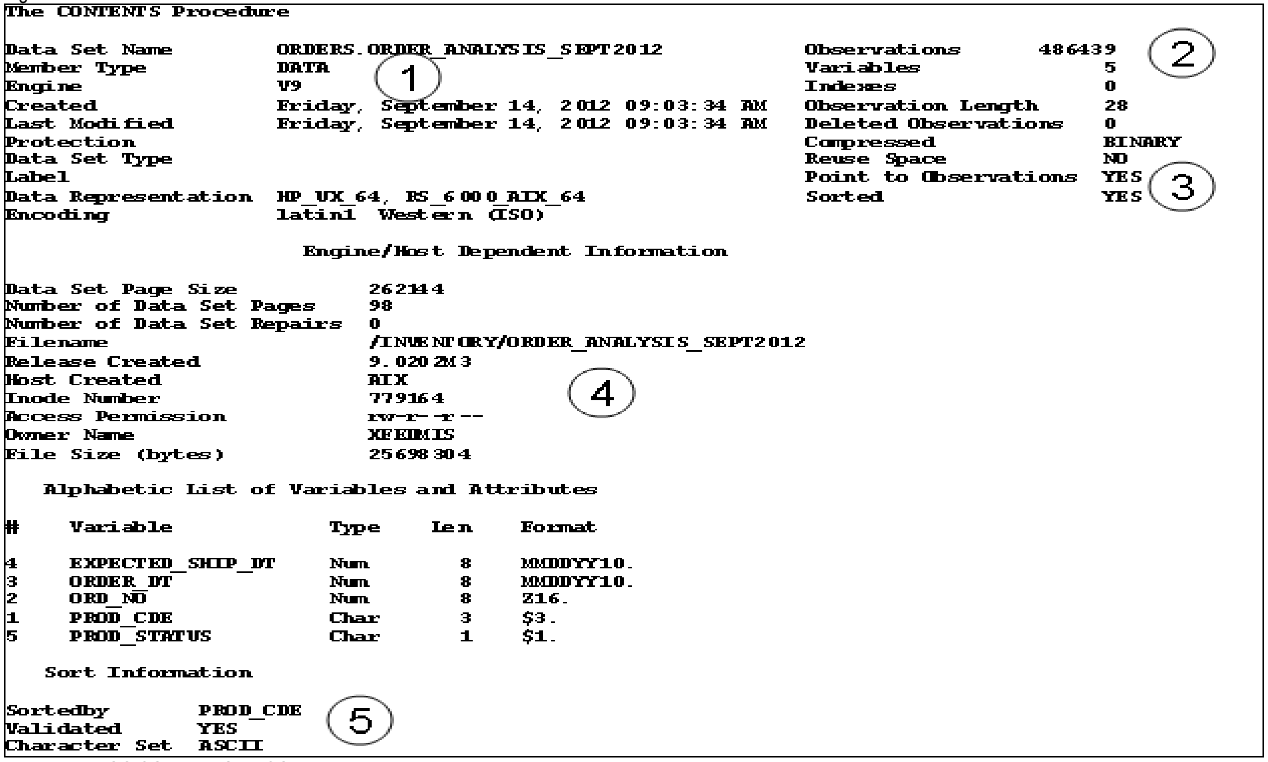
A TALE OF TWO DATA SETS

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 by 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.



```
The CONTENTS Procedure

Data Set Name        ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP   Observations        486961
Member Type          DATA                                    Variables           5
Engine               V9                                      Indexes             0
Created              Friday, September 14, 2012 07:30:34 AM  Observation Length  28
Last Modified        Friday, September 14, 2012 07:30:34 AM  Deleted Observations 0
Protection                                                   Compressed          BINARY
Data Set Type                                                Reuse Space         NO
Label                                                        Point to Observations YES
Data Representation  HP UX 64, RS 6000 AIX_64                Sorted              YES
Encoding             latin1  Western (ISO)

                    Engine/Host Dependent Information

Data Set Page Size         262144
Number of Data Set Pages   98
Number of Data Set Repairs 0
Filename                   /INVENTORY/ORDER_ANALYSIS_SEPT2012_BACKUP.sas7bdat
Release Created            9.0202M3
Host Created               AIX
Inode Number               713396
Access Permission          rw-r--r--
Owner Name                 XFEDMIS
File Size (bytes)          25698304

   Alphabetic List of Variables and Attributes

#    Variable             Type    Len    Format

4    EXPECTED SHIP DT     Num       8    MMDDYY10.
3    ORDER_DT             Num       8    MMDDYY10.
2    ORD_NO               Num       8    Z16.
1    PROD_CDE             Char      3    $3.
5    PROD_STATUS          Char      1


  Sort Information

Sortedby       ORD_NO
Validated      YES
Character Set  ASCII
```

