Gupta Programming
SAS Certified Professional
Simi Valley, CA
Register by phone (805) 551-3222 or by e-mail Sunil@GuptaProgramming.com

SAS® Essentials (Introduction to SAS) – *NEW!*

COURSE DESCRIPTION

Looking to getting started using one of the most powerful software tools available for data management and analysis? This course will introduce SAS by providing the key concepts required to understand the basics of the SAS programming language. All major topics are covered including accessing data, creating data structures, managing data, generating reports and ODS. Simple task-oriented examples are used to explain the SAS syntax for the DATA Step and SAS Procedures.

OUTLINE

Understanding SAS and the Data Step

Understanding typical tasks to perform using SAS Software
Understanding how SAS works
Using SAS Windows: Results, Explore, Editor, Output, Log, On-line Help
Working with SAS Files

Data Access

- Accessing data sets – approach, permission, SAS Viewer (LIBNAME)
- Viewing your data – numeric, character, dates
- Importing data from Excel (PROC IMPORT)

Data Management

- Using the Data Step to create data sets (DATA STEP)
- Selecting Variables to restrict information (KEEP/DROP)
- Selecting Observations to subset data – numeric, character, dates (WHERE)
- Creating Variables – numeric, character, dates (LENGTH)
  - Merging Data sets together to create new data sets
  - By statement (common variables)

Using SAS Procedures

Reviewing the Requirements

Data Analysis

- PROC Step Elements
- Understanding the data set structure with PROC CONTENTS
- Displaying data with PROC PRINT
- Sorting data with PROC SORT
- Summarizing data with PROC FREQ
- Displaying descriptive statistics with PROC MEANS

Data Presentation

- Creating Excel, RTF, PDF and HTML files with ODS
- E-mailing files to a distribution list
- Exporting data to Excel with ODS

Class Outline: SAS Essentials

Understanding SAS and the Data Step

Working in the Sales and Marketing Environment

Understanding typical tasks to perform using SAS Software

Category	Task
Data Access	View the Product Master Data set
Data Management	Subset and display ARANESP sales for the past month
Data Analysis	Summarize ARANESP sales by sales category for the current month
Data Presentation	Create an Excel file

Understanding how SAS works

Using SAS Windows
- Results, Explore, Editor, Output, Log, On-line Help

Working with SAS Files
- System Documentation
- Data set – viewing data
- Program – writing and running code
- List – viewing output
- Log – checking messages

Data Access
- *Example Task: View the Product Master Data set*
- Accessing Amgen data sets – approach(PC/UNIX), permission, SAS Viewer (LIBNAME)
- Viewing your data – numeric, character, dates
- Importing data from Excel (PROC IMPORT)

Data Management
- *Example Task: Subset and display ARANESP sales for the past month*
- Using the Data Step to create data sets (DATA STEP)
- Selecting Variables to restrict information (KEEP/DROP)
- Selecting Observations to subset data – numeric, character, dates (WHERE)
- Creating Variables – numeric, character, dates (LENGTH)

- Merging Data sets together to create new data sets
  - By statement (common variables)
  - - One-to-one
  - - One-to-many

Class Outline: SAS Essentials

Using SAS Procedures

Reviewing the Requirements

Task – determine which SAS Procedure to use
Data set – identify and assure all information is in a single data set
Variables – identify and know variable type (numeric, character)
Subset condition – know the data, how and when to apply
Report Layout – add titles and footnotes, by-group processing

Data Analysis

Example Task: Summarize ARANESP sales by sales category for the current month

PROC Step Elements

Understanding the data set structure with PROC CONTENTS
DATA = _ALL_

Displaying data with PROC PRINT
VAR statement
SUM statement
WHERE statement
BY statement

Sorting data with PROC SORT
BY statement (required)
WHERE statement
Permanent/Temporary Data set

Summarizing data with PROC FREQ
TABLES statement – one-way variable, two-way variables
WHERE statement

Displaying descriptive statistics with PROC MEANS
N, SUM, MEAN, MIN, MAX
VAR statement
CLASS and BY statements
WHERE statement

Data Presentation

Example Task: Create an Excel file
Creating Excel, RTF, PDF and HTML files with Output Delivery System (ODS)
E-mailing files to a distribution list
Exporting data to Excel with ODS

Class Notes: SAS Essentials

Introduction

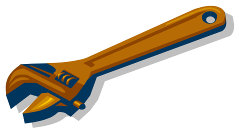
Sunil Gupta

I have been using SAS for over 12 years. I have found SAS to be one of the most powerful software tools today for data access, data management, data analysis and data presentation. It is also very easy and fun to learn as I discover more procedures and options. It has enabled me to generate reliable results in a quality-controlled environment for quick delivery of information. The ability to create RTF, HTML and PDF files greatly increases the uses of SAS to produce production quality reports and analysis.

Students - What are your expectations from this class? What do you hope to accomplish when competing this class? As we go through the examples, think of how SAS can help in your daily tasks to access, summarize and analyze data.

Understanding SAS and the Data Step

Working in the Sales and Marketing Environment



You need tools to answer the question: Who did what when?

Question	Options
WHO	Customer / Product / Territory
WHAT	Sales Information – dollar, unit, etc.
WHEN	Time Period - past month, recent 6-months, etc.

Understanding typical tasks to perform using Statistical Analysis Software (SAS)

SAS has been used for over 25 years by more than 3.5 million users worldwide across most industries such as pharmaceutical, automotive, finance, and manufacturing. FDA has required all drug submissions to contain SAS Data sets, programs and output results. The main power behind SAS is it's ability to access data in most any format, perform data management tasks, generate complex statistical analysis and tables and publish high quality reports for distribution.

SAS is a fourth generation programming language that is licensed as modules – SAS/BASE, SAS/STAT, SAS/GRAPH, etc. SAS also has other products including the Enterprise Guide and the Learning Edition. The Enterprise Guide is a menu-driven user interface to automatically perform tasks and construct SAS code. The main difference between the Learning Edition and the Enterprise Guide is that the Learning Edition does not require a SAS license and limits the number of observations in a data set to 1,000 records.

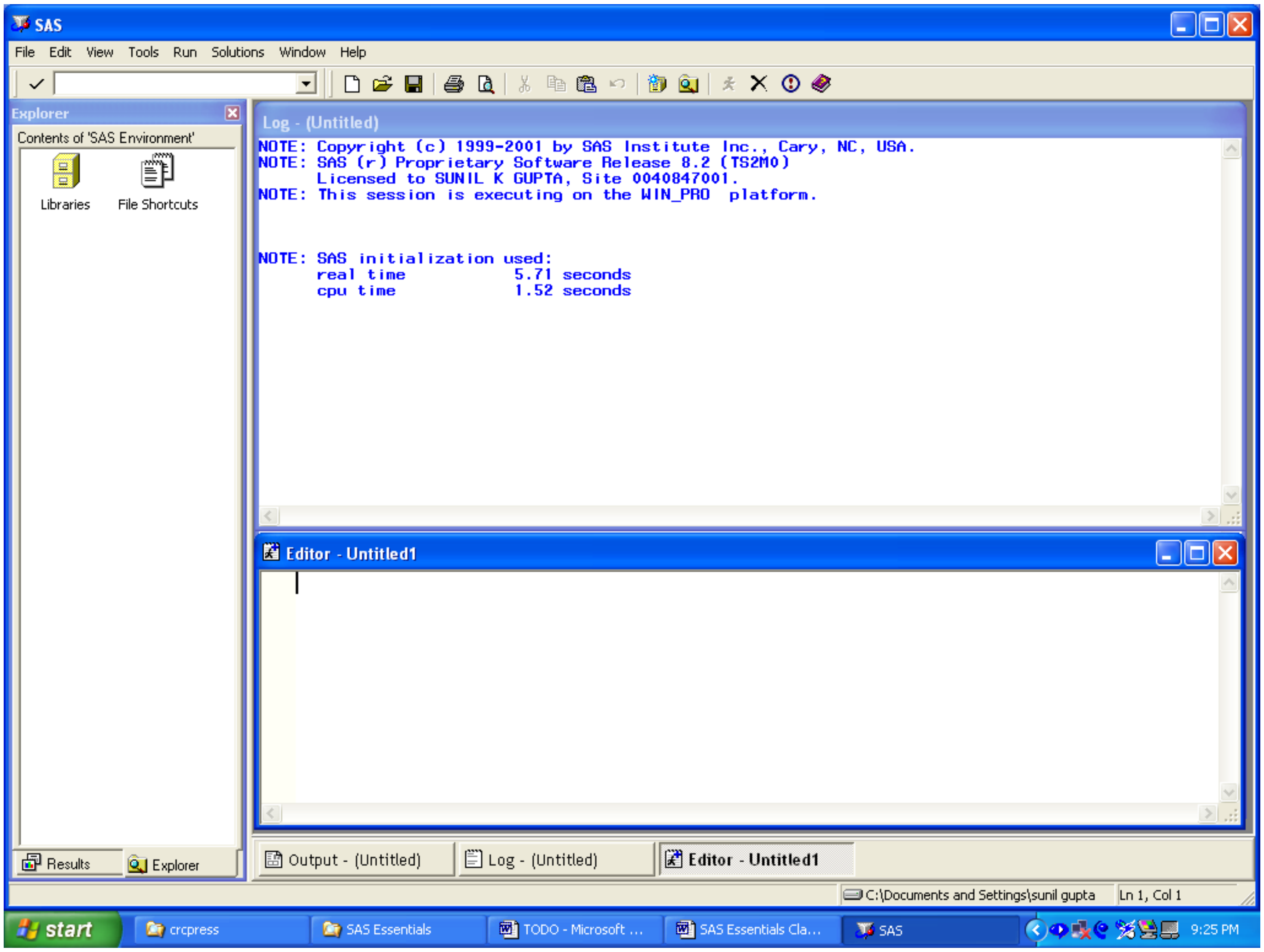
Category	Task
Data Access	View the Product Master Data set
Data Management	Subset and display ARANESP sales for the past month
Data Analysis	Summarize ARANESP sales by sales category for the current month
Data Presentation	Create an Excel file containing sales data

Understanding how SAS works

Process Flow	
Access RAW Data ⇩	DATA ACCESS of all SOURCE data sets and files
SAS DATA Step ⇩	DATA MANAGEMENT to create data sets
SAS Data Set Created ⇩	DATA MANAGEMENT to create variables
SAS PROC Step ⇩	DATA ANALYSIS to summarize and process data
Results	DATA PRESENTATION to display output in desired format

