Paper 004-2007

HOW SUITE IT IS -
Taking Full Advantage of SAS® Seamless Integration with Microsoft's Office Suite

Michael Compson and John C. Hennessey
Office of Research Evaluation and Statistics,
Social Security Administration

ABSTRACT
The Office of Policy (OP) in the Social Security Administration (SSA) produces a series of publications containing programmatic statistical estimates. In any given year, OP produces approximately 1,425 tables, most of which are generated using Microsoft Excel (MS-EXCEL). In 2005, OP decided to undertake a major project to evaluate the consistency of its statistical estimates and all the accompanying formats and documentation of the tables. Given the sheer magnitude of the number of tables produced, it was decided to use SAS to create metadata for the most recently available tables. The cooperation between SAS and the Microsoft Office Suite allows users to move between SAS and various programs within the Suite. The ability to move between SAS, MS-EXCEL, and Microsoft ACCESS (MS-ACCESS), plus the strong technical support from SAS has allowed OP to systematically evaluate the consistency and accuracy of all aspects of their publications.

KEYWORDS AND PHRASES
- Enterprise Guide®
- Metadata
- Text Search
- Microsoft EXCEL
- Microsoft ACCESS
- SAS Microsoft-EXCEL Engines

1. INTRODUCTION
The Office of Policy (OP) produces a series of publications containing statistics related to the programs administered by Social Security Administration (SSA). Most of the publications are produced in hard copy and on the website in html and/or PDF/EXCEL formats, but a number of them are virtual publications found only on the SSA website. In addition, OP produces tables that are included in external publications (e.g., the Census Bureau's Statistical Abstract). Some of the tables are produced on a monthly basis; most of the publications are produced on an annual basis. In any given year, OP produces approximately 1,425 standard tables.

The programmatic statistics and the underlying data are a critical component of efforts by researchers and policy makers to understand and evaluate the current status and long-term trends of the SSA programs. Given the widespread use of the statistics, it is critical that the estimates and the underlying documentation are accurate and consistent both within and across all publications.

There have been a number of instances when the estimates produced by OP have been different both within and across publications. A leading explanation for the difference in estimates is the use of different samples to produce the same estimate (e.g., using a 1 percent sample for one estimate and a 10 percent sample for another). There has also been some concern regarding the number of publications produced and the possibility of redundancy in the estimates.

In addition, there is concern about the consistency, accuracy, and completeness of titles, row/column headers, footnotes, sources, etc. – the metadata about the tables. For example, some of the titles don't describe everything in the tables; footnotes for the same issue appear differently in several tables; data sources are not always complete.

In 2005, OP undertook a major initiative – the Data Management Strategic Plan (DMSP) Project - to review and document all aspects of its data management, security, and publication processes. One of the teams put together for the DMSP project was the Statistical Publication Review (SPR) Team whose primary goals were to:

1. evaluate the accuracy and consistency of the underlying documentation of the tables, (e.g., titles, footnotes, sources, notes and so forth),
2. evaluate the possibility of redundant estimates across publications, and
3. evaluate the accuracy and consistency of the estimates both within and across publications,

The keystone to achieving these goals was the development of a metadata base for the most recent publications/tables produced by OP. This paper is about the process undertaken to achieve the goals of the SPR and the leading role SAS played in that process. The ability of SAS to seamlessly interface with Microsoft's Office Suite - in particular MS-EXCEL and MS-ACCESS – eliminated the prospect of having to manually enter much of the metadata from approximately 992 tables.[1] For the most part, moving between SAS and the Microsoft programs was relatively straight forward. When technical issues did arise, and there were a variety of issues, the timely technical support from SAS was critical in moving the project forward.

We had originally hoped to use SAS code to develop a "polished" automated system to generate the metadata. However, we soon realized that given the tight time constraints, the magnitude of the project, and the diversity among the 992 tables, we had to settle on a workable approach to generating the metadata. Once the project is completed we will assess the potential to develop a fully automated system in the future and incorporate that assessment into our report about the present status of the tables. We estimate that SAS was able to generate approximately 85 percent of the data from the tables. The remaining 15 percent had to be entered into the metadata base manually.