Figure 1: PROC CONTENTS for BASE



```
The CONTENTS Procedure

Data Set Name        ORDERS.ORDER_ANALYSIS_SEPT2012          Observations        486439
Member Type          DATA                                    Variables           5
Engine               V9                                      Indexes             0
Created              Friday, September 14, 2012 09:03:34 AM  Observation Length  28
Last Modified        Friday, September 14, 2012 09:03:34 AM  Deleted Observations 0
Protection                                                   Compressed          BINARY
Data Set Type                                                Reuse Space         NO
Label                                                        Point to Observations YES
Data Representation  HP_UX_64, RS_6000_AIX_64                Sorted              YES
Encoding             latin1  Western (ISO)

                    Engine/Host Dependent Information

Data Set Page Size         262144
Number of Data Set Pages   98
Number of Data Set Repairs 0
Filename                   /INVENTORY/ORDER_ANALYSIS_SEPT2012
Release Created            9.0202M3
Host Created               AIX
Inode Number               779164
Access Permission          rw-r--r--
Owner Name                 XFEDMIS
File Size (bytes)          25698304

   Alphabetic List of Variables and Attributes

#    Variable             Type    Len    Format

4    EXPECTED_SHIP_DT     Num       8    MMDDYY10.
3    ORDER_DT             Num       8    MMDDYY10.
2    ORD_NO               Num       8    Z16.
1    PROD_CDE             Char      3    $3.
5    PROD_STATUS          Char      1    $1.

   Sort Information

Sortedby       PROD_CDE
Validated      YES
Character Set  ASCII
```

Figure 2: PROC CONTENTS for COMPARE

Note five items about both data sets:

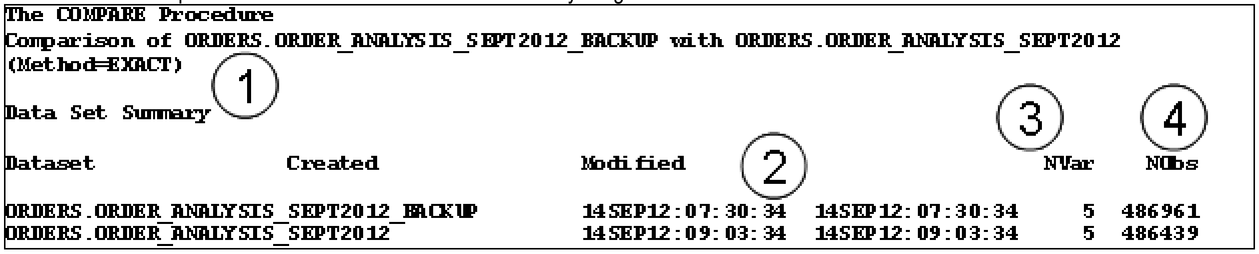
1. Both have a member type of DATA and engine V9. The backup was created just before the current version
2. Each has five variables but the BASE actually has more observations, 486,961vs. 486,439 – not what we were expecting
3. The encoding is the same and the directory location is the same for both
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.

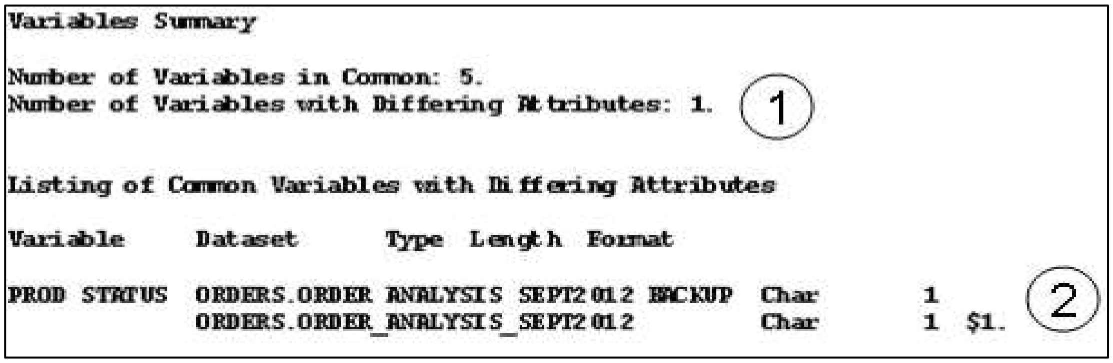


```
The COMPARE Procedure
Comparison of ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP with ORDERS.ORDER_ANALYSIS_SEPT2012
(Method=EXACT)

Data Set Summary

Dataset                       Created           Modified                        NVar    NObs

ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP          14SEP12:07:30:34  14SEP12:07:30:34     5  486961
ORDERS.ORDER_ANALYSIS_SEPT2012                 14SEP12:09:03:34  14SEP12:09:03:34     5  486439
```

Figure 3: the Data Set Summary

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:

