Basic SAS PROCedures for Generating Quick Results

Kirk Paul Lafler, Software Intelligence Corporation

Abstract
As an IT professional, saving time is critical. So is delivering timely and quality looking reports to management, end users, and customers. The SAS System provides numerous “canned” PROCedures for generating quick results to take care of both... and more. Attendees will see how basic SAS PROCedures such as SORT, PRINT, SQL, and FORMS are used to create detail reports; how CHART, FREQ, MEANS, PLOT, and UNIVARIATE are used to summarize and create graphical, tabular, or statistical output; and several useful techniques including how to inform the SAS System which data set to use as input to a procedure, how to subset data using a WHERE statement (or WHERE= clause), and how to perform BY-group processing to separate data into groups of like information.

Introduction
Once data has been collected and stored in a SAS data set, results can be produced quickly using one or more procedures. The SAS System provides numerous ready-to-use procedures designed for data analysis and presentation. Procedures are designed to be simple to use, and are what differentiate SAS from other software products. SAS’ built-in procedures offer users with a unique ability to generate quick results – requiring little, if any, programming skills. Using a procedure, or PROC, is similar to filling out a simple request form. By specifying the name of procedure and one or more options, you can produce results quickly and easily.

Procedures frequently write their results to the Output window (in SAS Display Manager), an output SAS data set, or an output file. When output is produced, it is often customized so it satisfies certain requirements such as automatic centering, printing or displaying dates and page numbers, and so on. Having the ability to customize the way output appears as well as the type of information that is produced is what makes procedures an indispensable tool for users everywhere.

SAS supports four categories of procedures: 1) reporting, 2) statistical, 3) scoring, and 4) utility. This paper investigates the use of several base-SAS procedures to enable the production of quick and useful reports, statistics, and tables of data, and will also look at procedures that can be used to perform simple data set management tasks.

The Anatomy of a PROC
Each procedure (or PROC) has unique characteristics and elements, but many common ones too. Each PROC consists of a keyword, one or more statements, and options – of which some are required and others are basically — optional. Although the statements and options vary from PROC to PROC, the basic anatomy of a PROC looks something like the following:

```
PROC procname DATA= __________________;
  TITLE ______________________________;
  FOOTNOTE __________________________;
  BY _________________________________;
  LABEL ______________________________;
  FORMAT _____________________________;
RUN; <or> QUIT;
```

The PROC Statement
Every PROC begins with the keyword PROC and is a required element. This keyword signals to the SAS supervisor (the internal traffic cop that controls everything that goes on in the SAS System) that a “canned” procedure is being launched, and not a DATA step or MACRO program.

The name of the PROC follows the keyword PROC – our example above references the anatomy PROC. In its simplest and purest form this is all that is required by a few procedures to run. As you might expect though, the results may also tend to be basic and output will appear without any customizations.

Once the name of the PROC is specified, you may then specify one or more options available with the PROC, and in any order. In our basic skeletal example above, the DATA= option appears. This option, if specified, informs the SAS System what data set to use as input to the PROC. If omitted, it automatically defaults to the most recently created data set – which may not be the most data set most recently used. (Readers will see several PROCs and their various options on the following pages).

TITLE and FOOTNOTE Statements
TITLE and FOOTNOTE statements are considered to be global statements and can generally be used universally throughout the SAS System (e.g., PROCs and DATA steps). A maximum of ten TITLE and FOOTNOTE statements can be specified in any PROC. TITLE statements, when specified, appear at the top of each output page and FOOTNOTE statements, if present, produce output at the bottom of each page of output. Readers are cautioned to use care when specifying TITLE and FOOTNOTE...
statements since they reduce the available space for printing detail lines.

**BY Statement**
A BY statement is optional in all PROCs except the SORT procedure. A BY statement in PROC SORT tells SAS what the order or arrangement should be for observations in a data set. A BY statement in any other PROC informs SAS to perform a separate analysis on the values in each BY group opposed to one large group. The data must have been sorted before it can be used in a reporting procedure.

**LABEL Statement**
A LABEL statement is also optional, and if present allows a more descriptive label to be assigned as variable (or column) headings. If omitted, SAS uses the variable names as column headers on output. When a LABEL statement is used in a PROC the assigned descriptive labels are only available for duration of the PROC step, and are not saved with the data set.