Using SAS Windows

SAS Display Manager

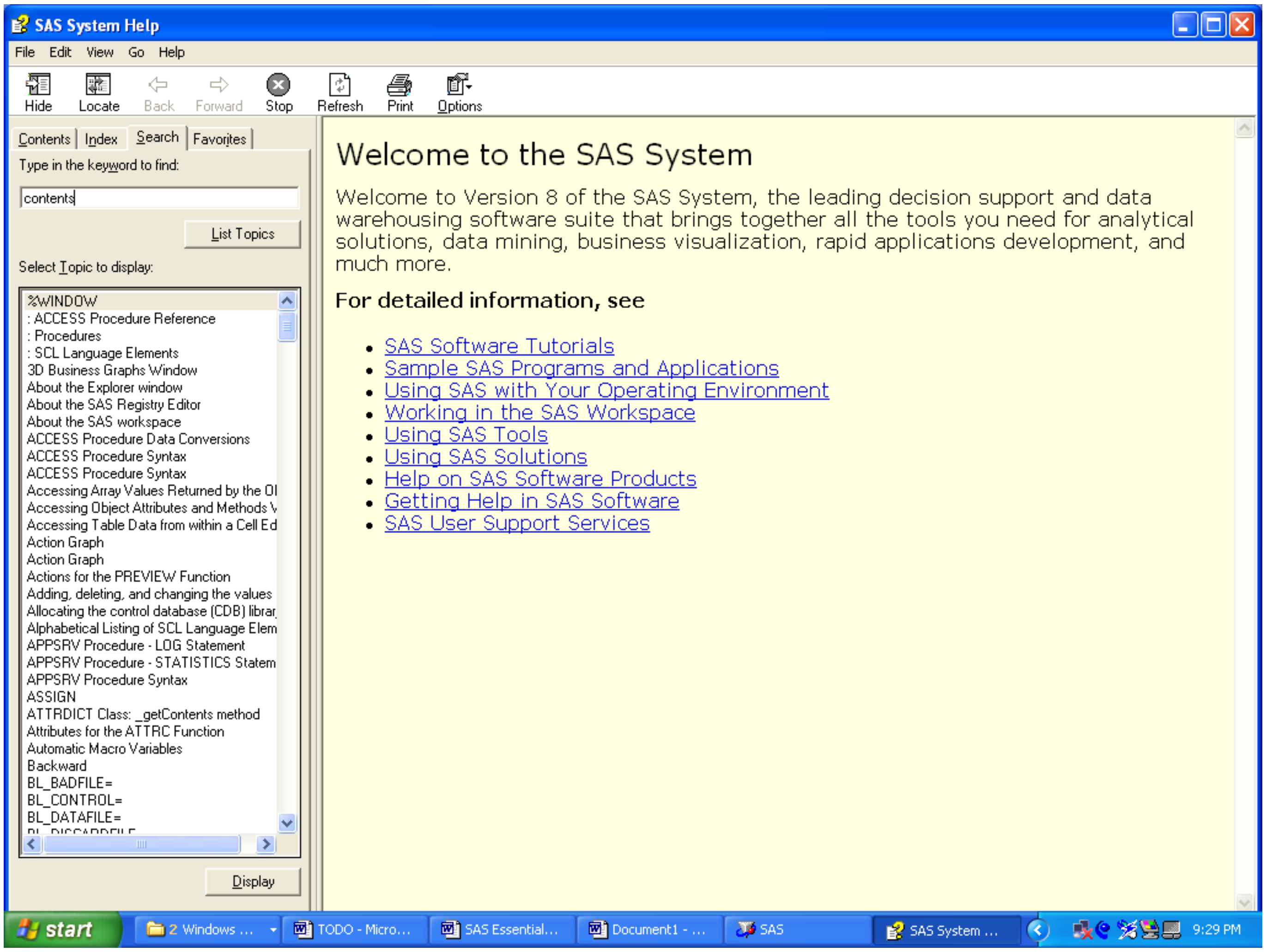


```
NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
      Licensed to SUNIL K GUPTA, Site 0040847001.
NOTE: This session is executing on the WIN_PRO  platform.


NOTE: SAS initialization used:
      real time           5.71 seconds
      cpu time            1.52 seconds
```

Window	Description
Results	A hierarchy display of ODS Output Results. Each SAS run creates output in the Results Window.
Explore	View Data Sets or Files. Assign libname statements to identify data set location.
Editor	Create Program. Type SAS statements and submit code.
Output	View List Results. View SAS list file.
Log	View Warning and Error Messages

On-line Help



Type in the keyword to find:

contents

Select Topic to display:

%WINDOW
: ACCESS Procedure Reference
: Procedures
: SCL Language Elements
3D Business Graphs Window
About the Explorer window
About the SAS Registry Editor
About the SAS workspace
ACCESS Procedure Data Conversions
ACCESS Procedure Syntax
ACCESS Procedure Syntax
Accessing Array Values Returned by the Ol
Accessing Object Attributes and Methods V
Accessing Table Data from within a Cell Ed
Action Graph
Action Graph
Actions for the PREVIEW Function
Adding, deleting, and changing the values
Allocating the control database (CDB) librar
Alphabetical Listing of SCL Language Elem
APPSRV Procedure - LOG Statement
APPSRV Procedure - STATISTICS Statem
APPSRV Procedure Syntax
ASSIGN
ATTRDICT Class: _getContents method
Attributes for the ATTRC Function
Automatic Macro Variables
Backward
BL_BADFILE=
BL_CONTROL=
BL_DATAFILE=

Welcome to the SAS System

Welcome to Version 8 of the SAS System, the leading decision support and data warehousing software suite that brings together all the tools you need for analytical solutions, data mining, business visualization, rapid applications development, and much more.

For detailed information, see

- SAS Software Tutorials
- Sample SAS Programs and Applications
- Using SAS with Your Operating Environment
- Working in the SAS Workspace
- Using SAS Tools
- Using SAS Solutions
- Help on SAS Software Products
- Getting Help in SAS Software
- SAS User Support Services

On-line Help	**Description**
Search Feature	Type in SAS syntax
Tutorials	Data Access, Data Management, Data Analysis, Data Presentation
Sample SAS Programs	See example programs

(All information is hyperlinked)

SAS Technical Support:	**1-(919)-677-8008**
Amgen License Number:	**0019155001**
Amgen PC SAS Installation Support:	**7HELP (74357)**

Working with SAS Files

System Documentation

- CMA: Customer Master and Alignment Tables
  List of all key data sets and variables
  Unique keys – useful to identify record and combine data sets

Data Set	Description/Unique Keys
ACCOUNT_CLASS	Account Classification File – Stores information such as major segments, minor segments, market segment, primary class, etc. CUSTOMER_NUM/PARTY_ID + CLASSIFY_TYPE + CLASSIFY_CD + START_DT + END_DT
ACCOUNT_IDS	Account ID's File – Stores each account's id's in various systems CUSTOMER_NUM/PARTY_ID + ID_TYPE + ID_NUM + START_DT + END_DT
ACCT_NOMINAL_INFO	Account Basic Information – names, address, status and type CUSTOMER_NUM/PARTY_ID
CUST_TERR_ALIGN	Customer to Territory Alignment – Link Customer with Territory CUSTOMER_NUM + TERR_ID/TERR_NUM + START_DT + END_DT
PROD_MASTER	Product Master Data – Link Product Name with Product Number. Source_id value is required. SOURCE_ID + P_GROUP + PROD_GROUP
TERRITORY_INFO	Each Amgen Territory information TERR_ID/TERR_NUM + PROD_BRAND + START_DT + END_DT
TNEPHD10	DDD monthly units and dollars data with month-specific variables PROD_GROUP + OUTLET + TERRITORY

COLOR CODE SYSTEM

Yellow	Alignments: CUST_TERR_ALIGN, TERRITORY_INFO
Violet	Professionals
Orange	General Information
Green	Accounts: ACCOUNT_CLASS, ACCOUNT_IDS, ACCOUNT_NOMINAL_INFO
Purple	BU Master
Dark Green	Sales Reps
	Other: PROD_MASTER, TNEPHD10

- SDS Migration Document: Alignment and Customer Master
  Proc Contents of key data sets

- BU_ID
  Value lookup list of BU_NAME variable from BU_INFO data set

- Crosstab: BU_NAME*SALES_FORCE_NAME*PRODUCT_BRAND
  Value lookup list of combinations of these three variables in the data

Most CMA files can be identified by either the CUSTOMER_NUM or PARTY_ID. CUSTOMER_NUM is a character variable with values inherited from the old ACM system's ACIS number. PARTY_ID is an automatically generated sequence number.

SAS Rules

- All statements must end with a semi-colon (;).
- A RUN statement should be placed at end of each DATA and PROC steps.
  Day 1: DATA Step to create data sets – required step
  Day 2: PROC Step to analyze data – depends on task required
- Comments can be applied as follows –

```
* This is a SAS comment on one line. ;
/* This is a SAS comment
   covering multiple lines. */
```

Data set – viewing data

- Columns are called variables ex. P_GROUP
- Rows are called observations ex. PROCRIT

	Variable 1	Variable 2	...
Observation 1			
Observation 2			
...			

Program – writing and running code

- Type in SAS statements
- Review and make changes
- Submit SAS statements for execution

List – viewing output

- Display SAS procedure results

Log – checking messages

- Display SAS statements submitted
- Display of error and warning messages from SAS execution

Data Access

Example Task: View the Product Master Data set

Accessing Amgen data sets

Approach (PC/UNIX Environment)

Option: Add this block of code to run SAS Programs from SAS on PC

```
filename rlink "C:\Program Files\SAS Institute\SAS\V8\connect\saslink\tcpunix.scr";

options comamid=tcp;
options remote=troy;     /* unix server name */
signon;
rsubmit;

       /* Insert SAS Code here */

endrsubmit;
```

Directory Structure
– Test: /SASTEST/SUNIL
- Production: /SASDATA/

Permission
Read and Write Access

SAS Viewer (LIBNAME)
Libname statement – used to create a libref to point to SAS Data sets – short hand notation of full path name of data set directory.

PC – uses a letter reference and the "\" symbol to specify the directory path.
Map Drive S to \ATO-Services\Yellowstone\Amgen Public\SAS Training\Datasets
Ex. LIBNAME CMA 'S:';

UNIX – does not use a letter reference and uses the "/" symbol to specify the directory path.
Ex. LIBNAME CMA '**/**SASTEST**/**SUNIL';
Ex. LIBNAME CONTRACT '/SASDATA/TRANSFER/ALIGN/CONTRACTS';

Instructions
- Type the PC CMA LIBNAME statement in the Editor window. Run Code.
- Double-click PROD_MASTER data set.
- Scroll across to see all variables.
- Scroll down to see different observations.