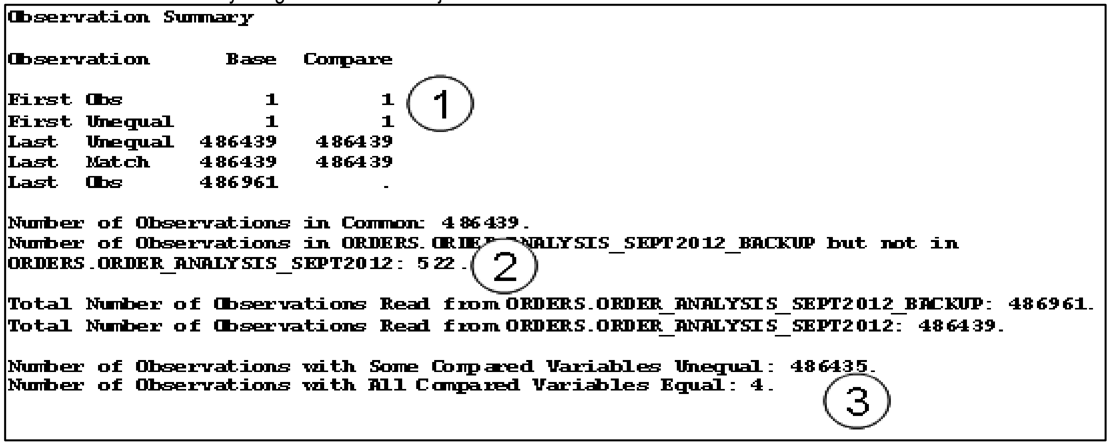


```
Variables Summary

Number of Variables in Common: 5.
Number of Variables with Differing Attributes: 1.


Listing of Common Variables with Differing Attributes

Variable     Dataset       Type  Length  Format

PROD_STATUS  ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP  Char       1
             ORDERS.ORDER_ANALYSIS_SEPT2012         Char       1   $1.
```

Figure 4: The Variables Summary

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.

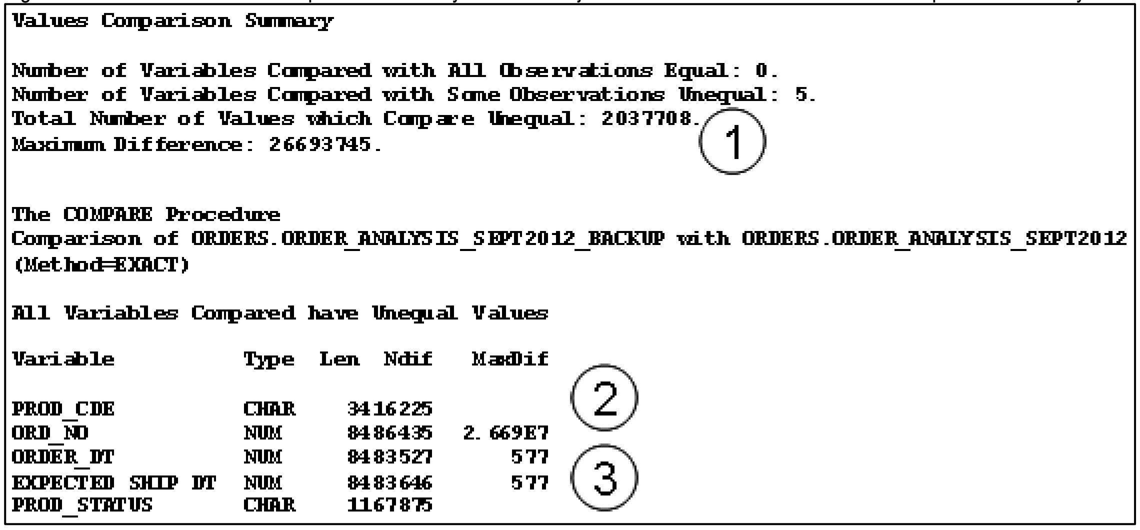


```
Observation Summary

Observation      Base   Compare

First Obs           1         1
First Unequal       1         1
Last  Unequal  486439    486439
Last  Match    486439    486439
Last  Obs      486961         .

Number of Observations in Common: 486439.
Number of Observations in ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP but not in
ORDERS.ORDER_ANALYSIS_SEPT2012: 522.

Total Number of Observations Read from ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP: 486961.
Total Number of Observations Read from ORDERS.ORDER_ANALYSIS_SEPT2012: 486439.

Number of Observations with Some Compared Variables Unequal: 486435.
Number of Observations with All Compared Variables Equal: 4.
```

Figure 5: the Observation Summary

1. The first inequality was found at the first observation
2. BASE has 522 few 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:

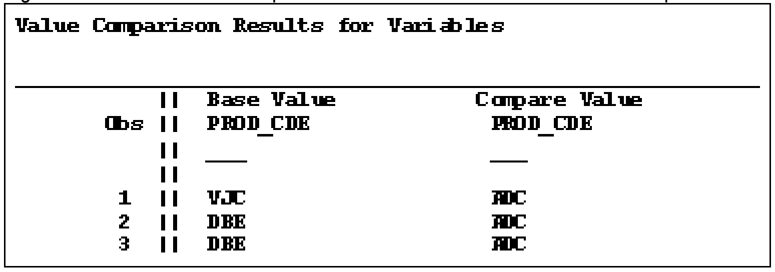


```
Values Comparison Summary

Number of Variables Compared with All Observations Equal: 0.
Number of Variables Compared with Some Observations Unequal: 5.
Total Number of Values which Compare Unequal: 2037708.
Maximum Difference: 26693745.


The COMPARE Procedure
Comparison of ORDERS.ORDER_ANALYSIS_SEPT2012_BACKUP with ORDERS.ORDER_ANALYSIS_SEPT2012
(Method=EXACT)

All Variables Compared have Unequal Values

Variable          Type  Len  Ndif    MaxDif

PROD_CDE          CHAR    3416225
ORD_NO            NUM     8486435  2.669E7
ORDER_DT          NUM     8483527      577
EXPECTED_SHIP_DT  NUM     8483646      577
PROD_STATUS       CHAR    1167875
```