2. PROJECT PARAMETERS

The first decision for the project was to choose the mechanism for collecting and storing all of the metadata from the nearly 1,000 tables included in our analysis. Two options were considered: (1) use MS-ACCESS to manually enter the metadata into MS-ACCESS tables, or (2) use SAS to capture the metadata directly from the MS-EXCEL files/spreadsheets. Given the sheer magnitude of the number of tables produced on an annual basis, the likelihood for data entry errors, and the potential future uses of the metadata beyond the DMSP project, it was decided to use SAS® Enterprise Guide® to create the metadata for all of the publications.

The cooperation between SAS and the Microsoft Office Suite allowed us to develop a five-step process to capture the legacy metadata:

1) Directly read most of the metadata from approximately 1,000 MS-EXCEL spreadsheets into SAS with each table representing a single row of data.
2) Use some of the powerful SAS text functions to parse the massive amount of text.
3) Directly convert the resulting SAS tables into an MS-ACCESS database.
4) Use the data entry forms in MS-ACCESS to edit and validate the predetermined values from SAS and to manually enter metadata not available from the MS-EXCEL tables.
5) Convert the resulting MS-ACCESS data base back to SAS to be used in the Enterprise Guide® point and click query system.

3. READING MS-EXCEL SPREADSHEETS INTO SAS

Using SAS, MS-EXCEL and MS-ACCESS interchangeably to capture, modify, verify, and query the legacy metadata for the most recent OP publications was a new experience for most of the members of SPR. The resulting process may seem cumbersome to more experienced SAS programmers, but given the time pressures to generate the metadata file and the wide range of technical expertise among team members, this approach became a necessity. The SSI Annual Statistical Report, 2004 served as a model publication because of the relatively large number of tables in the publication (61), the variation in the types of tables in the publication, and the fact that the tables were contained in 61 separate MS-EXCEL files. The plan was to start with a single publication and work our way through the five-step process discussed above. This allowed a test of the proposed process for capturing the metadata from start to finish.

The SAS MS-EXCEL Libname Engine allows SAS to directly read MS-EXCEL spreadsheets into SAS data tables. The tables in the SSI Annual Statistical Report, 2004 were contained in 61 separate MS-EXCEL workbooks. Table 2A in the appendix is an example of SAS macro code to read in all 61 MS-EXCEL workbooks at once and to create the corresponding SAS tables. This code was modified for the publications whose tables were contained in a single workbook with each of the tables in a separate worksheet.

1 Table 1A in the appendix lists all of the publications included in the analysis. The SPR team decided to focus on the major publications produced by OP and did not include several publications because they were being completely reformatted.

Using SAS to read MS-EXCEL files that are already structured as datasets can be challenging. Unfortunately the resulting SAS data tables derived from the metadata in the MS-EXCEL tables are not structured as a dataset. Specifically, the columns of the metadata do not represent a variable with its values in that column and the rows do not represent a data record. Instead, the cells contain metadata information in a haphazard way. For example, when reading in the metadata from Table 3A in the appendix, the information is not stored as a dataset. The first row contains the primary category for the table (Recipients of Social Security, SSI, or Both) in what appears to be a single column. However, the primary category is really right justified in a merged cell. If the merge cell option is removed, the primary category is found in the leftmost column of the string of columns that were merged. As a result, SAS places the primary category in the first column in the first row.

The table number and title is in the second row and first column of merged cells and is left justified. The third row contains the first level of column headers. For example, *Year* is actually in the first column in row three but appears to be in row 4 because it is at the bottom of string of cells that are merged vertically. The sources, notes, footnotes, etc. appear in merged cells below the row and column headers. SAS places the text in the leftmost column of the string of merged columns. For example, the text that follows “Sources” in Table 3A will appear in one column as a single string of characters.