Viewing your data – numeric, character, dates

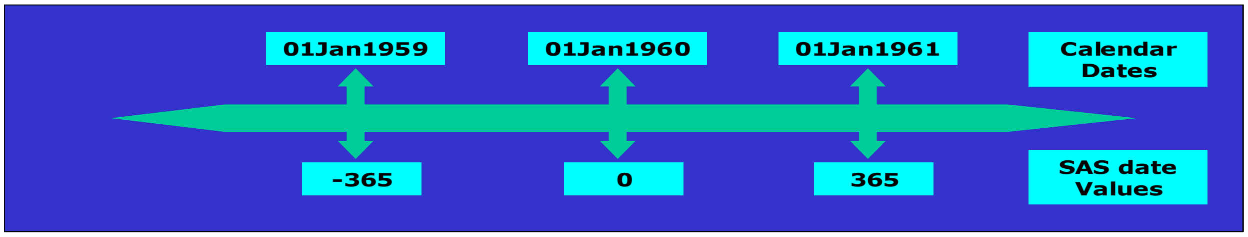
Data set Attributes: ❶ name, ❷ # observations, ❸ # variables
Variables
- Attributes: ❹ name, ❺ type (character, numeric), ❻ length, ❼ format, ❽ label.
- Maximum number of variables allowed in a data set is 32,767.
- Default Length of variables (character, numeric) is 8 bytes.

Character Variables
- Data is left justified. Data is entered from left to right. Ex 'ABC '.
- Character data is case-sensitive.
  For example: 'PROCRIT' is not the same as 'Procrit'.
- Length can be from 1 to 32,767 characters.
- Embedded blanks may exist in variables, eg. 'PROCRIT 20K'is allowed in the data.

Numeric Variables
- Data is right justified. Length can be from 3 to 8 bytes.

Date Variables
- Are numeric variables.

- Dates are converted to SAS date values. A SAS date value is the number of days from January 1, 1960, to the given date.



- SAS Data Values are numeric data type; values are stored as an integer. Formats are used to display dates in understandable form. Ex WORDDATE12. to display dates as SEP 12, 1990.

- Date Constants must be assigned in the format 'DDMMMYY' or 'DDMMMYYYY', where the text is enclosed in single quote, and followed with the letter 'D'.

- Any Mathematical Operation can be applied on SAS date values to perform date calculations.

TASK	EXAMPLE	DATA VALUE
Create Date	Dob = '01JAN1960'D;	0
Condition	Where dob lt '01JAN61'D;	365

- YEARCUTOFF = 1920 (00 – 20 is 2000 – 2020, 21 – 99 is 1921 - 1999)

Variable Information

```
Data Set Name: CMA.PROD_MASTER ❶                     ❷ Observations:          904
Member Type:   DATA                                  ❸  Variables:             10
Engine:        V8                                      Indexes:               0
Created:       21:21 Sunday, November 23, 2003         Observation Length:    293
Last Modified: 21:21 Sunday, November 23, 2003         Deleted Observations:  0
Protection:                                            Compressed:            CHAR
Data Set Type:                                         Reuse Space:           NO
Label:                                                 Point to Observations: YES
                                                       Sorted:                NO


          -----Engine/Host Dependent Information-----

Data Set Page Size:         16384
Number of Data Set Pages:   8
Number of Data Set Repairs: 0
File Name:                  /sastest/sunil/prod_master.sas7bdat
Release Created:            8.0202M0
Host Created:               SunOS
Inode Number:               5233830
Access Permission:          rw-rw-r--
Owner Name:                 jlegaspi
File Size (bytes):          139264

                  -----Alphabetic List of Variables and Attributes-----
       ❹                    ❺       ❻             ❼                      ❽
 #    Variable             Type    Len    Pos    Format    Informat    Label
-----------------------------------------------------------------------------------------
 7    EQUIV_UNIT_MEAS      Char     20    208    $20.      $20.       EQUIV_UNIT_MEAS
 4    FORMAL_PROD_NAME     Char     40    128    $40.      $40.       FORMAL_PROD_NAME
 8    PRODUCT_RPT_ORDER    Char     20    228    $20.      $20.       PRODUCT_RPT_ORDER
 2    PROD_GROUP           Char     40     48    $40.      $40.       PROD_GROUP
 6    PROD_GRP_KEY         Num       8      0                         PROD_GRP_KEY
 9    PROD_NDC_CODE        Char     25    248    $25.      $25.       PROD_NDC_CODE
10    PROD_NDC_NUM         Char     20    273    $20.      $20.       PROD_NDC_NUM
 5    P_GROUP              Char     40    168    $40.      $40.       P_GROUP
 3    SHORT_NAME           Char     40     88    $40.      $40.       SHORT_NAME
 1    SOURCE_ID            Char     40      8    $40.      $40.       SOURCE_ID
```

Data Content (OBS=1)

```
Obs  SOURCE_ID                         PROD_GROUP                   SHORT_NAME

  1  PlanTrak                          011                          PROCRIT 20K MDV 2ML


                                                           PROD_
Obs  FORMAL_PROD_NAME                        P_GROUP       GRP_KEY    EQUIV_UNIT_MEAS

  1  PROCRIT 20000 Units/2ML 2ML MDV         PROCRIT         101      20000


Obs  PRODUCT_RPT_ORDER       PROD_NDC_CODE                 PROD_NDC_NUM

  1  0                       59676-312-01                  0312-01
```

Observations

- PROD_MASTER is an important data set to access because it specifies the relationship between the Product Name (P_GROUP, ex. PROCRIT) and the Product Number (PROD_GROUP, ex. 011). This is needed because many other SAS data sets contain only the Product Number (PROD_GROUP) and in general, selection criteria is made using the Product Name (P_GROUP).

- One of the required WHERE conditions is to specify the SOURCE_ID value to prevent selecting multiple Product Numbers (PROD_GROUP). The Product Number (PROD_GROUP) variable is not unique in the PROD_MASTER data set.

- This data set has one unique record for the following key variables:
SOURCE_ID + P_GROUP + PROD_GROUP

Example -

SOURCE_ID	P_GROUP	PROD_GROUP
TCR Pro	101	Proc 20
TCR Pro	102	Proc 40
TCR Pro	103	Proc 100
TCR Neup	101	Nesp 1.6
TCR Neup	102	Nesp 4.0
TCR Neup	103	Nesp

- The key variables are used to define links with other SAS data sets to combine data sets.

- *What is the only one numeric variable? PROD_GRP_KEY*

Importing data from Excel (PROC IMPORT)

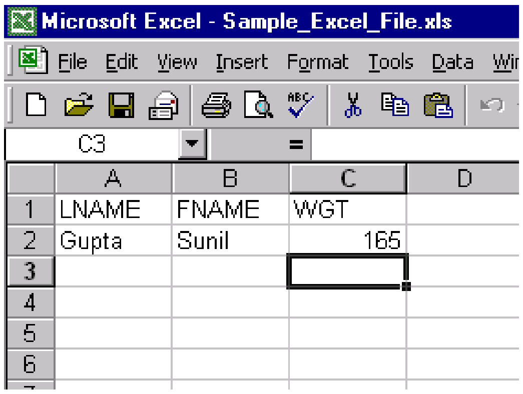
Can be used to convert Excel files to SAS data sets. *(SAS on PC ONLY)*

Accessing Excel files directly requires SAS/ACCESS.

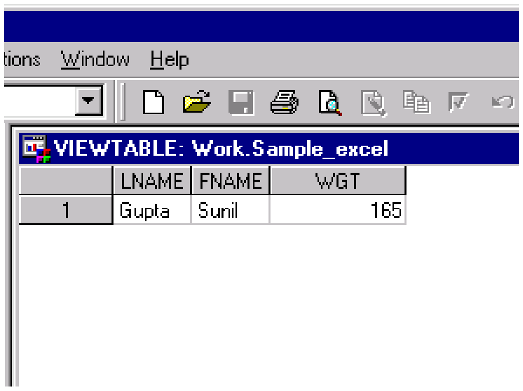
Use the Import Wizard, Save code to file.

Alternative is to save the Excel file as comma-delimited (.csv) or tab-delimited (.txt) file and then use SAS Programming Language to read the file.

Input – Excel File **Output – Data Set**


Microsoft Excel - Sample_Excel_File.xls
File Edit View Insert Format Tools Data Wi
C3 =

	A	B	C	D
1	LNAME	FNAME	WGT	
2	Gupta	Sunil	165	
3				
4				
5				
6				



	LNAME	FNAME	WGT
1	Gupta	Sunil	165

Example

```
PROC IMPORT OUT= WORK.Sample_excel
        DATAFILE = "C:\SAS Essentials\Sample_Excel_File.xls"
        DBMS=EXCEL2000 REPLACE;
        GETNAMES=YES;
RUN;
```

Observations

PROC IMPORT reads the Excel file Sample_Excel_File and creates the Sample_excel data set.

The variable names in the data set are defined from the first row in the Excel file. Ex. LNAME, FNAME, WGT.

The data set contains all values in the excel file.

The Import Wizard can use to automate this process and prevent writing SAS code.

Data Management

Example Task: Subset and display ARANESP sales for the past month

Using the Data Step to create data sets (DATA STEP)

PROD_TEMP
WORK Data Set

CMA.PROD_ALWAYS
PERMANENT Data Set

DATA Statement names and creates the SAS data set.

A **SAS Data Set** has two major components:
A descriptor portion contains information about the data set contents.
A data portion contains the actual data values – observations (rows) and variables (columns).

Temporary Data Set
- WORK libref is the implied default library.

- Available only for the duration of the SAS session
– data set must be recreated with each run.

Permanent Data Set
- Always Available - permanent MYLIB CMA is accessed.

- More Efficient than temporary data set.

SET Statement
- Reads all variables from the data sets specified.

- Any number of data sets can be combined.

Example

```
* This step creates two data sets;

data cma.prod_always prod_temp;

       set cma.prod_master;

run;
```

Observations

- DATA Step is used to create data sets

- SET statement is used to access existing data set

- Two data sets are created

- Permanent data set is PROD_ALWAYS and saved in the CMA location

- Temporary data set is PROD_TEMP

- RUN statement is required to complete the DATA Step

- All SAS statements end with a semi-colon

- A SAS comment is applied

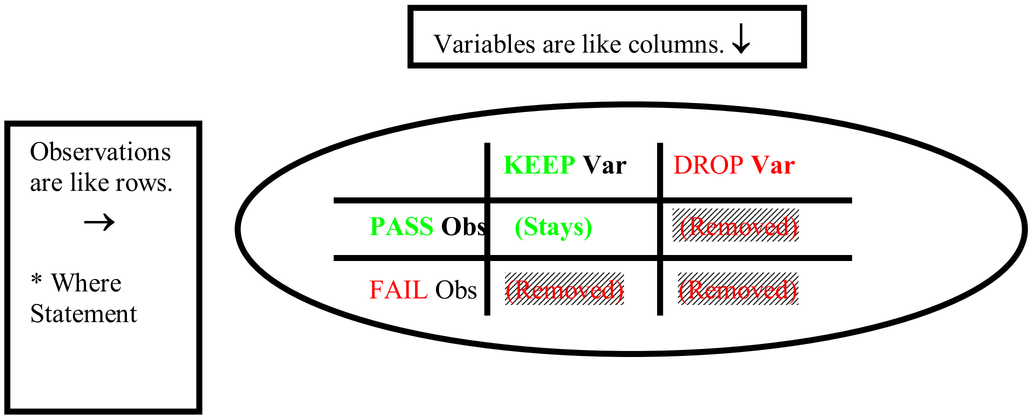
- Is the data contained in the two data sets the same or different? Same, both data sets have 904 observations and 10 variables.