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 MaxDif 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?



```
Value Comparison Results for Variables

_______________________________________________________________
           ||  Base Value           Compare Value
       Obs ||  PROD_CDE              PROD_CDE
           ||  ___                   ___
           ||
         1 ||  VJC                   ADC
         2 ||  DBE                   ADC
         3 ||  DBE                   ADC
```

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:

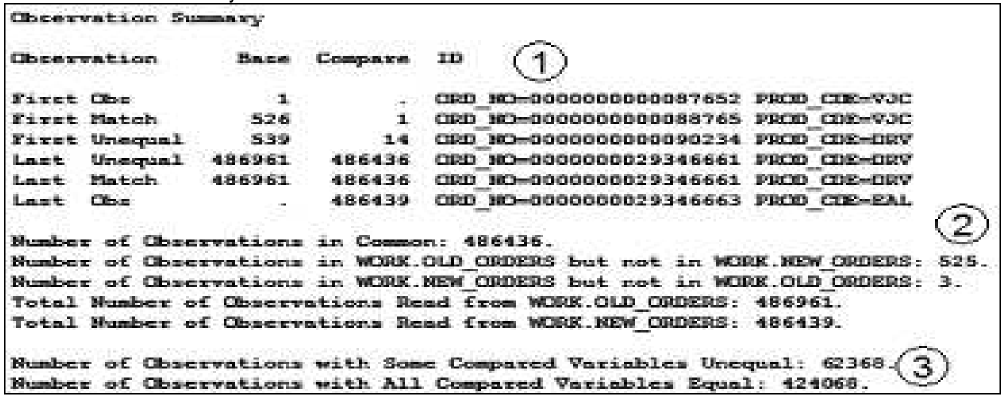
```
PROC COMPARE BASE=OLD_ORDERS COMPARE=NEW_ORDERS ;
ID ORD_NO PROD_CDE;
```

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.

```
Data set                    Created          Modified  NVar    NObs


WORK.OLD_ORDERS  14SEP12:14:13:57  14SEP12:14:13:57     5       .
WORK.NEW_ORDERS  14SEP12:14:13:57  14SEP12:14:13:57     5       .
```

Figure 8: Data Summary for ID Variables ORD_NO and PROD_CDE set up with views

The Observation Summary looks different too:

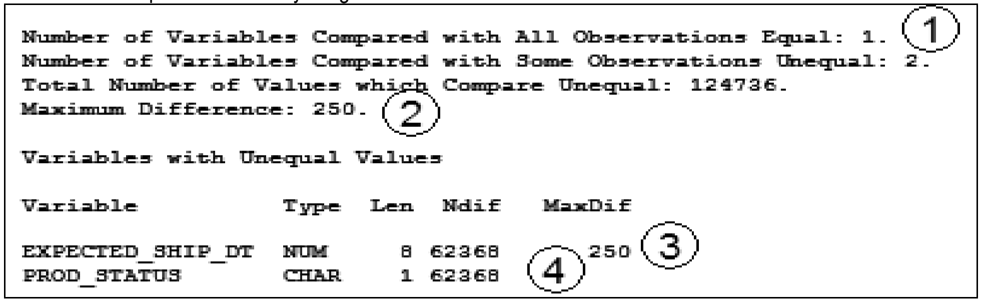


```
Observation Summary

Observation      Base  Compare  ID

First Obs           1        .  ORD_NO=0000000000087652 PROD_CDE=VJC
First Match       526        1  ORD_NO=0000000000088765 PROD_CDE=VJC
First Unequal     539       14  ORD_NO=0000000000090234 PROD_CDE=DRV
Last  Unequal  486961   486436  ORD_NO=0000000029346661 PROD_CDE=DRV
Last  Match    486961   486436  ORD_NO=0000000029346661 PROD_CDE=DRV
Last  Obs           .   486439  ORD_NO=0000000029346663 PROD_CDE=EAL

Number of Observations in Common: 486436.
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: 486961.
Total Number of Observations Read from WORK.NEW_ORDERS: 486439.

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

Figure 9: Observation Summary for ID Variables ORD_NO and PROD_CDE

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:

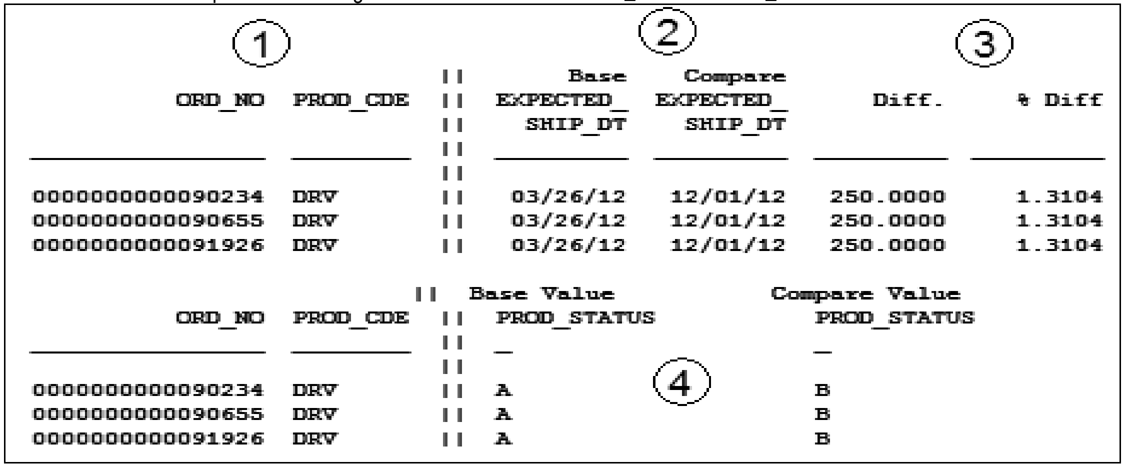