**FORMAT Statement**
A FORMAT statement is an optional statement that, when used, tells SAS to display information on output in a designated way. For example, you could have a date value displayed or written using a `mm/dd/yyyy` form such as 08/20/2001 or a `Month dd, yyyy` form such as August 20, 2001 to enhance readability. In the absence of a FORMAT statement, data is displayed using an internal date offset (the number of days from January 1, 1960) or a user-defined date format stored as part of the data set.

**RUN or QUIT Statement**
A RUN or QUIT statement tells SAS to terminate the PROC step before executing the next step in a program. A RUN statement is normally specified to designate an end to a non-interactive procedure like PROC PRINT, whereas a QUIT statement is specified to terminate an interactive procedure such as PROC SQL (more will be said about interactive procedures later). Although not required statements, specifying a RUN or QUIT statement can provide modest CPU improvements since the SAS Supervisor knows when one step ends and another begins.

**SAS Reporting Procedures**
So you may be wondering how the SAS System and its many PROCs can help with your reporting needs. The answer is simple. SAS and its many “canned, ready-to-go” PROCs turn information into structured and meaningful reports. More than 3.5 million software customers in 115 countries, and 98% of the Fortune 100 companies and 90% of the Fortune 500 companies can attest to the fact that producing detailed reports with the SAS System has never been easier.

**SAS and its many “canned, ready-to-go” PROCs turn information into structured and meaningful reports.**

Being able to put a powerful reporting tool in the hands of so many users, and not only programmers, is also an asset – not a liability. Having said this – it remains critical that an organization maintain a controlled environment to ensure information being dispensed in any report does not violate privacy and

---

**Description of Data Used in Reports**
The examples in this paper reference a data set containing a compilation of popular movies that I have watched over the years. This Movies data set consists of 22 observations, six variables, and contains the following data sorted in ascending order by the Movie Rating.

<table>
<thead>
<tr>
<th>Title</th>
<th>Length</th>
<th>Genre</th>
<th>Year</th>
<th>Studio</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxi Driver</td>
<td>101</td>
<td>Action</td>
<td>1995</td>
<td>MGM</td>
<td>D</td>
</tr>
<tr>
<td>Good Will</td>
<td>101</td>
<td>Drama</td>
<td>1999</td>
<td>Lions Gate</td>
<td>D</td>
</tr>
<tr>
<td>Robin Hood</td>
<td>101</td>
<td>Adventure</td>
<td>1975</td>
<td>Universal Studios</td>
<td>D</td>
</tr>
<tr>
<td>Beauty</td>
<td>101</td>
<td>ActionAdventure</td>
<td>1976</td>
<td>MGM</td>
<td>D</td>
</tr>
<tr>
<td>The Longest Yard</td>
<td>101</td>
<td>Action Sport</td>
<td>1938</td>
<td>Warner Bros</td>
<td>D</td>
</tr>
</tbody>
</table>
| The Flying
| 101    | Action Sport | 1937 | Warner Bros | D      |
| The Adventures of<br>Young<br>Men | 101 | Action Adventure | 1932 | Warner Bros | D      |
| National lampoon's<br>Holidays<br>hope | 101 | Comedy | 1982 | Voyage Brothers | D      |
| Christmas Vacation | 101 | Comedy | 1983 | Voyage Brothers | D      |
| Home for the holidays | 101 | Adventure Sport | 1936 | Warner Bros | D      |
| The Great Divide | 101 | Documentary | 1915 | Warner Bros | D      |
| The Conqueror | 101 | War | 1927 | Paramount Pictures | D      |
| North by Northwest | 101 | Adventure Sport | 1959 | Paramount Pictures | D      |
| The Great<br>McLintock | 101 | Comedy | 1961 | Paramount Pictures | D      |
| Witness for the<br>People | 101 | Drama | 1961 | Paramount Pictures | D      |
| The Tenderloin | 101 | Action Sport | 1928 | Warner Bros | D      |
| The Godfather | 101 | Crime, Drama | 1972 | Paramount Pictures | D      |
| The Godfather Part II | 101 | Crime, Drama | 1974 | Paramount Pictures | D      |
| The Godfather Part III | 101 | Crime, Drama | 1976 | Paramount Pictures | D      |
security issues. It is also critical to maintain a high level of accuracy in all reported information.

SAS reporting PROCs consist of a broad range of easy-to-use report formats. Because output is frequently requested in a variety of formats to satisfy a vast number of requirements, SAS is ready and able to help by bundling base-SAS PROCs that produce reports in the following formats:

1) **Detail** – prints one or more observations without collapsing data that meet specific report criteria.