Given the unstructured nature of the resulting SAS data tables, the process for transforming the data into usable metadata was incremental. Our strategy was to convert the spreadsheets to a SAS dataset and then search through all of the rows and columns of the dataset for metadata information. This search process is delineated into three distinct steps: (1) capture the relatively easy metadata via parsing, (2) capture the row headers separately, and (3) capture the column headers separately. At the end of the process, all three components of the metadata were combined into a single SAS data table for each publication, with one row per table.

PARSING TEXT USING SAS FUNCTIONS

After appending all of the records for each table into a single SAS data table, various SAS text functions were used to parse out basic information from the tables into separate variables, e.g. table number, table title, data source, footnotes, any notes in the table, and any contact information.

As luck would have it, most of this information appeared in the first column of the SAS dataset making this process relatively straightforward. When SAS pulled text from a merged cell, it placed all of the text in the left column of the merged cell. The first step in the process used either the SCAN, SUBSTR or INPUT(SUBSTR) functions to grab the desired block of text. In most cases it was not possible to directly isolate the exact text we were hoping to capture in a single step. As a result, subsequent use of the same functions further refined the parsing process until the desired result was achieved. The following provides an example of the coding used to parse the text captured from the MS-EXCEL tables.

Most of the tables contain a person contact using the following form:

CONTACT: John Doe (410) 000-0000 or ssi.abc@ssa.gov

The code for parsing the text into useable data was the following:

```
%let numparts=5;
length Contact1-contact&numparts  $600;
format Contact1-contact&numparts $600.;
array c {&numparts} $ contact1-contact&numparts;
do i = 1 to &numparts;
c{i} = scan(contact,i,';');
end;
drop i;

CONTACT1_NAME = (Left(SCAN(SCAN(Contact1,-2,'('),2,':')));
CONTACT1_PHONE = ('('||SCAN(SCAN(contact1,2,'('),1,'or'));
CONTACT1_EMAIL = (SCANQ(contact1,-1));
CONTACT2_NAME= (SCAN(Contact2,-2,'('));
CONTACT2_PHONE = ('('||SCAN(SCAN(contact2,2,'('),1,'or'));
CONTACT2_EMAIL = (SCANQ(contact2,-1));
```

This process became more complicated when parsing text to capture the Notes or footnotes contained in some of the tables. For example, some of the tables had up to 10 notes and several actually had a table as part of a note. In addition, many tables contained multiple footnotes (16 was the largest number) or no footnotes at all. There were

also instances when the notes and/or footnotes were extremely large. So, the length of the variables had to be set to 1024.

CAPTURING THE ROW AND COLUMN HEADERS

The second and third steps captured all row and column headers in each of the tables. This process turned out to be much more complicated relative to the metadata captured in step 1. The primary difficulty occurred with row or column headers that contained multiple levels of categories. This issue is highlighted using Table 3A in the appendix which has two levels of categories for the column headers. By convention, we assumed that the first column of each table was a row header. The top level column header for columns 2-5 is "Numbers" appears in a merged cell. The text in the merged cells is contained in the leftmost cell when converted to a SAS dataset. The secondary column heading for column 2 is "Total" and for column 3 is "Social Security only". Taken separately, the column headings would have little use in the process of determining what information is included in the table. The problem was how to generate a column header variable that incorporates both levels of categories and thus, could be useful for our purposes. A two-step process was used to resolve this problem. Recall that when SAS reads in the primary column header "Number" that text is placed in the leftmost column of the string of merged cells. The remaining cells within the string of merged cells are blank. In this example, there is no text in columns 3-5 at the primary level. The first step assigns the text in the leftmost column to each of the blank cells in the string of merged cells. The second step combines the text in the primary level with the text in the column directly below it (the secondary level). For each column, we combined the text for each level separated by a colon. For example, the third column of data (the first column is a row header) has the full title of

"Number: SSI only"

This process became even more complicated for the tables that had column headers with 3 or 4 levels. The code to fill in the additional information in the merged cells became a bit complex, but basically amounted to the same principle of filling in the same information across the merged cells in each row of the column headers. For the most part, the process of generating row headers was somewhat less complicated because fewer tables had multiple layers of row headers. Basically, the transpose of the procedure above worked for this case.