```
Number of Variables Compared with All Observations Equal: 1.
Number of Variables Compared with Some Observations Unequal: 2.
Total Number of Values which Compare Unequal: 124736.
Maximum Difference: 250.

Variables with Unequal Values

Variable           Type  Len  Ndif   MaxDif

EXPECTED_SHIP_DT   NUM     8 62368      250
PROD_STATUS        CHAR    1 62368
```

Figure 10: Values Comparison Summary for Matched Observations

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



```
                               ||        Base     Compare
          ORD_NO   PROD_CDE    ||  EXPECTED_   EXPECTED_         Diff.      % Diff
                               ||    SHIP_DT     SHIP_DT
________________   ________    ||  _________   _________     _________   _________
                               ||
0000000000090234   DRV         ||   03/26/12    12/01/12      250.0000      1.3104
0000000000090655   DRV         ||   03/26/12    12/01/12      250.0000      1.3104
0000000000091926   DRV         ||   03/26/12    12/01/12      250.0000      1.3104

                             ||  Base Value           Compare Value
          ORD_NO   PROD_CDE    ||  PROD_STATUS              PROD_STATUS
________________   ________    ||  _                        _
                               ||
0000000000090234   DRV         ||  A                        B
0000000000090655   DRV         ||  A                        B
0000000000091926   DRV         ||  A                        B
```

Figure 11: Variables with Unequal Values for ID variables ORD_NO and PROD_CDE

1. The ID variables ORD_NO and PROD_CDE 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

A MATTER OF METHODS

Can we avoid a comparison result like this? :

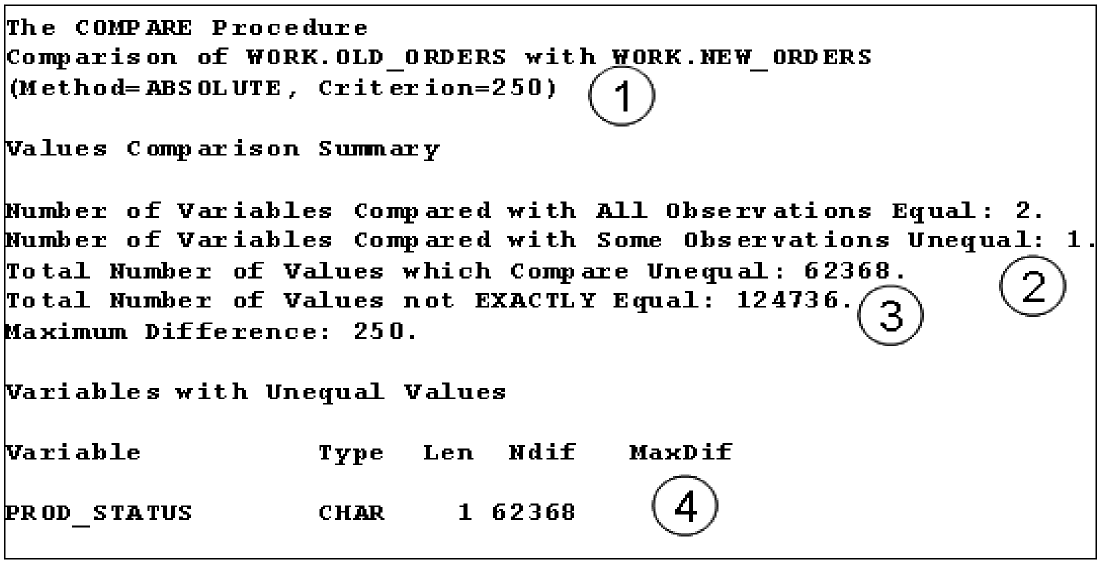
```
BASE      COMPARE          DIFF.      % DIFF.
508.0500  508.0500         -6.79E-15  -2.254E-15
```

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.