2) **Summary** – collapses and prints information.

3) **Tabular** – collapses and prints information with borders.

4) **Statistical** – computes and prints descriptive statistics.

5) **Graphical** – prints information as simple line-oriented bar and pie charts, and line plots.

An alphabetical list of several SAS reporting PROCs and their output formats is illustrated in table 1 below.

<table>
<thead>
<tr>
<th>Report Output Style</th>
<th>PROC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detail Output</strong></td>
<td>FORMS PRINT REPORT SQL</td>
</tr>
<tr>
<td><strong>Summary Output</strong></td>
<td>CHART FREQ MEANS PLOT SQL UNIVARIATE</td>
</tr>
<tr>
<td><strong>Tabular Output</strong></td>
<td>FREQ TABULATE MEANS SQL UNIVARIATE</td>
</tr>
<tr>
<td><strong>Statistical Output</strong></td>
<td>MEANS SQL UNIVARIATE</td>
</tr>
<tr>
<td><strong>Graphical Output</strong></td>
<td>CHART PLOT</td>
</tr>
</tbody>
</table>

### Table 1. PROCs and Report Formats

#### PROCs for All That Detail

Sometimes a report must show all the detail it can. When this is the case, SAS provides the PRINT, REPORT, and SQL procedures to generate detail reports and the FORMS procedure to produce repetitive forms and labels. Although each PROC is relatively easy to use, they can also provide the level of support needed by even the most demanding programmer. By using one or more statements and options each PROC can produce simple to semi-custom reports.

### Using PROC PRINT

The PRINT procedure is a popular reporting tool that is used by users everywhere. In its simplest form, PROC PRINT prints all variables for all observations in a data set. The SAS System writes a default title line at the top of each report page automatically.

Suppose you had to create a report containing all Movie data (all observations and variables) with the littlest amount of code possible. The PROC PRINT statement illustrated below is about as simple as it gets – would produce a detail-oriented report consisting of all observations and variables.

**Procedure Code:**

```sas
PROC PRINT DATA=SSU.MOVIES;
RUN;
```

When the PROC PRINT code is executed, the SAS System applies certain defaults in creating report output including the default title, the number of observations, and list of variables.

#### Results:

Let's take it a step further. Suppose you did not want all the variables from the Movies data set – say you wanted only the Movie Title, Length, and Rating in that order. You could start with the PROC PRINT code from the previous example and insert a VAR statement between the PROC and RUN statements to tell SAS what variables to output along with their specific order on the report.

**Procedure Code:**

```sas
PROC PRINT DATA=SSU.MOVIES;
VAR TITLE LENGTH RATING;
RUN;
```
Although the PROC PRINT code from the previous example printed the three variables you asked for, it also printed the Observation number as column one on the report. The observation number is automatically displayed as the first column on all PROC PRINT output—but can be suppressed by specifying the NOOBS option. Suppose you wanted to remove the observation column and change the current title appearing at the top of the report.

**Procedure Code:**
```
PROC PRINT DATA=SSU.MOVIES NOOBS;
TITLE 'Movie Classics';
VAR TITLE LENGTH RATING;
RUN;
```

**Results:**
```
Obs Title Length Rating
1 The Wizard of Oz 101 G
2 Casablanca 103 PG
3 Jaws 125 PG
4 Rocky 120 PG
5 Star Wars 124 PG
6 Poltergeist 115 PG
7 The Hunt for Red October 105 PG
8 National Lampoon’s Vacation 98 PG-13
9 Christmas Vacation 97 PG-13
10 Ghost 127 PG-13
11 Jurassic Park 127 PG-13
12 Forrest Gump 142 PG-13
13 Michael 106 PG-13
14 Titanic 106 PG-13
15 Dressed to Kill 105 R
16 The Terminator 108 R
17 The Terminator 108 R
18 Lethal Weapon 110 R
19 Coming to America 115 R
20 Silence of the Lambs 115 R
21 Dracula 129 R
22 Braveheart 177 R
```

Let’s make one last change to the PROC PRINT code. Suppose you wanted to compute a total number of minutes for all movies in a rating group (e.g., G, PG, PG-13, and R). Since the Movies data set was originally sorted in ascending order by Rating, a BY statement can be specified with the Rating variable. A SUM statement is used to compute the total number of minutes for all movies in a By-group.