Log

```
15         data cma.product_tst prod_temp;
16
17              set cma.prod_master;
18
19         run;

NOTE: There were 904 observations read from the data set CMA.PROD_MASTER.
NOTE: The data set CMA.PRODUCT_TST has 904 observations and 10 variables.
NOTE: The data set WORK.PROD_TEMP has 904 observations and 10 variables.
NOTE: DATA statement used:
      real time           0.21 seconds
      cpu time            0.13 seconds
```

Selecting Variables to restrict information (KEEP/DROP)



KEEP statement specifies selected variables to _save_ in the final data set.

DROP statement specifies selected variables _not to save_ in the final data set.

Works similar to a traffic light:
GREEN – KEEP variables and pass WHERE condition
RED – DROP variables and do not pass WHERE condition

One or more variables may be kept or dropped. Each variable is separated by a space.

Best to use *either* DROP or KEEP statement, not both, in any DATA Step. Use the one that is shortest variable list to specify. If both DROP and KEEP statements are applied together, then the order of priority is DROP and then KEEP.

Example

```
data prod_temp;

     set cma.prod_master;

     keep prod_group p_group;

run;
```

Observations

- In the new data set PROD_TEMP, only the following variables will be saved: PROD_GROUP and P_GROUP.

- In the KEEP statement, multiple variables are separated by spaces.

- Are all observations kept from the CMA.PROD_MASTER data set? Yes, because there was no WHERE condition.

Log

```
22        data prod_temp;
23
24             set cma.prod_master;
25
26             keep  prod_group p_group;
27
28        run;

NOTE: There were 904 observations read from the data set WORK.PROD_TEMP.
NOTE: The data set WORK.PROD_TEMP has 904 observations and 2 variables.
NOTE: DATA statement used:
      real time          0.08 seconds
      cpu time            0.07 seconds
```

Selecting Observations to subset data – numeric, character, dates (WHERE)

- WHERE statements are used to subset the data set based on selected values of selected variables. SAS uses these statements to filter out observations that do not meet the specified condition on the variables.

- Numeric or character variables can be applied. One of more variables can be specified in the WHERE statement. Make sure expression is appropriate for variable type. Note that numeric data can be stored in character variable. When this happens, numeric data is treated as a character value.

- WHERE statements are <u>cumulative</u> in that additional WHERE statements are added to the subset condition.

- Can apply directly in many SAS Procedures ex. PROC PRINT.

- Character Variable Based Conditions compare text within quotes. The text is case sensitive, ex. there is a difference between 'PROCRIT' and 'Procrit'. Because of the case sensitivity, you may want to use the UPCASE() function to compare the upper cases text with the upper case constant value. Also account for spaces, numbers and symbols when applying search conditions on character variables.
Equal – includes constant value,
ex. WHERE UPCASE(CLASS_TYPE) = 'SALES CLASS';

Colon Modifer (:) – compares shorter text with longer text by truncating the longer text to the length of the shorter text. This is done to include all similar records that match the specified text without listing each unique variation of the text. The default behavior is to pad the shorter text with trailing blanks in order to make the comparison with the longer text. This option is only available in a DATA STEP.
ex. WHERE P_GROUP =: 'EPO';
This will select records, for example, when P_GROUP = 'EPO 20' or 'EPO 40'.

- **Numeric Variable Based Conditions** compare numeric values. The numeric operators include =, ^=, >, <, >=, and <=. Can also use EQ, NE, GT, LT, GE, and LE. ex. WHERE AGE >= 18;

- Date Variable Based Conditions – includes date constant and date ranges. Processing date constant requires the letter D after the 'DDMONYYY' date value.
ex. WHERE START_DT <= TODAY() AND END_DT >= TODAY();
ex. WHERE DOB < '01JAN1961'D;

Special Operators include

BETWEEN … AND – includes values defined in the range,
ex. WHERE AGE BETWEEN 18 AND 34;
This is the same as the following: WHERE AGE >= 18 AND AGE <= 34 ;

IN – includes a list of values separated by spaces or comma,
Ex. WHERE CLASS_TYPE IN (‘SALES CLASS’, ‘MAJOR SEGMENT’);

Example

```
data prod_temp;

     set cma.prod_master;

     where p_group = 'ARANESP';
     keep prod_group p_group;

run;
```

Observations

- WHERE statement is applied to subset the new data set PROD_TEMP to contain P_GROUP equal to ‘ARANESP’. The number of observations went from 904 to 98. Make sure you have correct spelling and case-sensitivity of ARANESP.

- KEEP statement is also used to restrict the variables kept in the PROD_TEMP data set. More than one SAS statement may be specified in the DATA Step.

- Is the WHERE statement based on a character or numeric variable? Character variable because of the quotes.

- Is the WHERE condition also applied to the original data set PROD_MASTER? No, only to the newly created data set.

Log

```
34        data prod_temp;
35
36             set cma.prod_master;
37
38             where p_group = 'ARANESP';
39             keep  prod_group p_group;
40
41        run;

NOTE: There were 98 observations read from the data set CMA.PROD_MASTER.
     WHERE p_group='ARANESP';
NOTE: The data set WORK.PROD_TEMP has 98 observations and 2 variables.
NOTE: DATA statement used:
     real time           0.13 seconds
     cpu time             0.07 seconds
```

Output

```
The FREQ Procedure

                                    P_GROUP

                                                  Cumulative    Cumulative
P_GROUP                              Frequency     Percent      Frequency      Percent
-------------------------------------------------------------------------------------
ARANESP                                   98        100.00            98        100.00
```

Observations

Only the records related to the ARANESP product are saved in the data set.

Creating Variables – numeric, character, dates (LENGTH)

- Used to create new variables or modify existing variables.

- Useful for performing calculations or assigning values.

```
  GRADE = 'Grade A';                  /* character variable */
  Total = bparded + trans + nonrev;    /* numeric variable */
  newdt = '01jan2002'd;               /* date variable */
```

Variable names rules

- First character must be a letter or underscore. It can not be a number or a special character or symbol. Also, you can not have blanks in the variable names.

- Name can be up to 32 characters in length.

- Do not use SAS RESERVED words as SAS variable names.
  ex. DATA, KEEP, DROP, WHERE, etc.

- Either character or numeric variables can be created or modified.

LENGTH Statement

- Using the LENGTH Statement is *very important* because it defines the space allocated to store data values. This is important to avoid truncation of data. Numeric variable can have length from 3 to 8. Character variables can or have length from 1 to 32, 767 in version 8.2 and have the '$' before the length value. The default length of variables (character, numeric) is 8 bytes. A dot is required for the length of numeric variables.

- Used to define the variable's type as numeric or character.

- Multiple Variable lengths are separated by spaces or on separate LENGTH statements.

- Multiple Variables can be assigned to the same length by listing all variables separated by spaces before the length specification.

LENGTH statement before the SET statement – this is required to redefine an existing variable. The default is to define the variable attributes from the data set specified in the SET statement.

FORMAT statement

- Used to specify the display instructions of variables.
- Syntax is similar to the LENGTH statement except that a dot is required at the end of each length value.

SAS Functions can be used when creating variables.

Returns values from computations.

Requires arguments to be enclosed in parentheses.

Multiple functions have a combined effect.

Character Functions manipulate character data. Remember that character data is left justified. This means that data is entered from left to right side.

Character values can be concatenated together with the || operator.

SUBSTR('New York, NY', 5, 4) = York; (extracts part of a value based on starting position of the 5th character and the length of 4)
Ex. Short_name = substr(long_name, 5, 4);

NAME=TRIM(LEFT(name));
(Trims trailing blanks after making left justified text)
Ex. name = ' SG '; - spaces are entered before and after the SG text and the name variable is $4.
left(name) = 'SG '; - LEFT function removes leading spaces
trim(left(name) = 'SG'; - TRIM function removes trailing spaces

NAME=RIGHT(name);
Ex. name = ' SG '; - spaces are entered before and after the SG text and the name variable is $4.
right(name) = ' SG'; - RIGHT function removes trailing spaces

NAME=UPCASE(name);

NAME=LOWCASE(name);

Numeric Functions manipulate numeric data.

XN=N(1, 2, ., ., 3); (returns 3, number of non-missing values)

XMIN= MIN(x, y, z); (returns non-missing value, minimum of arguments or zero)

XSUM=SUM(x,y,z); (ignores missing values, returns non-missing value, sum of arguments or zero)

XMEAN=MEAN(., 0, 2, 4); (ignores missing values, returns 2, i.e. 6/3 – divides by number of non-missing values)

Keyword OF can be used for greater flexibility to specify variable lists and array elements. Ex. **XSUM = SUM(OF BUCKET1 – BUCKET5);**

Date Functions manipulate dates.

TODT=TODAY(); (creates a sas date value for today’s date)

MONTHC = MONTH(STARTDT); (1 to 12 value is returned)

DTDIFF=INTCK(‘WEEK‘, STARTDT, STOPDT); (# of weeks between startdt and stopdt)

May Return a date value that will not look like a date unless it has been assigned a format.

Example

```
data hcaremain;

       length acis_num $10 ac_name $120 new_party_id 8.;

       set cma.acct_nominal_info;

       acis_num = substr(left(customer_num),1,10);
       ac_name=account_name;
       new_party_id = party_id * 10;

       format acis_num $10. ac_name $75. new_party_id 12.;
       where status = 'A';        /* A – Active vs. I – Inactive */
run;

proc print data=hcaremain (obs=5);
 var acis_num ac_name new_party_id status;
run;
```

Observations

- LENGTH statement is used to define the variable ACIS_NUM a length of 10, variable AC_NAME with a length of 120 and variable NEW_PARTY_ID a length of 8. The $ specifies a character variable. FORMAT specifies how the variables are displayed.

- The following new variables are created -

ACIS_NUM -Substring the first 10 characters of the CUSTOMER_NUM variable (example of a character function)

AC_NAME – Equal to the ACCOUNT_NAME variable (make sure ACCOUNT_NAME variable already exists)

NEW_PARTY_ID – Equal to PARTY_ID times 10 (make sure PARTY_ID variable already exists)

- *Are the ACIS_NUM and AC_NAME variables numeric or character? Character*

Output

```
 Obs   acis_num     ac_name                                new_party_id   STATUS

   1   100029      HOSPITAL RAMON E BETANCES                    680          A
   2   100001      CASTANER GENERAL HOSPITAL                    460          A
   3   100003      MENONITA GENERAL HOSPITAL DE CAYEY           470          A
   4   100006      METROPOLITAN HOSPITAL                        480          A
   5   100360      NEWPORT HOSPITAL                            3810          A
```

Merging Data sets together to create new data sets

Merging is an important step because almost always, you will find that data sets do not have all the variables you need. The data sets are designed this way to be efficient and organized. You are required to merge data sets together to extract the information stored in multiple data sets. The net effect is that you are adding variables from another data set based on a link through common variables. With a BY statement, the common values will be joined to the same record.