```
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) LISTCOMPOBS – 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) LISTOBS – all observations found in only one data set, h) LISTVAR – out all variables found in only 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)   1

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 2.
Number of Variables Compared with Some Observations Unequal: 1.
Total Number of Values which Compare Unequal: 62368.        2
Total Number of Values not EXACTLY Equal: 124736.  3
Maximum Difference: 250.

Variables with Unequal Values

Variable          Type  Len  Ndif   MaxDif

PROD_STATUS       CHAR    1 62368     4
```

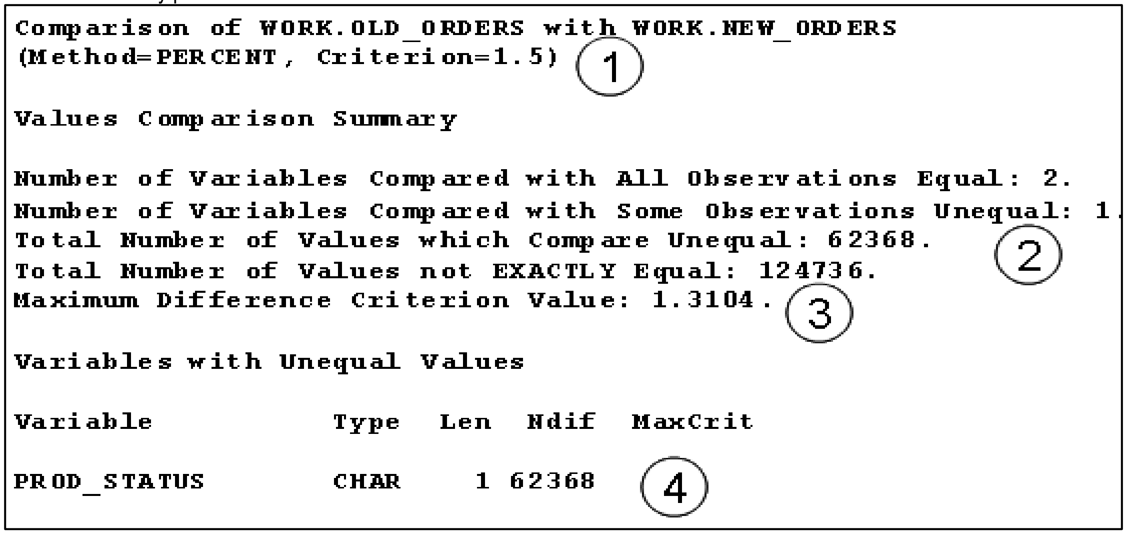
Figure 12: ABSOLUTE Method Values Comparison Summary

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:

```
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:



```
Comparison of WORK.OLD_ORDERS with WORK.NEW_ORDERS
(Method=PERCENT, Criterion=1.5)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 2.
Number of Variables Compared with Some Observations Unequal: 1
Total Number of Values which Compare Unequal: 62368.
Total Number of Values not EXACTLY Equal: 124736.
Maximum Difference Criterion Value: 1.3104.

Variables with Unequal Values

Variable           Type  Len  Ndif  MaxCrit

PROD_STATUS        CHAR    1 62368
```

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

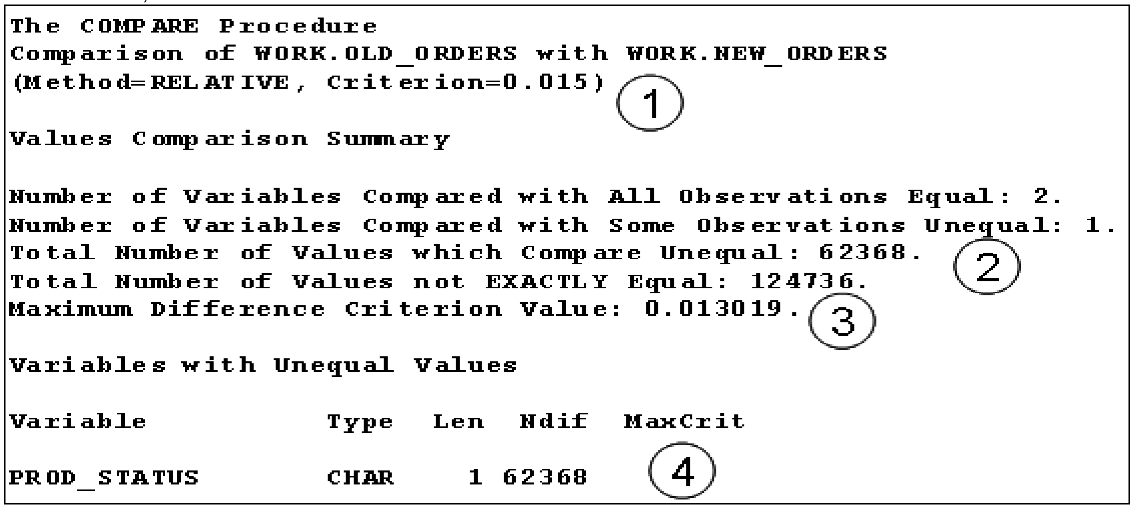
The last is the RELATIVE method which compares the absolute relative difference to the value specified by the criterion. The δ is the numeric precision of the system – the default is 0 which we'll use here.