In terms of trying to construct row and column headers for all of the tables we are documenting, the biggest problem was to construct code which would assess where the row and column headers were and how many levels of categories each had. One of the issues we encountered in this process is illustrated in Table 3A of the appendix. In this case, the row headers turn out to be numbers, so it is hard to write a program that will know when it is in the row or column header and when it is in the tables entries. After some initial attempts with some success and with our deadline looming large, we abandoned that effort and manually grouped the tables into different families, depending on the structure of the row and column headers. We then manually set the parameters in the code and ran it against that family of tables. Sample code for some of the logic appears in Table 4A in the appendix.

We have combined all of the row headers and column headers into two separate variables that can be queried within Enterprise Guide®. For example, we will query the resulting column header variable to see if it contains age, sex, race or any of the other table classifications. Combining this query with other variables within the metadata, like table title and primary category, we will be able to determine exactly what information is contained in each and every table.

The process of generating the row and column headers created two distinct SAS tables for each publication. These tables were subsequently merged with the table containing the parsed metadata to create a single SAS data table containing all the usable metadata for each publication. As previously noted, time constraints forced us to develop a working approach to gather and analyze the metadata rather than invest more time into developing a much more efficiently designed data base. However, SAS was able to handle the 2,500 row header variables and the relatively large text strings contained in some of the footnotes and notes.

4. MODIFYING AND VERIFYING THE CAPTURED SAS METADATA

A critical aspect of creating the metadata is to verify, and where necessary modify, the metadata captured via SAS. In addition, there were a number of desired variables that could not be captured from the MS-EXCEL spreadsheet. For example, we are concerned whether the titles in the tables are inclusive, i.e., do they identify all of the information contained in the table? Since this information could not be culled directly from the tables, we created a number of variables with default values set to DK. We then relied on the judgment of analysts to enter the appropriate values for each of these variables.

Unfortunately the resulting spreadsheet view of the SAS data tables are unwieldy in terms of using them to verify, modify, and/or enter new data. Editing data which is displayed in spreadsheet form, as is the case in Enterprise Guide®, is awkward. For example, there is the classic problem of making sure that one stays on the correct record as one slides to the right to display more variables for the same record. Since the ability to convert the SAS Dataset

to an MS-ACCESS database is fairly simple, we turned to MS-ACCESS to create data entry forms that are very user friendly in a number of ways. First, the data entry forms are very easy to create and second, any changes that need to be made to correct the SAS table can be done right in the form.

The following code was used to convert the resulting SAS data tables to MS-ACCESS data tables:

```
Libname MS-ACCESS 'K:\Usrfiles\Publication-Team\Production\PRODUCTION-TABLES-
STORAGE-VS1.mdb' dbmax_text=5000;
Libname TABLES 'Q:\DMSP\Publication Review Team\Publication-Data\2005
Publications\METADATA';
DATA MS-ACCESS.SAS2MS-ACCESS_001;
SET TABLES.SSI_ANNUAL;
RUN;
```

The resulting MS-ACCESS tables for each of the publications were stored in a single MS-ACCESS database named: PRODUCTION-TABLES-STORAGE-VS1.mdb. This central location made it easier to import the appropriate table into the separate MS-ACCESS database created for each publication. Data entry forms were then created used to verify the metadata generated by SAS and to enter values for the variables that could not be culled from the original MS-EXCEL tables.