MERGE Statement
- Creates data set containing variables from all data sets listed.

- Requires Data Types to match for all common variables in all data sets.

- Data Values for common variables from the *second data set* overwrite the data values from the first data set. In general, this should not be problem because source data sets contain different variables except for key variables.

- Any Number of data sets can be merged together.

- **(IN =)** option along with the IF statement is used to keep records from a specified data set. The variable after the equal sign is temporarily created in the data set to identify all records with a value of 1 coming from the specified data set. All other records not coming from the specified data set will have a value of 0. The IF statement keep all records with a value of 1. The default is to keep all records from each data set merged.

PROD_MASTER		TNEPHD10		PROD_2		
prod_group	**A**	**prod_group**	**B**	**prod_group**	**A**	**B**
001	1			001	1	0
002	1	*002*	*1*	*002*	*1*	*1*
		003	*1*	*003*	*0*	*1*
		004	*1*	*004*	*0*	*1*

BY statement (common variables) – two types of merges
- One record-to-one record data set merge results in the same number of observations. This is because the same number or less records exist in the ACCOUNT_NOMINAL_INFO data set.

ex. ACCOUNT_CLASS merge with ACCOUNT_NOMINAL_INFO by CUSTOMER_NUM

- One record-to-many records data set merge results in more observations. This is because multiple records exist in the TNEPHD10 data set for the P_GROUP variable.

ex. PROD_MASTER merge with TNEPHD10 sales by P_GROUP

- Requires BY variables to match variable name, type, and attributes in both data sets. More than one BY variable may exist.

- Pairs observations only when BY variables match. Creates new observations when BY variables do not match.

- Requires pre-sorting the data sets in advance if not already sorted by the BY variables. **Missing Values** (. for numeric data and blank for character data) will be assigned for uncommon variables from either data sets, in the corresponding records from the other data set.

- New data set contains all unique observations for BY variables unless the IF statement and the (IN=) option are used. New Data Set is sorted in order of the BY variable. Can use the keyword DESCENDING before the variable's name to indicate variable is sorted in descending order.

Without having the BY statement results in a one record to one record merge.

Example

```
/* Sort the PROD_MASTER data set
   Subset to select only TCR Pro records */

Proc sort data=cma.prod_master out=prod_master;
 By prod_group;
 Where source_id = "TCR Pro";
Run;

Title 'Sample Data from PROD_MASTER – One record per PROD_GROUP';
Proc print data=prod_master;
 Var prod_group p_group;
 where prod_group in ('001', '002', '003');
Run;
```

Output

```
Sample Data from PROD_MASTER - One record per PROD_GROUP                        1
                                              17:18 Wednesday, December 17, 2003

Obs    PROD_GROUP          P_GROUP

  1    001                 EPOGEN
  2    002                 PROCRIT/CLARITIN24/CLARITIN12
  3    003                 DEXFERRUM/FERRLECIT/INFED
```

Example

```
* Sort the TNEPHD10 data set – DDD sales data set;

Proc sort data=cma.tnephd10 out=tnephd10;
 By prod_group;
Run;

/* Select a specific product group code to print */

Title 'Sample Data from TNEPHD10 – Multiple records per PROD_GROUP';
Proc print data=tnephd10 (obs=4);
 Var prod_group sales_cat outlet bucket1 bucket2;
Run;
```

Output

```
^LSample Data from TCR - Multiple records per PROD_GROUP                    5

        PROD_   SALES_
   Obs   GROUP    CAT     OUTLET                  bucket1               bucket2

     1    001     2       33716153             2071890.50            2735416.00
     2    001     2       92807162                   0.00                  0.00
     3    001     2       96910150              233968.80             189104.80
     4    001     2       96940200                   0.00                  0.00
```

Example

```
* Merge PRODUCT_MASTER and TNEPHD10 data sets to combine data;
data prod_2;

  merge prod_master tnephd10 (in=b);
  by prod_group;
  if b;
run;

Title 'Sample Data from Merge of PROD_MASTER and TNEPHD10';
Proc print data=prod_2 (OBS=4);
 Var p_group prod_group sales_cat outlet bucket1 bucket2;
 Sum bucket1 bucket2;
Run;
```

Output

```
Sample Data from Merge of PROD_MASTER and TNEPHD10                             5

                     PROD_   SALES_
   Obs    P_GROUP    GROUP    CAT      OUTLET                 bucket1                bucket2

     1     EPOGEN    001      2       33716153             2071890.50             2735416.00
     2     EPOGEN    001      2       92807162                   0.00                   0.00
     3     EPOGEN    001      2       96910150              233968.80              189104.80
     4     EPOGEN    001      2       96940200                   0.00                   0.00
                                                  ====================    ====================
                                                           2305859.30             2924520.80
```

Observations

- The data sets must be pre-sorted before merging them.

- The WHERE condition reduces the number of observations from 904 to 48 records. It is required to assure that the Product Number has a unique Product Name.

- Two new temporary data sets (PROD_MASTER, TNEPHD10) are created from the SORT procedure. This is often done to prevent replacement of the original data set.

- New data set PROD_2 is created from the merge of PROD_MASTER and TNEPHD10 data sets.

- The two data sets are merged by the BY variable PROD_GROUP. This means that for the same PROD_GROUP value, the information from the two data sets will be on the same record.

- The PROD_GROUP is the common variable in the two data sets. It must have the same variable attributes.

- The result is that we now have a single data set with the decoded Product Numbers to more meaningful Product Names for the SOURCE_ID value of 'TCR Pro'.

- The IF B; statement keeps records only from the TNEPHD10 data set.

- The BUCKET1 to BUCKET24 variables represent the most recent 24 months. BUCKET1 is the most recent month. For the TNEPHD10 data set, BUCKET1 is 2003 October's data and BUCKET24 is 2001 October data. The SUM statement adds the first four observations to get total values. This is similar to the SUM statistics from PROC MEANS. The data stored in these BUCKET variables are either units, dollars, or scripts.
- The DDD sales data follows this data set naming convention – **TNEPHD10**:

T	**NEPH**	**D**	**10**
T – TCR	NEPH – NEPHROLOGY	D – DOLLARS	OCTOBER (MONTH 1-12)
X – XPO	ONC – ONCOLOGY	U - UNITS	
P – PLAN		S - SCRIPTS	

- Several of the key variables in the TNEPHD10 data set include:
    SALES_CAT – Sales Category
    1 – Retail, 2 – Non-Retail

    OUTLET – Represents a location. Combination of ZIP + OUTLET #
    Ex. 91010 200

- Is the PROD_2 data set a temporary or permanent data set? Temporary

Log

```
93       /* Sort the PROD_MASTER data set
94          Subset to select only TCR Pro records */
95       Proc sort data=cma.prod_master out=prod_master;
96        By prod_group;
97        Where source_id = "TCR Pro";
98       Run;

NOTE: There were 48 observations read from the data set CMA.PROD_MASTER.
      WHERE source_id='TCR Pro';
NOTE: The data set WORK.PROD_MASTER has 48 observations and 10 variables.
NOTE: PROCEDURE SORT used:
      real time          0.27 seconds
      cpu time           0.10 seconds


99
100       Title 'Sample Data from PROD_MASTER - One record per PROD_GROUP';
101       Proc print data=prod_master;
102        Var prod_group p_group;
103        where prod_group in ('001' '002' '003');
104       Run;

NOTE: There were 3 observations read from the data set WORK.PROD_MASTER.
      WHERE prod_group in ('001', '002', '003');
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used:
      real time          0.22 seconds
      cpu time           0.03 seconds

114       * Sort the TNEPHD10 data set - DDD sales data set;
115       Proc sort data=cma.tnephd10 out=tnephd10;
116        By prod_group;
117       Run;

NOTE: There were 264792 observations read from the data set CMA.TNEPHD10.
NOTE: The data set WORK.TNEPHD10 has 264792 observations and 33 variables.
NOTE: PROCEDURE SORT used:
      real time          17.18 seconds
      cpu time           6.90 seconds
```

```
118
119      Title 'Sample Data from TNEPHD10 - Multiple records per PROD_GROUP';
120      Proc print data=tnephd10 (obs=4);
121       Var prod_group sales_cat outlet bucket1 bucket2;
^L4 The SAS System                      17:18 Wednesday, December 17, 2003

124      Run;

NOTE: There were 4 observations read from the data set WORK.TNEPHD10.
NOTE: The PROCEDURE PRINT printed page 2.
NOTE: PROCEDURE PRINT used:
      real time           0.04 seconds
      cpu time            0.00 seconds

127      * Merge PRODUCT_MASTER and TNEPHD10 data sets to combine data;
128
129      data cma.prod_2;
130
131        merge prod_master tnephd10 (in=b);
132        by prod_group;
133       if b;
134      run;

NOTE: There were 48 observations read from the data set WORK.PROD_MASTER.
NOTE: There were 264792 observations read from the data set WORK.TNEPHD10.
NOTE: The data set WORK.PROD_2 has 264792 observations and 42 variables.
NOTE: DATA statement used:
      real time           10.32 seconds
      cpu time            4.95 seconds


135
136      Title 'Sample Data from Merge of PROD_MASTER and TNEPH10';
137
138      Proc print data=cma.prod_2 (obs=4);
139       Var prod_group p_group sales_cat outlet bucket1 bucket2;
140      Sum bucket1 bucket2;
141      Run;

NOTE: There were 4 observations read from the data set WORK.PROD_2.
```