$$\mathrm{ABS}\,(y-x)\,/\,((\mathrm{ABS}\,(x)+\mathrm{ABS}\,(y))\,/2+\delta)$$

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

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



```
The COMPARE Procedure
Comparison of WORK.OLD_ORDERS with WORK.NEW_ORDERS
(Method=RELATIVE, Criterion=0.015)

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 2.
Number of Variables Compared with Some Observations Unequal: 1.
Total Number of Values which Compare Unequal: 62368.
Total Number of Values not EXACTLY Equal: 124736.
Maximum Difference Criterion Value: 0.013019.

Variables with Unequal Values

Variable           Type  Len  Ndif  MaxCrit

PROD_STATUS        CHAR    1 62368
```

Figure 15: RELATIVE Method Values Comparison Summary

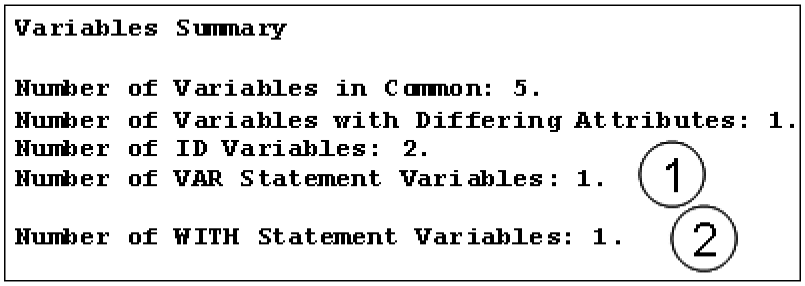
1. RELATIVE Method and Criterion are part of the heading
2. Down to just one variable with some unequal but-
3. Comparison notes the number comparing not exactly equal. Shows the maximum relative value which is below our criterion (0.015 vs. 0.013019)
4. No effect on PROD_STATUS or indeed any character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

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

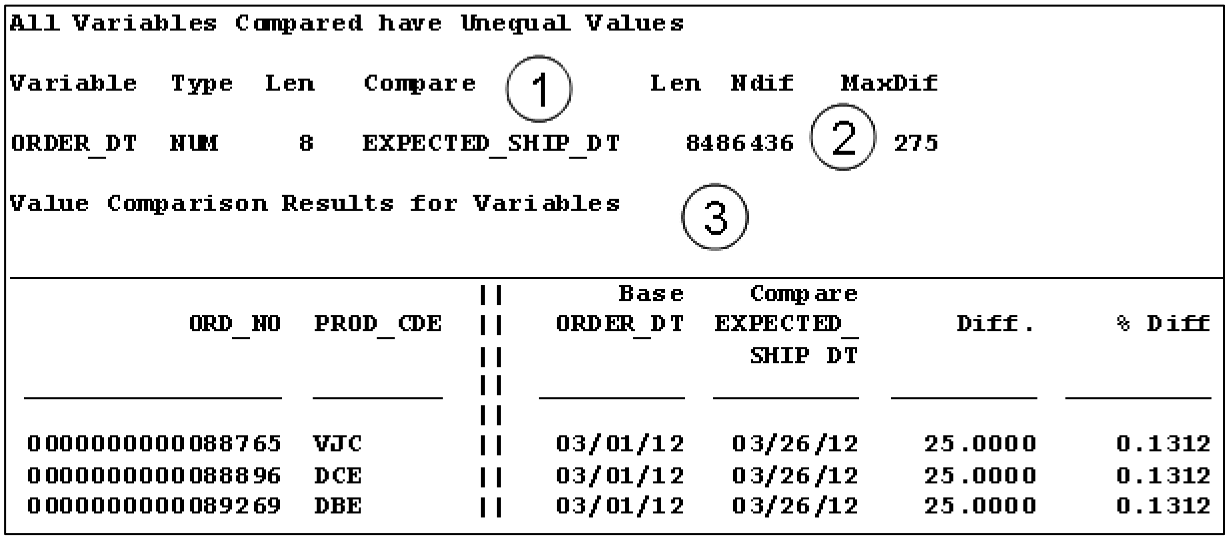
The above would compare 100% unequally as the comparison would be limited to ORDER_DT in BASE and EXPECTED_SHIP_DT in COMPARE. This would actually be our expected results and would show as in figures 16 and 17:



```
Variables Summary

Number of Variables in Common: 5.
Number of Variables with Differing Attributes: 1.
Number of ID Variables: 2.
Number of VAR Statement Variables: 1.

Number of WITH Statement Variables: 1.
```

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)



```
All Variables Compared have Unequal Values

Variable   Type  Len   Compare          Len  Ndif   MaxDif

ORDER_DT   NUM     8   EXPECTED_SHIP_DT    8486436  275

Value Comparison Results for Variables

                          ||        Base    Compare
           ORD_NO  PROD_CDE ||   ORDER_DT  EXPECTED_        Diff.      % Diff
                          ||               SHIP_DT
 ________________  ________ ||  _________  _________  ___________  __________
                          ||
 0000000000088765  VJC      ||   03/01/12   03/26/12      25.0000      0.1312
 0000000000088896  DCE      ||   03/01/12   03/26/12      25.0000      0.1312
 0000000000089269  DBE      ||   03/01/12   03/26/12      25.0000      0.1312
```

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.