5. "LIFE IS NEVER SIMPLE" ISSUES:

As one would expect, we experienced a number of "technical" issues in the process of creating the metadata. The following discussion illustrates a number of these issues. The most fundamental problem we had was with the design of the overall database. A number of variables in the data base had multiple values. For example, some tables had 3 or 4 footnotes and others had more. Other tables which contain statistics by state and county have 2500 row headers. Because of the time constraint, we simply created one record for each table with the maximum number of footnotes, row headers, etc. that any table needed. A more sophisticated version would create relational data tables for each of these variables.

The first problem we faced occurred when attempting to read the MS-EXCEL spreadsheets into a SAS data table. We found that certain defaults in MS-EXCEL are set so that SAS would only read a limited number of characters in each cell. Some of our notes and footnotes were large. A phone call to SAS solved the problem. As shown in the code in table 2A in the appendix, we played with various combinations of options for header= mixed= scantext= and we declared the dbsatype= as stated in the set statement of the data step. By doing so, we captured all of the metadata into the SAS dataset.

We experienced several issues when converting the SAS data tables into MS-ACCESS tables and when converting back to SAS. The first problem was that some of the longer footnotes were getting cutoff in the process. As a result we had to go back and set the length for each of the variables. We originally attempted to use PROC EXPORT to convert the SAS data tables to MS-ACCESS tables. However, we got repeated errors regarding the length of the records when we tried to open the resulting table within MS-ACCESS. After consulting with SAS technicians, we used the code presented above.

Once the data was verified and all of the additional variables had been entered, the MS-ACCESS data tables were converted back to a single SAS data base to take full advantage of the power of the SAS Enterprise Guide® Query System. The point & click import feature of Enterprise Guide® did not work to convert the MS-ACCESS dataset into a SAS Dataset. Instead, within Enterprise Guide®, we opened a code window and used the libname engine to assign the MS-ACCESS dataset and then used PROC SQL code to read the data into a SAS dataset. Unfortunately this approach to converting to SAS data table does not support the replace option. If the same program is run more than once, you have to assign a different name for the resulting data base. An additional issue when converting from MS-ACCESS to SAS the password for the MS-EXCEL data base had to be turned off in order for SAS to get access to the data

6. USING SAS TO QUERY THE METADATA

We engaged in one final check of the SAS data base before proceeding with our analysis of the metadata. The check consisted of analyzing one way frequencies of each of the variables to make sure the values for each variable were appropriate. Inevitably, there were a number of instances where the values for some variables were not appropriate. For example, the variable for who had entered the data had a number of values for the same person (MC, Mc, mc, Mike, MIKE, and a missing value). In addition, there were several cases where names that appeared to have identical spelling were yielding distinct and separate values (for example, there were 75 observations for John Doe and another 15 observations for John Doe even though the names appeared to be identical). A first attempt to

correct this problem was to use code. However, the resulting code did not correct many of the instances of multiple values. Subsequent analysis revealed that some of the blank spaces in names were being treated as missing and that there was no systematic explanation for the location of the missing values. After attempting to use a number of SAS functions to remove leading and trailing blanks and several other we soon realized that Enterprise Guide® could be used to recode the values using the actual values for the variables. For example, the snapshot of the computed column window in Enterprise Guide ® in Table 5A of the appendix shows the various values entered for Mike C. Notice that the first value is missing. Enterprise Guide® point and click allowed us to use the actual values in the SAS data table to recode the variable and subsequently ensure that all of the values for each of the variables are appropriate.

The final SAS data set contains 992 observations (tables) and 2,624 variables. There were 2,500 row header variables, 62 column headers and 62 other variables. Initial attempts to use the data base on a personal computer were quite sluggish. As a result, we removed the 2,500 row header variables from the working data base knowing that when they were needed, they could be merged back into the data base very easily.

We are currently in the process of querying the metadata to systematically evaluate the accuracy and the consistency of the underlying documentation of the tables, (e.g., titles, footnotes, sources and so forth). Initial results reveal that our tables have 4 different types of title formats. We also found that nearly 30 percent of the tables in our analysis do not contain contact information. In addition, we discovered that only 15 percent of our tables had an email address that the public could use to contact staff with any questions. The next step after identifying inconsistencies in the formatting and presentation of the tables to make recommendations is to determine if there is a need to incorporate the actual estimates into the metadata to give us the ability to compare like estimates or do this manually.