**Procedure Code:**
```
PROC PRINT DATA=SSU.MOVIES NOOBS;
TITLE 'Movie Classics';
BY RATING;
VAR TITLE LENGTH RATING;
SUM LENGTH;
RUN;
```

**Results (Partial Output – Excludes R Ratings):**
```
Movie Classics
Title Length Rating
The Wizard of Oz 101 G
Casablanca 103 PG
Jaws 125 PG
Rocky 120 PG
Star Wars 124 PG
Poltergeist 115 PG
The Hunt for Red October 105 PG
National Lampoon’s Vacation 98 PG-13
Christmas Vacation 97 PG-13
Ghost 127 PG-13
Jurassic Park 127 PG-13
Forrest Gump 142 PG-13
Michael 106 PG-13
Titanic 106 PG-13
Dressed to Kill 105 R
The Terminator 108 R
Lethal Weapon 110 R
Coming to America 115 R
Silence of the Lambs 115 R
Dracula 129 R
Braveheart 177 R
```

It is frequently necessary to subset the rows of data generated on output with a WHERE statement. Suppose you wanted to generate a report on PG and PG-13 rated movies.

**Procedure Code:**
```
PROC PRINT DATA=SSU.MOVIES NOOBS;
TITLE 'Movie Classics';
WHERE RATING IN ('PG','PG-13');
RUN;
```

**Results:**
```
Movie Classics
Title Length Rating
Casablanca 103 PG
Jaws 125 PG
Rocky 120 PG
Star Wars 124 PG
Poltergeist 115 PG
The Hunt for Red October 105 PG
National Lampoon’s Vacation 98 PG-13
Christmas Vacation 97 PG-13
Ghost 127 PG-13
Jurassic Park 127 PG-13
Forrest Gump 142 PG-13
Michael 106 PG-13
Titanic 106 PG-13
```

Using PROC SQL
The SQL procedure is known as the Structured (or Standard) Query Language and is a popular reporting tool among database users (e.g., SAS, Oracle, IBM,
etc.). In its simplest form, PROC SQL prints all variables (or columns) for all observations (or rows) in a data set (or table). As was illustrated with PROC PRINT, the SAS System writes a default title line at the top of each report page automatically. One or more TITLE statements can be specified to customize the title at the top of each page.

Besides using different statement syntax, an obvious distinction between PROC PRINT and PROC SQL is that a QUIT statement is specified for the latter, rather than a RUN statement, to terminate processing. The QUIT statement is used with interactive procedures.

Suppose you had to create a report containing all Movie data (all observations and variables) with the smallest amount of code possible. The PROC SQL statement illustrated below produces a detail-oriented report (similar to PROC PRINT) consisting of all observations and variables.

**Procedure Code:**

```sql
PROC SQL;
  TITLE 'Movie Classics';
  SELECT * FROM SSU.MOVIES;
QUIT;
```

When the PROC SQL code is executed, the SAS System applies certain defaults in creating report output including the default title, the number of observations, and list of variables.

**Results:**

<table>
<thead>
<tr>
<th>Movie Classics</th>
<th>Length</th>
<th>Category</th>
<th>Year Studio</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Wizard of Oz</td>
<td>151</td>
<td>Adventure</td>
<td>1939</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>Casablanca</td>
<td>103</td>
<td>Drama</td>
<td>1942</td>
<td>RKO</td>
</tr>
<tr>
<td>Gone</td>
<td>153</td>
<td>Action Adventure</td>
<td>1937</td>
<td>Universal Pictures</td>
</tr>
<tr>
<td>Rocky</td>
<td>129</td>
<td>Action Adventure</td>
<td>1976</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>Star Wars</td>
<td>134</td>
<td>Action Sci-Fi</td>
<td>1977</td>
<td>Lucas Film Ltd</td>
</tr>
<tr>
<td>Patton</td>
<td>111</td>
<td>War</td>
<td>1970</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>The Hunt for Red October</td>
<td>126</td>
<td>Action Adventure</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
<tr>
<td>National Lampoon's Vacation</td>
<td>59</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Christmas Vacation</td>
<td>97</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Home Alone</td>
<td>127</td>
<td>Drama Romance</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
<tr>
<td>Home</td>
<td>128</td>
<td>Action</td>
<td>1995</td>
<td>Columbia TriStar</td>
</tr>
<tr>
<td>Home</td>
<td>127</td>
<td>Action Adventure</td>
<td>1995</td>
<td>Paramount Pictures</td>
</tr>
</tbody>
</table>

PROC SQL allows for the subsetting of observations as was illustrated earlier with a WHERE statement in PROC PRINT. The only difference is that a WHERE clause is used in PROC SQL.