Contents

```
Data Set Name: CMA.TNEPHD10                                   Observations:          264792
Member Type:   DATA                                           Variables:             33
Engine:        V8                                             Indexes:               0
Created:       15:06 Wednesday, November 19, 2003             Observation Length:    232
Last Modified: 15:06 Wednesday, November 19, 2003             Deleted Observations:  0
Protection:                                                   Compressed:            NO
Data Set Type:                                                Sorted:                NO
Label:


        -----Engine/Host Dependent Information-----

Data Set Page Size:         24576
Number of Data Set Pages:   2523
First Data Page:            1
Max Obs per Page:           105
Obs in First Data Page:     86
Number of Data Set Repairs: 0
File Name:                  /sastest/sunil/tnephd10.sas7bdat
Release Created:            8.0202M0
Host Created:               SunOS
Inode Number:               5233832
Access Permission:          rw-rw-r--
Owner Name:                 jlegaspi
File Size (bytes):          62013440


                                                -----Alphabetic List of Variables and Attributes-----

 #    Variable       Type    Len    Pos    Format    Informat    Label
----------------------------------------------------------------------------------------------
32    CLIENT_NUM     Char      3    224
31    DATA_PERIOD    Char      4    220
 1    OUTLET         Char      8    192    $8.       $8.         OUTLET
 4    PROD_GROUP     Char      3    211    $3.       $3.         PRODUCT GROUP
33    REPORT_NUM     Char      2    227
 3    SALES_CAT      Char      3    208    $3.       $3.         SALES CATEGORY
 2    TERRITORY      Char      8    200    $8.       $8.         TERRITORY
 5    bucket1        Num       8      0    20.2      20.2        BUCKET01
 6    bucket2        Num       8      8    20.2      20.2        BUCKET02
 7    bucket3        Num       8     16    20.2      20.2        BUCKET03
 8    bucket4        Num       8     24    20.2      20.2        BUCKET04
 9    bucket5        Num       8     32    20.2      20.2        BUCKET05
10    bucket6        Num       8     40    20.2      20.2        BUCKET06
11    bucket7        Num       8     48    20.2      20.2        BUCKET07
12    bucket8        Num       8     56    20.2      20.2        BUCKET08
13    bucket9        Num       8     64    20.2      20.2        BUCKET09
14    bucket10       Num       8     72    20.2      20.2        BUCKET10
15    bucket11       Num       8     80    20.2      20.2        BUCKET11
16    bucket12       Num       8     88    20.2      20.2        BUCKET12
17    bucket13       Num       8     96    20.2      20.2        BUCKET13
18    bucket14       Num       8    104    20.2      20.2        BUCKET14
19    bucket15       Num       8    112    20.2      20.2        BUCKET15
20    bucket16       Num       8    120    20.2      20.2        BUCKET16
21    bucket17       Num       8    128    20.2      20.2        BUCKET17
22    bucket18       Num       8    136    20.2      20.2        BUCKET18
23    bucket19       Num       8    144    20.2      20.2        BUCKET19
24    bucket20       Num       8    152    20.2      20.2        BUCKET20
25    bucket21       Num       8    160    20.2      20.2        BUCKET21
26    bucket22       Num       8    168    20.2      20.2        BUCKET22
27    bucket23       Num       8    176    20.2      20.2        BUCKET23
28    bucket24       Num       8    184    20.2      20.2        BUCKET24
```

```
29    prodgrp        Char      3    214                          PRODUCT GROUP - This col will
be removed soon - Use col PROD_GROUP
30    salescat       Char      3    217                          SALES CATEGORY - This col will
be removed soon - Use col SALES_CAT



                           SALES_  PROD_
   Obs OUTLET   TERRITORY   CAT    GROUP             bucket1             bucket2

     1          ION         1       010          45276235.20         56159551.50
     2          ION         1       011           1792645.30          2244321.30
     3          ION         1       012            141253.40           275479.80
     4          ION         2       010         207500160.80        269593490.00
     5          ION         2       011             70471.60            75611.50


   Obs             bucket3                bucket4                bucket5

     1         44828133.40            44685961.80            55963468.10
     2          1667056.40             1164986.40             2636247.10
     3           278604.70              305828.60              663989.50
     4        222033239.00           215483672.20           280996842.90
     5            66845.60               88051.10              229507.70


   Obs             bucket6                bucket7                bucket8

     1         43921732.80            42747963.00            52858510.30
     2          2912750.70             3596325.20             5176942.50
     3           767741.20             1235372.80             2197246.80
     4        216632742.70           215449369.20           264802702.20
     5           413890.50              634268.60              873286.40


   Obs             bucket9               bucket10               bucket11

     1         40430419.10            40294442.20            53374160.90
     2          4809693.40             7672945.60            25296277.30
     3          2454545.50             4917962.80            17101477.20
     4        203938432.20           203600874.70           257528095.00
     5           693302.20              986657.00             2311602.20


   Obs            bucket12               bucket13               bucket14

     1         42400567.80            40374468.40            49510662.60
     2         28182043.20            29527171.10            39980899.90
     3         12949436.30            18503872.80            23571424.50
     4        206288005.00           207415562.10           267228885.50
     5          2489596.20             2558834.10             3516008.90


   Obs            bucket15               bucket16               bucket17

     1         40593287.80            40454691.10            48777862.40
     2         27251822.70            33749108.20            39623836.00
     3         19695316.50            15797394.60            22445419.70
     4        218102830.80           219428728.40           266161445.40
     5          2281642.10             2543112.80             2866939.70
```

```
Obs            bucket18              bucket19              bucket20

  1         36460455.30           37490008.40           45872515.60
  2         33177970.60           36825352.10           43253569.50
  3         21452686.20           23108193.10           26808347.30
  4        208439878.20          209544143.40          254318535.10
  5          2188551.70            2246381.70            3238469.20


Obs            bucket21              bucket22              bucket23

  1         39456127.20           36256754.80           42401708.70
  2         31042690.20           29134495.70           37944556.20
  3         22069149.60           19697601.10           27202405.40
  4        198986769.30          188632251.40          242231148.60
  5          2453034.40            2351482.60            2974924.80

                                                  DATA_    CLIENT_   REPORT_
Obs            bucket24    prodgrp   salescat   PERIOD      NUM        NUM

  1         44006429.10     010         1        0310       233        70
  2         35022402.20     011         1        0310       233        70
  3         26888075.10     012         1        0310       233        70
  4        238032551.40     010         2        0310       233        70
  5          2996664.30     011         2        0310       233        70
```

Class Outline: SAS Essentials

Using SAS Procedures

Reviewing the Requirements

Task – determine which SAS Procedure to use
Data set – identify and assure all information is in a single data set
Variables – identify and know variable type (numeric, character)
Subset condition – know the data, how and when to apply
Report Layout – add titles and footnotes, by-group processing

Data Analysis

Example Task: Summarize ARANESP sales by sales category for the current month

PROC Step Elements

Understanding the data set structure with PROC CONTENTS
DATA = _ALL_

Displaying data with PROC PRINT
VAR statement
SUM statement
WHERE statement
BY statement

Sorting data with PROC SORT
BY statement (required)
WHERE statement
Permanent/Temporary Data set

Summarizing data with PROC FREQ
TABLES statement – one-way variable, two-way variables
WHERE statement

Displaying descriptive statistics with PROC MEANS
N, SUM, MEAN, MIN, MAX
VAR statement
CLASS and BY statements
WHERE statement

Data Presentation

Example Task: Create an Excel file
Creating Excel, RTF, PDF and HTML files with Output Delivery System (ODS)
E-mailing files to a distribution list
Exporting data to Excel with ODS

Class Notes: SAS Essentials

Using SAS Procedures

Reviewing the Requirements

Task Determine which SAS **PROC**edures to use

- Displaying data with **PROC** PRINT
- Sorting data with **PROC** SORT
- Summarizing data with **PROC** FREQ
- Displaying descriptive statistics with **PROC** MEANS

Data set Identify and assure all information is in a single data set

- Merging Data sets together to create new data sets

Variables Identify and assure variable exists and know variable type

- Creating Variables – numeric, character, dates (LENGTH)
- Understanding the data set structure with **PROC** CONTENTS

Subset condition Know the data, how and when to apply

- Selecting Observations to subset data – numeric, character, dates (WHERE)
- Selecting variables to restrict data (KEEP/DROP)

Report Layout

- Add Titles and Footnotes
- Order of variables
- BY-Group Processing

Data Analysis

Example Task: Summarize ARANESP sales by sales category for the current month.

PROC Step Elements

PROC Statement
- **DATA =** Specifies data set to use.

- VAR statement – specifies variables to process.

- FORMAT statement – specifies the *temporary* display instructions of variables.

- BY statement – specifies by-group processing. Requires pre-sorting of the data set.

- WHERE statement - apply *temporary* subset condition to restrict observations.

- **Options** - Not all options are covered. SAS Procedure specific.

Handling of missing data for numeric variables – In general, most procedures, by default, will include missing data if it exists. PROC MEANS, however, by default, excludes missing values.

REPORT Layout

TITLE statements print up to 10 lines of titles.
eg. TITLE1 'This is the title of the Report';

FOOTNOTE statements print up to 10 lines of footnotes.
eg. FOOTNOTE1 'This is a footnote.';

Once Titles and Footnotes are specified, they remain in effect for all output until changed or cancelled.

Order of Variables specifies the order in which the variables are displayed or analyzed.

By-group processing specifies if the display or analysis is separated by by-group variables.

Understanding the data set structure with PROC CONTENTS

Important Procedure to display data set structure
- Number of variables and observations
- All variables and all attributes
- Information you need to write SAS programs to access and process the data

Example

```
Title1 'This is the Data Structure of PROD_MASTER';
Proc contents data = cma.prod_master;
Run;
Title1;
```

Output

```
                        This is the Data Structure of PROD_MASTER


Data Set Name: CMA.PROD_MASTER ❶                      ❷ Observations:          904
Member Type:   DATA                                    ❸ Variables:             10
Engine:        V8                                        Indexes:               0
Created:       21:21 Sunday, November 23, 2003           Observation Length:    293
Last Modified: 21:21 Sunday, November 23, 2003           Deleted Observations:  0
Protection:                                              Compressed:            CHAR
Data Set Type:                                           Reuse Space:           NO
Label:                                                   Point to Observations: YES
                                                         Sorted:                NO
          -----Engine/Host Dependent Information-----

Data Set Page Size:         16384
Number of Data Set Pages:   8
Number of Data Set Repairs: 0
File Name:                  /sastest/sunil/prod_master.sas7bdat
Release Created:            8.0202M0
Host Created:               SunOS
Inode Number:               5233830
Access Permission:          rw-rw-r--
Owner Name:                 jlegaspi
File Size (bytes):          139264


                  -----Alphabetic List of Variables and Attributes-----
        ❹                    ❺       ❻              ❼                  ❽
 #    Variable             Type    Len    Pos    Format    Informat    Label
-------------------------------------------------------------------------------------------
 7    EQUIV_UNIT_MEAS      Char     20    208    $20.      $20.       EQUIV_UNIT_MEAS
 4    FORMAL_PROD_NAME     Char     40    128    $40.      $40.       FORMAL_PROD_NAME
 8    PRODUCT_RPT_ORDER    Char     20    228    $20.      $20.       PRODUCT_RPT_ORDER
 2    PROD_GROUP           Char     40     48    $40.      $40.       PROD_GROUP
 6    PROD_GRP_KEY         Num       8      0                         PROD_GRP_KEY
 9    PROD_NDC_CODE        Char     25    248    $25.      $25.       PROD_NDC_CODE
10    PROD_NDC_NUM         Char     20    273    $20.      $20.       PROD_NDC_NUM
 5    P_GROUP              Char     40    168    $40.      $40.       P_GROUP
 3    SHORT_NAME           Char     40     88    $40.      $40.       SHORT_NAME
 1    SOURCE_ID            Char     40      8    $40.      $40.       SOURCE_ID
```

Observations

- Data structure of one data set – PROD_MASTER is displayed
  DATA = CMA.PROD_MASTER.