7. CONCLUSION

The process of generating metadata for nearly 1,000 tables has been a long and arduous journey. However, given the magnitude and complexity of the project this was to be expected. It should be noted that prior to this undertaking, members of the SPR team had very little, if any, experience using SAS interactively with the various components of the Microsoft Office Suite. As a result, each step in the incremental development of the metadata was a learning process. The ability to query the metadata and systematically evaluate the consistency of the formatting, documentation, and accuracy of OP's publications has generated benefits well beyond the investment in time and resources. This process of creating the metadata has generated a strong working knowledge of the possibilities of interaction between SAS and Microsoft which bodes well for the continued development of the metadata and the potential for further extensions in the future.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the significant amount of work done on this project by Anne DeCesaro, Jeffrey Hemmeter, and Todd Williams, all staff members in the Office of Research, Evaluation and Statistics in the Social Security Administration.

CONTACT INFORMATION

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

Michael Compson
Office of Research, Evaluation, and Statistics
The Social Security Administration
6401 Security Blvd.
4-C-15 Operations Bldg.
Woodlawn, MD 21235
Work Phone: (410) 965-3949
Fax: (410) 966-4071
E-mail: Michael.Compson@ssa.gov

John Hennessey
Office of Research, Evaluation, and Statistics
The Social Security Administration
6401 Security Blvd.
4-C-15 Operations Bldg.
Woodlawn, MD 21235
Work Phone: (410) 965-0102
E-Mail: John.C.Hennessey@ssa.gov

APPENDIX TABLE 1A - PUBLICATIONS REVIEWED BY THE STATISTICAL PUBLICATION REVIEW TEAM

1. Annual Statistical Report on the Social Security Disability Insurance Program
2. Annual Statistical Supplement, 2004 to the Social Security Bulletin
3. Children Receiving SSI, 2004
4. Congressional Statistics, December 2004
5. Earnings and Employment Data for Workers Covered under Social Security and Medicare, by State and County, 2002
6. OASDI Beneficiaries by State and County, 2004
7. OASDI Beneficiaries by State and ZIP Code, 2004
8. OASDI Monthly Statistics, October 2005
9. SSI Annual Statistical Report, 2004
10. SSI Disabled Recipients Who Work, 2004
11. SSI Monthly Statistics, August 2005
12. SSI Recipients by State and County, 2004