**Procedure Code:**

```sql
PROC SQL;
  TITLE 'Movie Classics';
  SELECT * FROM SSU.MOVIES WHERE RATING IN ('PG', 'PG-13');
QUIT;
```

**Results:**

<table>
<thead>
<tr>
<th>Movie Classics</th>
<th>Length</th>
<th>Category</th>
<th>Year Studio</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casablanca</td>
<td>103</td>
<td>Drama</td>
<td>1942</td>
<td>RKO</td>
</tr>
<tr>
<td>Gone</td>
<td>153</td>
<td>Action Adventure</td>
<td>1937</td>
<td>Universal Pictures</td>
</tr>
<tr>
<td>Rocky</td>
<td>129</td>
<td>Action Adventure</td>
<td>1976</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>Star Wars</td>
<td>134</td>
<td>Action Sci-Fi</td>
<td>1977</td>
<td>Lucas Film Ltd</td>
</tr>
<tr>
<td>Patton</td>
<td>111</td>
<td>War</td>
<td>1970</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>The Hunt for Red October</td>
<td>126</td>
<td>Action Adventure</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
<tr>
<td>National Lampoon’s Vacation</td>
<td>59</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Christmas Vacation</td>
<td>97</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Home Alone</td>
<td>127</td>
<td>Drama Romance</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
</tbody>
</table>

Using PROC FORMS

The FORMS procedure provides a handy tool for printing label and form information. Generally the FORMS procedure is used when information is of a repetitive nature, such as mailing labels. In its simplest form, PROC FORMS prints just the information you specify using one or more LINE statements. Suppose you wanted to output the Movie Title, Category, and Rating variables for each observation in a single column.

**Procedure Code:**

```sql
TITLE;
PROC FORMS DATA=SSU.MOVIES;
LINE 1 TITLE;
LINE 2 CATEGORY;
LINE 3 RATING;
RUN;
```

**Results (Partial Output):**

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>Category</th>
<th>Year Studio</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Wizard of Oz</td>
<td>Adventure</td>
<td>1939</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>Casablanca</td>
<td>Drama</td>
<td>1942</td>
<td>RKO</td>
</tr>
<tr>
<td>Gone</td>
<td>Action Adventure</td>
<td>1937</td>
<td>Universal Pictures</td>
</tr>
<tr>
<td>Rocky</td>
<td>Action Adventure</td>
<td>1976</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>Star Wars</td>
<td>Action Sci-Fi</td>
<td>1977</td>
<td>Lucas Film Ltd</td>
</tr>
<tr>
<td>Patton</td>
<td>War</td>
<td>1970</td>
<td>MGM / WB</td>
</tr>
<tr>
<td>The Hunt for Red October</td>
<td>Action Adventure</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
<tr>
<td>National Lampoon’s Vacation</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Christmas Vacation</td>
<td>Comedy</td>
<td>1983</td>
<td>Warner Brothers</td>
</tr>
<tr>
<td>Home Alone</td>
<td>Drama Romance</td>
<td>1990</td>
<td>Paramount Pictures</td>
</tr>
<tr>
<td>Home</td>
<td>Action</td>
<td>1995</td>
<td>Columbia TriStar</td>
</tr>
<tr>
<td>Home</td>
<td>Action Adventure</td>
<td>1995</td>
<td>Paramount Pictures</td>
</tr>
</tbody>
</table>
The FORMS procedure statement has several options that can be specified to control the appearance of output. Table 2 below illustrates many important options and their descriptions.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>Identifies the input data set.</td>
</tr>
<tr>
<td>FILE=</td>
<td>Identifies an external output file.</td>
</tr>
<tr>
<td>LINES=</td>
<td>Number of lines in a form unit.</td>
</tr>
<tr>
<td>WIDTH=</td>
<td>Number of columns in a form unit.</td>
</tr>
<tr>
<td>ACROSS=</td>
<td>Number of form units across a page.</td>
</tr>
<tr>
<td>BETWEEN=</td>
<td>Number of spaces between form units.</td>
</tr>
<tr>
<td>DOWN=</td>
<td>Number of lines to skip before printing the first form unit.</td>
</tr>
<tr>
<td>ALIGN=</td>
<td>Number of dummy form units to print for alignment purposes.</td>
</tr>
<tr>
<td>COPIES=</td>
<td>Number of form units to print for each observation in data set.</td>
</tr>
</tbody>
</table>

Table 2. PROC FORMS options

Suppose you wanted to print the same information as the previous example, but instead of a single column of form units, you prefer to instruct PROC FORMS to construct two columns of form units. This is a popular format used with many of the leading printer-label products in use today (e.g., Avery).