Data set Attributes: ❶ name, ❷ # observations, ❸ # variables

Variables
- Attributes: ❹ name, ❺ type (character, numeric), ❻ length, ❼ format, ❽ label.
- Maximum number of variables allowed in a data set is 32,767.

Example

Proc contents data = cma._all_;
Run;

Observations

Data structure of all data sets in CMA library are displayed
 DATA = cma._ALL_.

Output

```
The SAS System                              17:24 Wednesday, December 17, 2003   1

The CONTENTS Procedure

       -----Directory-----

Libref:            CMA
Engine:            V8
Physical Name:     /sastest/sunil
File Name:         /sastest/sunil
Inode Number:      5233824
Access Permission: rwxrwxr-x
Owner Name:        jlegaspi
File Size (bytes): 4096

 #  Name               Memtype   File Size  Last Modified
-----------------------------------------------------------------
 1  ACCOUNT_CLASS      DATA      139796480   24NOV2003:13:31:33
 2  ACCOUNT_IDS        DATA       59285504   24NOV2003:13:31:42
 3  ACCT_NOMINAL_INFO  DATA       22429696   24NOV2003:13:31:11
 4  CUST_TERR_ALIGN    DATA      413818880   24NOV2003:13:32:19
 5  D_OCT2             DATA      164372480   24NOV2003:13:33:32
 6  PRODUCT_TST        DATA         303104   26NOV2003:14:46:10
 7  PROD_ALWAYS        DATA         303104   26NOV2003:14:53:42
 8  PROD_MASTER        DATA         139264   24NOV2003:13:33:04
 9  TERRITORY_INFO     DATA        1015808   24NOV2003:13:32:44
10  TNEPHD10           DATA       62013440   25NOV2003:16:13:23
```

Displaying data with PROC PRINT

Universal Procedure to display a simple list of the data set
- Can Be Used to restrict which variables and observations are listed.

- Can Use to order variables in listing.

- Can Use to perform simple column counts and sums.

- Can temporarily use formats and labels to change display of column headers and data values.

VAR statement
– specifies order and selection of variables to display

FORMAT statement
– specifies the display instructions of variables

SUM statement
- totals values of specified numeric variables

WHERE statement
- subset data set before displaying, otherwise all observations in data set are displayed. Note that this statement can not be used with the (OBS=) data set option.

BY statement
- group processing – separate listing is generated for each combination of by variables

FORMAT statement
– specifies the temporary display instructions of variables.

Options
– LABEL - used to display variable label instead of variable name in listing

- (OBS=10) – used to display only the first 10 observations. Note that this option can not be used with the WHERE statement.

Example

```
Proc sort data = cma.prod_master out= prod_master;
 By p_group;
run;

Proc print data = prod_master (obs=3) label;
 Var prod_group p_group prod_grp_key;
 Sum prod_grp_key;
 By p_group;
 Format prod_grp_key 6.2;
Run;
```

Output

```
The SAS System                    17:29 Wednesday, December 17, 2003   1

P_GROUP=ADRIAMYCIN


 Obs   PROD_GROUP        P_GROUP                   PROD_GRP_KEY


    1   001              ADRIAMYCIN                   256


P_GROUP=ALKERAN


Obs   PROD_GROUP         P_GROUP                   PROD_GRP_KEY

    2   015              ALKERAN                      216


P_GROUP=ALL-CHEMO


Obs   PROD_GROUP         P_GROUP                   PROD_GRP_KEY

    3   004              ALL-CHEMO                   301
                                                   =======
                                                    773.00
```

Observations

The variables in the list are ordered by the VAR statement.

Only the first three rows are displayed using the variable labels instead of the default variable name.

The SUM statement displays the PROD_GRP_KEY total for the three rows.

The list is grouped by the P_GROUP variable.

Example

```
* Display all numeric variables;

Proc print data = cma.prod_master (obs=3) label;
 Var _numeric_;
 Run;
```

Output

```
PROD_MASTER All numeric variables            16:34 Friday, January 2, 2004  23

       PROD_
Obs     GRP_KEY

  1      256
  2      216
  3      301
```

Example

```
* Display all character variables;

Proc print data = cma.prod_master (obs=3) label;
 Var _character_;
 Run;
```

Output

```
PROD_MASTER All character variables            16:34 Friday, January 2, 2004  24

Obs    SOURCE_ID

  1    POTENTIAL
  2    POTENTIAL
  3    TCR LEV

Obs    PROD_GROUP

  1    001
  2    015
  3    004

Obs    SHORT_NAME

  1    ADRIAMYCIN 50
  2    ALKERAN 50
  3    ALL-CHEMO

Obs    FORMAL_PROD_NAME

  1    ADRIAMYCIN 50mg (2mg/mL)
  2    ALKERAN 50mg (5mg/mL)
  3    ALL-CHEMO

Obs    P_GROUP                                    EQUIV_UNIT_MEAS

  1    ADRIAMYCIN                                 50mg
  2    ALKERAN                                    50mg
  3    ALL-CHEMO

Obs    PRODUCT_RPT_ORDER       PROD_NDC_CODE                PROD_NDC_NUM

  1    0                       0013-1106-79                 1106-79
  2    0                       0173-0130-93                 0130-93
  3                            ALL-CHEMO                    ALL-CHEMO
```

Sorting data with PROC SORT

Procedure to rearrange observations in a data set by sorting on specified variables.
- Can sort by variables in ASCENDING or DESCENDING. The keyword DESCENDING before the variable's name is required to sort in descending order.

- Does not generate printed output.

- Does not change data values.

- Missing Values are treated as lowest possible values.

- **Required** to merge data sets with BY statement

BY statement (required)
- specify one or more variables

WHERE statement
- subset data set before sorting

Permanent/Temporary Data set
- Can replace existing data set to make a permanent change or use OUT = option to save the new sort to another data set.

Duplicate Observations can be addressed
- NODUPKEY option will remove observations with duplicate BY values from the data set.

- NODUPLICATES option will remove adjacent duplicate observations. This occurs as SAS writes the output data set. Best to use the special _ALL_ keyword in the BY statement to automatically include all variables in the sort. BY _ALL_;

Example

```
Proc sort data=cma.prod_master nodupkey out=prod_master;
 By prod_group descending p_group;
Run;

Proc print data=prod_master (obs=10);
 Var prod_group p_group;
Run;
```

Log

```
11         Proc sort data=cma.prod_master nodupkey out=prod_master;
12          By prod_group descending p_group;
13         Run;

NOTE: 524 observations with duplicate key values were deleted.
NOTE: There were 904 observations read from the data set CMA.PROD_MASTER.
NOTE: The data set WORK.PROD_MASTER has 380 observations and 10 variables.
NOTE: PROCEDURE SORT used:
      real time           0.16 seconds
      cpu time            0.06 seconds



14
15         Proc print data=prod_master (obs=10);
16          Var prod_group p_group;
17         Run;

NOTE: There were 10 observations read from the data set WORK.PROD_MASTER.
```

Output

```
The SAS System                       17:32 Wednesday, December 17, 2003   1

Obs    PROD_GROUP           P_GROUP

  1    001                  NEUPOGEN
  2    001                  LEUKINE DERIVED
  3    001                  KINERET COMBINED
  4    001                  INTRON
  5    001                  EPOGEN
  6    001                  ADRIAMYCIN
  7    002                  PROCRIT/CLARITIN24/CLARITIN12
  8    002                  PROCRIT DERIVED
  9    002                  NEUPOGEN
 10    002                  LEUKINE/ETHOYL/VELBAN
```

Observations

Notice the reduction in the number of observations from 904 to 524 due to the NODUPKEY. Any duplicate values for the PROD_GROUP and P_GROUP variables will be removed.

New data set PROD_MASTER is created.

The sort order is ascending order of PROD_GROUP and descending order of P_GROUP variable. The 001 number is before the 002 number and NEUPOGEN is before ADRIAMYCIN.

Summarizing data with PROC FREQ

Procedure used to show the distribution of variable values. All unique values are displayed and tabulated. This saves you the time to manually count each occurrence.

- To create one-way, two-way and multi-way cross-classification tables.

- Crosstabulation shows the combined frequency distribution for two or more variables.

- Information within each cell: cell frequency, cell percent, row percent, and column percent.

- Does not require pre-sorting of the data set.

TABLES statement – one-way variable, two-way variables
- Variables are separated by spaces in the TABLES statement.

- Two-way tables are accomplished by crossing the row variable * with the column variable.

- By default, variable labels are used as column headers.

- Without a TABLES statement, a frequency table is created for each variable.

- It is possible to have more than one TABLES statement.

Options – Include after the / in a **TABLES** statement
- LIST used to list combination of variables by rows instead of generating a cross-classification table.

WHERE statement
- subset data set before summarizing

Example

```
Proc freq data=cma.prod_master (obs=10);
 tables prod_group p_group prod_group*p_group;
 tables prod_group*p_group/list ;

 format prod_group $10. p_group $15.;
Run;
```

Observations

Two one-way tables PROD_GROUP and P_GROUP are generated.

One two-way table PROD_GROUP*P_GROUP is generated. PROD_GROUP is the row variable and P_GROUP is the column variable.

Second TABLES statement creates the same two-way table PROD_GROUP*P_GROUP in a row-by-row list format.

Information within each cell include: cell frequency, cell percent, row percent, and column percent. Of the first ten records, 80% had PROD_GROUP = 011 and 20% had PROD_GROUP = 012 and 30% had P_GROUP = PROCRIT.

Without the (obs=10) data set option, all data records are used in the calculation. We can see that the ARANESP product represents about 10% of all records.