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

We will see a new section: Comparison Results for Observations

```
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

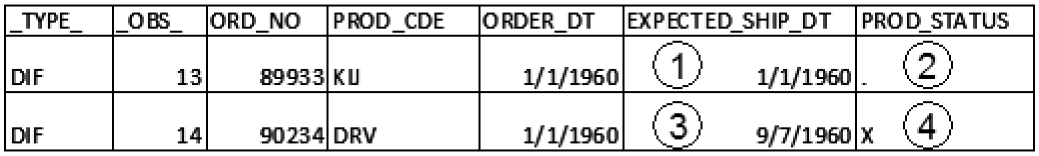
Instead of getting a listing of 32,000 or so, depending on your installation's default setting, we'll get only 100 for EXPECTED_SHIP_DT and PROD_STATUS each.

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.



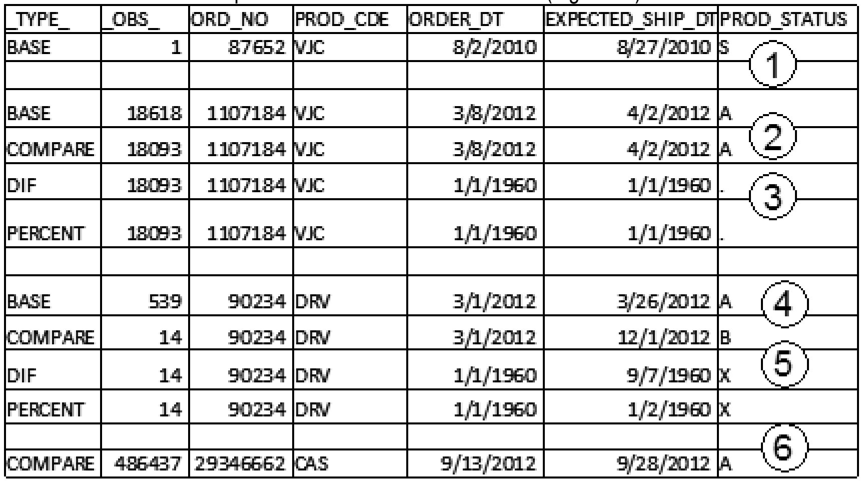
TYPE	_OBS_	ORD_NO	PROD_CDE	ORDER_DT	EXPECTED_SHIP_DT	PROD_STATUS
DIF	13	89933	KU	1/1/1960	① 1/1/1960	. ②
DIF	14	90234	DRV	1/1/1960	③ 9/7/1960	X ④

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):



TYPE	_OBS_	ORD_NO	PROD_CDE	ORDER_DT	EXPECTED_SHIP_DT	PROD_STATUS	
BASE	1	87652	VJC	8/2/2010	8/27/2010	S	1
BASE	18618	1107184	VJC	3/8/2012	4/2/2012	A	
COMPARE	18093	1107184	VJC	3/8/2012	4/2/2012	A	2
DIF	18093	1107184	VJC	1/1/1960	1/1/1960	.	3
PERCENT	18093	1107184	VJC	1/1/1960	1/1/1960	.	
BASE	539	90234	DRV	3/1/2012	3/26/2012	A	4
COMPARE	14	90234	DRV	3/1/2012	12/1/2012	B	
DIF	14	90234	DRV	1/1/1960	9/7/1960	X	5
PERCENT	14	90234	DRV	1/1/1960	1/2/1960	X	
							6
COMPARE	486437	29346662	CAS	9/13/2012	9/28/2012	A	

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:

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

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

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:

```
PROC COMPARE 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

VAR	_TYPE_	_BASE_	_COMP_	_DIF_	_PCTDIF_
ORDER_DT	N	486436	486436	486436	486436
ORDER_DT	MEAN	19150	19150.29	0	0
ORDER_DT	STD	56.17	56.16954	0	0
ORDER_DT	MAX	19248	19248	0	0
ORDER_DT	MIN	19053	19053	0	0
ORDER_DT	STDERR	0.0805	0.080536	0	0
ORDER_DT	T	237787	237786.6		
ORDER_DT	PROBT	0	0		
ORDER_DT	NDIF	0	0		
ORDER_DT	DIFMEANS	0	0	0	
ORDER_DT	R,RSQ	1	1		

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 Butkovich 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

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;
      build_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_='DIF') &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;
```