Procedure Code:
```
TITLE;
PROC FORMS DATA=SSU.MOVIES ACROSS=2;
LINE 1 TITLE;
LINE 2 CATEGORY;
LINE 3 RATING;
RUN;
```

Results:

PROCs That Summarize

Detail reports are great in many situations – but sometimes contain so much information that it makes understanding their contents nearly impossible. When this is the case, a summary report may be in order. The purpose of a summary report is to collapse all the detail information in a report into easy-to-understand summary-level information. This helps to digest the enormous amounts of data frequently stored in a data set.

SAS provides the CHART, FREQ, MEANS, PLOT, SQL, TABULATE, and UNIVARIATE procedures to generate summary-level reports. For purposes of illustration, CHART, FREQ, MEANS, and UNIVARIATE will be presented.

As illustrated with the detail-level reporting procedures, each PROC is relatively easy to use, but each can also provide the level of support needed by even the most demanding programmer. By using one or more statements and options each PROC can produce simple to semi-custom reports.

Using PROC CHART

The CHART procedure is a line-oriented graphics tool that is used to print simple histograms (horizontal and vertical bar charts), block charts, and pie charts for numeric and character data. In its simplest form, PROC CHART summarizes the observations in a data set based on the variables listed in the type of chart being produced.

Suppose you wanted to display the number of movies grouped by their rating in a vertical bar chart.

Procedure Code:
```
PROC CHART DATA=SSU.MOVIES;
   TITLE ‘Chart by Movie Rating’;
   VBAR RATING;
RUN;
```

Results:
Suppose you wanted to see the number of movies grouped by their rating as a horizontal bar chart. Notice that basic-level statistics (e.g., frequency, cumulative frequency, percent, and cumulative percent) are automatically displayed with a horizontal bar chart.

Procedure Code:
```
PROC CHART DATA=SSU.MOVIES;
  TITLE 'Chart by Movie Rating';
  HBAR RATING;
RUN;
```

Results:

```
Rating | Freq | Percent | Cumulative Frequency | Cumulative Percent
---|---|---|---|---
G | 1 | 4.55 | 1 | 4.55
PG | 7 | 31.02 | 8 | 36.36
PG+13 | 14 | 63.64 | 22 | 100.00
```

To see a cross-tabulation table for two variables, you will need to specify two variable names separated with an asterisk `*`.

Procedure Code:
```
PROC FREQ DATA=SSU.MOVIES;
  TITLE '2-Way Frequency Table';
  TABLES RATING * CATEGORY;
RUN;
```

Results (Partial Output):

```
<table>
<thead>
<tr>
<th>Rating</th>
<th>Action</th>
<th>Action &amp; Adventure</th>
<th>Action &amp; Sci-Fi</th>
<th>Action &amp; Thriller</th>
<th>Comedy</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>PG</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PG+13</td>
<td>2.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.00</td>
</tr>
<tr>
<td>R</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
```

Using PROC FREQ
The FREQ procedure produces one-way to n-way frequency and cross-tabulation tables for numeric or character variables. In its simplest form, PROC FREQ produces a one-way frequency table.

Procedure Code:
```
PROC FREQ DATA=SSU.MOVIES;
  TITLE 'Frequency by Movie Rating';
  TABLES RATING;
RUN;
```

Results:

```
Rating | Freq | Percent | Cumulative Frequency | Cumulative Percent
---|---|---|---|---
G | 1 | 4.55 | 1 | 4.55
PG | 7 | 31.02 | 8 | 36.36
PG+13 | 14 | 63.64 | 22 | 100.00
```

Using PROC MEANS
The MEANS procedure produces descriptive statistics for numeric variables only. In its simplest form, PROC MEANS will produce descriptive statistics for all numeric variables in a data set.