APPENDIX TABLE 2A - EXAMPLE OF SAS CODE USED TO CONVERT 61 SEPARATE MS-EXCEL SPREADSHEETS INTO SAS DATA TABLES

***MS-EXCEL2SAS PROGRAM;**

```
   libname SSI_ANN 'Q:\DMSP\Publication Review Team\SAS_Projects\SSI_ANNUAL';
   *Need to set the number of tables in the publication;
   %macro SSIANN;
   %DO K=1 %TO 61;

   Libname Table excel %unquote(%bquote('Q:\DMSP\Publication Review Team\Publication-
   Data\SSI\SSI-Annual\table&K..XLS'))
        Header=NO Mixed=YES  scantext=NO;

   /*changed the mixed option to MIXED=NO to get rid of ? and a few other characters that was
   chopping short note1       in table 31.  Unfortunately, that option resulted in missing values
   for tables 19, 54, 55, 56*/

   Data SSI_ANN.T&K ;
        Set table."table &K$"n (DBSASTYPE=(f1='char(1024)')) End=T1;

        Length Title Table Source Contact Footnote_a Footnote_b Footnote_c Footnote_d
        Footnote_e Footnote_f Footnote_g Footnote_h Footnote_i Footnote_j
        Footnote_k Footnote_l Footnote_m Note1-Note10 $ 1024;

        RETAIN Title Table Source Contact Footnote_a Footnote_b Footnote_c Footnote_d
        Footnote_e Footnote_f Footnote_g Footnote_h Footnote_i Footnote_j
        Footnote_k Footnote_l Footnote_m Note1-Note10 Count;

        KEEP Title Table Source Contact Footnote_a Footnote_b Footnote_c Footnote_d
        Footnote_e Footnote_f Footnote_g Footnote_h Footnote_i Footnote_j
        Footnote_k Footnote_l Footnote_m Note1-Note10;


        IF _N_ = 1 Then Title = F1;
        IF _N_ =1 THEN COUNT=0;
        IF COUNT (F1,'Table') NE 0 Then Table = F1;
        IF COUNT(F1,'SOURCE') NE 0 Then Source=F1;
        IF COUNT(F1, 'CONTACT') NE 0 Then Contact =F1;
        IF F1 = 'a.' Then Footnote_a = F2;
        IF F1 = 'b.' Then Footnote_b = F2;
        IF F1 = 'c.' Then Footnote_c = F2;
        IF F1 = 'd.' Then Footnote_d = F2;
        IF F1 = 'e.' Then Footnote_e = F2;
        IF F1 = 'f.' Then Footnote_f = F2;
        IF F1 = 'g.' Then Footnote_g = F2;
        IF F1 = 'h.' Then Footnote_h = F2;
        IF F1 = 'i.' Then Footnote_i = F2;
        IF F1 = 'j.' Then Footnote_j = F2;
        IF F1 = 'k.' Then Footnote_k = F2;
        IF F1 = 'l.' Then Footnote_l = F2;
        IF F1 = 'm.' Then Footnote_m = F2;

        Array Note [10] $ Note1-Note10;

        IF COUNT(F1,'NOTE') NE 0 THEN COUNT =1;
        IF COUNT GT 0 & (COUNT(F1, 'a.') = 0 & COUNT(F1, 'CONTACT') = 0)
               THEN COUNT=COUNT + 1;
Example of SAS Code Continued –

        ELSE COUNT = 0;
        IF COUNT GT 0 THEN NOTE(COUNT-1)=F1;
        IF T1 Then Output;

RUN;

%END;
%MEND;
%SSIANN;
```

APPENDIX TABLE 3A - EXAMPLE OF TYPICAL TABLE PUBLISHED BY THE OFFICE OF POLICY, SOCIAL SECURITY ADMINISTRATION

Recipients of Social Security, SSI, or Both

Table 17.
Persons aged 18–64 receiving benefits on the basis of disability and their total and average monthly payments, December 1996–2004

Year	Number				Total payments (millions of dollars)			Average monthly payment [a] (dollars)		
	Total	Social Security only	SSI only	Both Social Security and SSI	Social Security only	SSI only	Both Social Security and SSI	Social Security only	SSI only	Both Social Security and SSI
1996	7,689,664	4,122,152	2,559,750	1,007,762	3,072	1,222	584	744.60	456.00	546.90
1997	7,811,748	4,250,155	2,550,105	1,011,488	3,245	1,257	604	762.80	458.10	557.10
1998	8,086,259	4,440,264	2,618,615	1,027,380	3,444	1,313	622	775.00	467.90	564.30
1999	8,399,309	4,703,774	2,650,586	1,044,949	3,691	1,346	643	784.10	477.60	576.70
2000	8,599,465	4,850,835	2,690,446	1,058,184	3,975	1,408	675	818.80	489.00	594.90
2001	8,791,338	4,979,844	2,732,020	1,079,474	4,299	1,491	719	862.60	506.80	615.20
2002	9,106,014	5,228,262	2,768,782	1,108,970	4,629	1,544	747	884.60	522.50	625.20
2003	9,445,573	5,492,325	2,811,647	1,141,601	5,024	1,603	790	914.10	533.50	638.20
2004	9,773,201	5,756,093	2,850,815	1,166,293	5,464	1,686	829	947.80	545.90	655.20

SOURCES: Social Security Administration, Disabled Beneficiaries and Dependents Master Beneficiary Record file and the Supplemental Security Record (Characteristic Extract Record format), 100 percent data.