Procedure Code:
```
PROC MEANS DATA=SSU.MOVIES;
  TITLE 'Descriptive Statistics';
RUN;
```

Results:

```
Variable | N | Mean | Std Dev | Minimum | Maximum
---|---|---|---|---|---
Length | 22 | 124.503989 | 25.8204714 | 97.001000 | 196.000000
Year | 22 | 1398.31 | 15.212926 | 1393.00 | 1397.00
```
Analysis can be performed to help understand the data by using a numeric or character categorical variable in a CLASS statement. Using the CLASS statement will create descriptive statistics as subgroups. The next example illustrates descriptive statistics for each subgroup of movie by Movie Rating.

Procedure Code:

```sas
PROC MEANS DATA=SSU.MOVIES;
  TITLE 'Descriptive Statistics';
  CLASS RATING;
RUN;
```

Results:

```
Descriptive Statistics
The MEANS Procedure

          N   Mean  Std Dev  Minimum  Maximum
Movie Rating
1 064.0000      94.0000    19.5000     64.0000     116.0000
1 070.0000     102.0000    22.0000     70.0000     128.0000
1 075.0000     106.0000    20.7500     82.0000     122.0000
```

As with PROC MEANS, analysis can be performed to help understand the data by using a numeric or character categorical variable in a CLASS statement. Using the CLASS statement will create descriptive statistics as subgroups. The next example illustrates descriptive statistics for each subgroup of movie by Movie Rating.

Procedure Code:

```sas
PROC UNIVARIATE DATA=SSU.MOVIES;
  TITLE 'Descriptive Statistics';
  CLASS RATING;
RUN;
```

Results (Partial Output):

```
Descriptive Statistic
The UNIVARIATE Procedure

Moments
  Mean  101.0
  Variance  104
  Skewness
  Kurtosis
  Uncorrected SS  104
  Corrected SS  104
  Coef Variation
```

Using PROC UNIVARIATE

The UNIVARIATE procedure works similar to the MEANS procedure, except it provides a larger number of descriptive statistics. In its simplest form, PROC UNIVARIATE produces descriptive statistics for all numeric variables in a data set.

Procedure Code:

```sas
PROC UNIVARIATE DATA=SSU.MOVIES;
  TITLE 'Descriptive Statistics';
RUN;
```

Results (Partial Output):

```
Descriptive Statistics
The UNIVARIATE Procedure

Moments
  Mean  101.0
  Variance  104
  Skewness
  Kurtosis
  Uncorrected SS  104
  Corrected SS  104
  Coef Variation
```

PROCs for Data Management

Now we will turn our attention to a PROC that is used universally within the SAS user community to help with data management tasks: the DATASETS procedure. PROC DATASETS is a powerful PROC for any SAS user to know. It provides all the tools necessary to manage a SAS data library and the members within it. Table 3 below illustrates the various statements and tasks the DATASETS procedure can perform.

Since the DATASETS procedure is an interactive procedure (like the SQL procedure), it remains active even after a RUN statement is issued. To turn it off you would issue a QUIT statement. The general form looks something like the following:

```sas
PROC DATASETS LIBRARY=libref;
  Datasets-statement _________________;
QUIT;
```

The beauty of the DATASETS procedure is that it copy, save, age, rename, and delete data sets. It can also produce a contents listing containing one or more members of a data library. Finally, it can be used to create a backup and recovery or data set aging process for important data sets.
Table 3. PROC DATASETS Statements and Tasks

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>Create a backup and recovery process for important data sets.</td>
</tr>
<tr>
<td>APPEND</td>
<td>Concatenate one or more data set observations to the end of a “master” data set.</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>Produce a detailed description of the members in a SAS data library.</td>
</tr>
<tr>
<td>COPY</td>
<td>Replicate one or more members in a SAS data library.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete (remove) one or more data sets from a SAS data library.</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Change attributes for one or more variables in a data set.</td>
</tr>
<tr>
<td>REPAIR</td>
<td>Restore damaged data sets or catalogs to a usable condition.</td>
</tr>
<tr>
<td>SAVE</td>
<td>Save specified members in a SAS data library and automatically deletes members not specified.</td>
</tr>
</tbody>
</table>

Describing Members of a Data Library

Suppose you were asked to produce a contents listing of an important SAS data library. By using PROC DATASETS, you could generate a detailed member listing of any SAS data library easily and quickly.