NOTE: Social Security counts include disabled workers, disabled widow(er)s, and disabled adult children. SSI counts include recipients of federal SSI, federally administered state supplementation, or both.

a. Averages are not obtained simply by dividing the total dollars by the number of recipients. Averages exclude payments made in the current month for prior-month eligibility, such as back pay for new awards, so that large retroactive payments do not distort the averages.

CONTACT: Art Kahn (410) 965-0186 or ssi.asr@ssa.gov.

File available from:
Social Security Administration, Office of Policy
SSI Annual Statistical Report, 2004
http://www.socialsecurity.gov/policy/docs/statcomps/ssi_asr/2004/

APPENDIX TABLE 4A - SAMPLE CODE TO GENERATE COLUMN AND ROW HEADER VARIABLES

```
Data ssiann.Tst2_&K (keep= pub tableno columnhead1-columnhead25);
        set work.Tst&K end=t1;
        array field {*} _all_ ;
        retain rowcol i pub tableno;
        if _N_=1 then rowheaderflag=0;
        rowcombo=' ';
        Pub = 'SSI_Annual';
        TableNo=&k;

        if _N_ < 3 then do;
                delete;
                return;
        end;
        if _N_=3 then do;
                i = 2;
                if field(2) = ' ' then do until (field(i) ^= '  ' or i=&fvars);
                        i = i+1;
                end;
                rowcol = i;
                rowmax=i-1;
                call symput('rowmax',rowmax);
        end;

        /* Create the Column Headers - Eventually convert to macro variables   */

        length ch1-ch50    $  1024;
        array colhd {3:4, 1:25}   ch1-ch50;
        length columnhead1 - columnhead25 $ 1024;
        retain ch: columnhead: rowheaderflag;
        array columnhead {1:25} columnhead1 - columnhead25;
        if _N_=3 then do;
                j=3;
                do k=rowcol to &fvars;
                        if field(k) = ' ' then colhd(j,k)=colhd(j,k-1);
                        else colhd(j,k)=field(k);
                end;
        end;
        if _N_=4 then do;
                j=4;
                do k = rowcol to &fvars;
                        if field(k) = ' ' and colhd(j-1,k)=colhd(j-1,k-1)
                                then colhd(j,k)=colhd(j,k-1);
                        else colhd(j,k)=field(k);
                end;
        end;

        if _N_ =5 then do;
                do k = rowcol to &fvars;
                        substr(colhd(3,k),anycntrl(colhd(3,k)),1)= ' ';
                        substr(colhd(4,k),anycntrl(colhd(4,k)),1)= ' ';
                        columnhead(k) =trim(colhd(3,k))||':'||trim(colhd(4,k));
                        if substr(columnhead(k),length(columnhead(k)),1) = ':' then
substr(columnhead(k),length(columnhead(k)),1)=' ';
                        if k > rowcol then do;
                                if ((columnhead(k) = columnhead(k-1)) or (columnhead(k-1)=' ' )) then columnhead(k) = ' ';
                        end;
                end;
                do i=1 to (rowcol-1);
                                rowcombo = trim(left(rowcombo||trim(left(field(i)))));
                end;
                if trim(rowcombo)=' ' and rowheaderflag=0 then rowheaderflag=_N_;
                output;
                *return;
        end;
```

```
        if _N_>5 then do;
                do i=1 to (rowcol-1);
                        rowcombo = trim(left(rowcombo||trim(left(field(i)))));
                end;
                if trim(rowcombo) = '  ' and rowheaderflag=0 then do;
                        rowheaderflag=_N_;
                        call symput('rowheaderbottom',rowheaderflag);
                end;
        end;
run;
```

APPENDIX TABLE 5A- SNAPSHOT OF COMPUTED COLUMN RECODE OPTION IN ENTERPRISE GUIDE®