Procedure Code:

```
PROC DATASETS LIBRARY=SSU;
QUIT;
```

Results:

```
Proc DATASETS Library=SSU;  ----Directory-----
Libref:     SSU
Engine:     V8
 physical name: C:\SSU\2001\New Orleans\Papers
             File Name: C:\SSU\2001\New Orleans\Papers
             Files
             + Name          Mctype          Size          Last Modified
             MOVIES DATA     9316          01/Jan/2000 16:19:54
QUIT;
```

Aging for Backup and Recovery Purposes

Being able to create a backup and recovery process for important data sets can be critical should disaster strike. One or more related data sets can be assigned an aging number corresponding to when it was last updated. Suppose you wanted to safeguard the Movies data set by creating a safety net consisting of three versions of the data. The most recent version would be called MOVIES1, the next most recent version would be called MOVIES2, and the oldest and least recent version would be called MOVIES3.

Procedure Code:

```
PROC DATASETS LIBRARY=SSU;
    AGE MOVIES MOVIES1-MOVIES3;
QUIT;
```

Analysis:

Once this procedure is executed, each time the MOVIES data set is updated it will automatically rename the current data set to the first member name in the list, the next most recent data set in the member list will be renamed to the second name in the member list, and the oldest data set in the member list will be deleted. This aging process is especially useful for important data sets that are updated frequently.

Repairing Damaged Data Sets

On rare instances where a system failure occurs (e.g., during a power brownout or electrical storm) during an update operation, a data set may become damaged and unusable. To fix a problem like this and restore a data set to a usable condition, it may be necessary to try to rebuild data set indexes. The PROC DATASETS REPAIR statement can be a life-saving statement (at least as it related to rescuing your damaged data set).

Suppose during an unexpected power outage our computer system experienced a problem that caused the Movies data set to become damaged. You could then use the REPAIR statement to try to restore the data set back to usability.

Procedure Code:

```
PROC DATASETS LIBRARY=SSU;
    REPAIR MOVIES;
QUIT;
```

Results:

```
Proc DATASETS Library=SSU;  ----Directory-----
Libref:     SSU
Engine:     V8
 physical name: C:\SSU\2001\New Orleans\Papers
             File Name: C:\SSU\2001\New Orleans\Papers
             Files
             + Name          Mctype          Size          Last Modified
             MOVIES DATA     9316          01/Jan/2000 16:19:54
QUIT;
```

NOTE: Repairing SSU MOVIES (mctype=DATA).
NOTE: PROCEDURE DATASETS used; real time 0.04 seconds
Conclusion
Delivering timely and quality looking reports to management, end users, and customers is critical. With the SAS System's "canned" PROCedures for generating quick results, users around the world appreciate the benefits of using these "tried and proven" tools. Basic SAS PROCedures such as PRINT, SQL, and FORMS for detail reporting, and CHART, FREQ, MEANS, and UNIVARIATE for summary reporting are worth their weight in gold. So learn their syntax, use them wisely, and you will be happy you did. Happy computing!

Acknowledgments
The author would like to thank Tom Winn, Texas State Auditor's Office, Imelda Go, Richland County School District One, and Andrew T. Kuligowski, Nielsen Media Research for their support and encouragement in the creation of this paper, and for asking me to be a an invited speaker in the first place. I would also like to thank Deborah Babcock Buck, D. B. & P. Associates and S. David Riba, JADE Tech, Inc. for all their hard work and for doing a great job as SSU 2001 Conference C-Chairs. Thank you!

References
Lafler, Kirk Paul, SAS® Fundamentals, Version 8 Course Notes, Revised and Updated 2001, Software Intelligence Corporation, Spring Valley, CA, USA.

Trademark Citations
SAS, SAS Quality Partner, and SAS Certified Professional are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

About the Author
Kirk is a SAS Quality Partner® and SAS Certified Professional® with 25 years of experience working with the SAS System. He has authored over one hundred articles on computing and technology and has presented at SAS Users Group International (SUGI) conferences, regional SAS User Groups, and local SAS User Groups since 1981. His popular SAS Tips column appears regularly in the SANDS and SESUG Newsletters and is read by thousands of SAS users. When Kirk isn't writing, teaching or consulting, he can be found enjoying all that San Diego has to offer with his wife and son.

Kirk can be reached at:

Kirk Paul Lafler
Author, Speaker, and Consultant
Software Intelligence Corporation
P.O. Box 1390
Spring Valley, California 91979-1390
E-mail: KirkLafler@cs.com
Website: http://www.software-intelligence.com