Output with (obs=10) option

```
The SAS System                   17:35 Wednesday, December 17, 2003   1

The FREQ Procedure

                      PROD_GROUP
                                                 Cumulative    Cumulative
PROD_GROUP              Frequency     Percent     Frequency      Percent
--------------------------------------------------------------------------
011                             8      80.00             8        80.00
012                             2      20.00            10       100.00

                       P_GROUP
                                                 Cumulative    Cumulative
P_GROUP                 Frequency     Percent     Frequency      Percent
--------------------------------------------------------------------------
ETHYOL                          1      10.00             1        10.00
INFERGEN                        1      10.00             2        20.00
NEUPOGEN                        2      20.00             4        40.00
PROCRIT                         3      30.00             7        70.00
ROFERON                         2      20.00             9        90.00
VINBLASTINE                     1      10.00            10       100.00
```

```
The FREQ Procedure

Table of PROD_GROUP by P_GROUP

PROD_GROUP(PROD_GROUP)     P_GROUP(P_GROUP)

Frequency        |
Percent          |
Row Pct          |
Col Pct          |ETHYOL  |INFERGEN|NEUPOGEN|PROCRIT |ROFERON |VINBLAST|  Total
                 |        |        |        |        |        |INE     |
                 |        |        |        |        |        |        |
                 |        |        |        |        |        |        |
                 |        |        |        |        |        |        |
-----------------+--------+--------+--------+--------+--------+--------+
011              |      1 |      1 |      0 |      3 |      2 |      1 |      8
                 |  10.00 |  10.00 |   0.00 |  30.00 |  20.00 |  10.00 |  80.00
                 |  12.50 |  12.50 |   0.00 |  37.50 |  25.00 |  12.50 |
                 | 100.00 | 100.00 |   0.00 | 100.00 | 100.00 | 100.00 |
-----------------+--------+--------+--------+--------+--------+--------+
012              |      0 |      0 |      2 |      0 |      0 |      0 |      2
                 |   0.00 |   0.00 |  20.00 |   0.00 |   0.00 |   0.00 |  20.00
                 |   0.00 |   0.00 | 100.00 |   0.00 |   0.00 |   0.00 |
                 |   0.00 |   0.00 | 100.00 |   0.00 |   0.00 |   0.00 |
-----------------+--------+--------+--------+--------+--------+--------+
Total                   1        1        2        3        2        1       10
                    10.00    10.00    20.00    30.00    20.00    10.00   100.00
```

```
                                                     Cumulative    Cumulative
PROD_GROUP    P_GROUP            Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------------------
011           ETHYOL                     1       10.00             1        10.00
011           INFERGEN                   1       10.00             2        20.00
011           PROCRIT                    3       30.00             5        50.00
011           ROFERON                    2       20.00             7        70.00
011           VINBLASTINE                1       10.00             8        80.00
012           NEUPOGEN                   2       20.00            10       100.00
```

<u>Output without (obs=10) option</u>

```
                              P_GROUP

                                          Cumulative    Cumulative
P_GROUP            Frequency     Percent     Frequency      Percent
--------------------------------------------------------------------
ADRIAMYCIN                 1        0.11             1         0.11
ALKERAN                    1        0.11             2         0.22
ALL-CHEMO                  1        0.11             3         0.33
AMEVIVE                    7        0.77            10         1.11
ARANESP                   98       10.84           108        11.95
```

Displaying descriptive statistics with PROC MEANS

Procedure to perform descriptive statistics on data set

Variables
- must be numeric

Statistics
- Specific statistics are listed on the PROC MEANS statement

N	number of non-missing values
SUM	sum of all values
MEAN	average value of non-missing values
MIN	minimum (lowest value)
MAX	maximum (highest value)

Missing data is excluded by default unless the MISSING option is specified.

NWAY option provides only the highest level of combination of all variables. The value of _TYPE_ will be equal to 3, the highest possible value. This represents ARANESP Retail Sales and ARANESP Non-Retail Sales. When the **NWAY** option is specified, only the unique combination of P_GROUP and SALES_CAT is generated. The default is to generate all three levels – P_GROUP, SALES_CAT and P_GROUP and SALES_CAT.

P_GROUP	**SALES_CAT**	**_TYPE_**	**Description**
All Products	All Sales	0	All Products & All Sales (Retail, Non-Retail)
All Products	All Retail Sales	1	All Products & Retails Sales
All Products	All Non-Retail Sales	1	All Products & Non-Retails Sales
ARANESP	All Sales	2	All ARANESP Sales (Retail, Non-Retail)
ARANESP	**Retail Sales**	**3**	**ARANESP Retail Sales**
ARANESP	**Non-Retail Sales**	**3**	**ARANESP Non-Retail Sales**

VAR statement
- identifies the analysis variable

- list of numeric variables to summaize

- Without the VAR statement, all numeric variables will be analyzed

CLASS and **BY** statements
- generates group processing

- **CLASS** statement specifies variables whose unique values will be summarized by to form subgroups. CLASS variables may be numeric or character. Normally, each variable has a small number of unique values. Does not require presorting of data set.

- **BY** statement specifies variables whose separate analysis will be summarized by. This requires presorting of data set.

The difference between the CLASS and BY statements is in the format of the output.

In general, if both CLASS and BY statements are specified, then different variables should be listed in each statement.

FORMAT statement
– specifies the temporary display instructions of variables.

WHERE statement
- subset data set before summarizing

OUTPUT OUT = statement is used to save the results to a data set and to name statistical variables.

Example – Model 1

```
Title 'Model 1 – generate summary statistics';
PROC MEANS DATA= prod_2 (obs=4) N SUM;
 CLASS P_GROUP SALES_CAT;
 VAR BUCKET1 BUCKET2;
RUN;
```

Output with (obs=4) option

```
Model 1 - generate summary statistics        13:49 Monday, January 19, 2004  30
The MEANS Procedure

              SALES         N
P_GROUP       CATEGORY    Obs    Variable    Label       N             Sum
--------------------------------------------------------------------------
EPOGEN        1             1    bucket1     BUCKET01    1               0
                                 bucket2     BUCKET02    1               0

              2             4    bucket1     BUCKET01    4      2305859.30
                                 bucket2     BUCKET02    4      2924520.80
--------------------------------------------------------------------------
```

Partial Output without (obs=4) option

```
Model 1 - generate summary statistics         15:51 Friday, January 2, 2004  28
The MEANS Procedure

             SALES
P_GROUP      CATEGORY    N Obs   Variable   Label          N            Sum
----------------------------------------------------------------------------
ARANESP      1           21926   bucket1    BUCKET01   21926    14643773.60
                                 bucket2    BUCKET02   21926    17721525.20

             2            8930   bucket1    BUCKET01    8930      183959268
                                 bucket2    BUCKET02    8930      224970628

CALCIJEX     1              45   bucket1    BUCKET01      45        1088.70
                                 bucket2    BUCKET02      45   -570.7000000

             2              31   bucket1    BUCKET01      31        4758.50
                                 bucket2    BUCKET02      31              0
```

Observations

Access PROD_2 data set to get summary statistics N and SUM by Product Group and Sales Category. This data set was created from the merge of PROD_MASTER and TNEPHD10 data sets.

For the first four observations, the SUM values are the same as those from PROC PRINT (See page 28). When running the procedure without the (obs=4) option, ARANESP product had retail sales of $ 14,643,773.60 and non-retail sales of $ 183,959,268.00 during the month of October 2003. There were 21,926 records that were totaled for ARANESP retail sales.

All results are sent to the Output window.

- What were the retail sales for ARANESP during the month of September 2003?
$ 17,721,525.20

Example – Model 2

```
Title 'Model 2 – save summary statistics SUM to data set';
PROC MEANS DATA= prod_2;
 CLASS P_GROUP SALES_CAT;
 VAR BUCKET1 BUCKET2;
 FORMAT P_GROUP $10. BUCKET1 BUCKET2 COMMA15.;
 * Save results to SALES_MEAN data set;
 * Default: Save as original variable names – BUCKET1 BUCKET2;
   OUTPUT  OUT= SALES_MEAN  SUM=;
RUN;

Title 'SALES_MEAN Data set – Model 2';
PROC CONTENTS DATA=SALES_MEAN; RUN;

PROC SORT DATA=SALES_MEAN;
 BY P_GROUP SALES_CAT; RUN;

PROC PRINT DATA=SALES_MEAN;
 VAR P_GROUP SALES_CAT _TYPE_ BUCKET1 BUCKET2; RUN;
```

Output

```
Data Set Name: WORK.SALES_MEAN                         Observations:        60

            -----Alphabetic List of Variables and Attributes-----

#    Variable     Type    Len    Pos    Format    Informat    Label
-----------------------------------------------------------------------------
1    P_GROUP      Char     40     32    $10.      $40.        P_GROUP
2    SALES_CAT    Char      3     72    $3.       $3.         SALES CATEGORY
4    _FREQ_       Num       8      8
3    _TYPE_       Num       8      0
5    bucket1      Num       8     16    COMMA15.  20.2        BUCKET01
6    bucket2      Num       8     24    COMMA15.  20.2        BUCKET02

                     SALES_
Obs    P_GROUP        CAT    _TYPE_              bucket1            bucket2
  1                            0            1,351,367,997      1,754,704,743
  2                   1        1              177,831,363        221,543,407
  3                   2        1            1,173,536,634      1,533,161,336
  4    ARANESP                 2              198,603,042        242,692,153
  5    ARANESP        1        3               14,643,774         17,721,525
  6    ARANESP        2        3              183,959,268        224,970,628
  7    CALCIJEX                2                    5,847               -571
  8    CALCIJEX       1        3                    1,089               -571
  9    CALCIJEX       2        3                    4,759                  0
```

Observations

SALES_MEANS data set contains the results of PROC MEANS. FORMAT statement is applied to make the results easier to read.

P_GROUP or SALES_CAT variables with missing values represent all products or sales categories respectively. Thus blank values for P_GROUP and SALES_CAT in the first row represents total product sales.

Example – Model 3

```
Title 'Model 3 – save summary statistics SUM and N to data set';
* Use the NWAY option to calculate only P_GROUP and SALES_CAT combinations;
PROC MEANS DATA= prod_2 NWAY;
 CLASS P_GROUP SALES_CAT;
 VAR BUCKET1 BUCKET2;
 FORMAT P_GROUP $10. BUCKET1 BUCKET2 COMMA15.;
* Save as new variable names – required for processing multiple statistics;
 OUTPUT   OUT= SALES_MEAN_NWAY
          SUM(BUCKET1)= SUM_1          SUM(BUCKET2)= SUM_2
         N(BUCKET1) = N_1              N(BUCKET2)=N_2;
RUN;

Title 'SALES_MEAN_NWAY Data set – Model 3';
PROC CONTENTS DATA=SALES_MEAN_NWAY; RUN;

PROC SORT DATA=SALES_MEAN_NWAY;
 BY P_GROUP SALES_CAT; RUN;

PROC PRINT DATA=SALES_MEAN_NWAY;
 VAR P_GROUP SALES_CAT _TYPE_ SUM_1 N_1 SUM_2 N_2; RUN;
```

Output

```
Data Set Name: WORK.SALES_MEAN_NWAY                   Observations:        38

            -----Alphabetic List of Variables and Attributes-----

#    Variable      Type    Len    Pos    Format     Informat    Label
----------------------------------------------------------------------------
7    N_1           Num       8     32                           BUCKET01
8    N_2           Num       8     40                           BUCKET02
1    P_GROUP       Char     40     48    $10.       $40.        P_GROUP
2    SALES_CAT     Char      3     88    $3.        $3.         SALES CATEGORY
5    SUM_1         Num       8     16    COMMA15.   20.2        BUCKET01
6    SUM_2         Num       8     24    COMMA15.   20.2        BUCKET02
4    _FREQ_        Num       8      8
3    _TYPE_        Num       8      0

                     SALES_
Obs   P_GROUP         CAT     _TYPE_             SUM_1     N_1             SUM_2     N_2

  1   ARANESP         1          3          14,643,774   21926        17,721,525   21926
  2   ARANESP         2          3         183,959,268    8930       224,970,628    8930
  3   CALCIJEX        1          3               1,089      45              -571      45
  4   CALCIJEX        2          3               4,759      31                 0      31
```

Observations

Now N and SUM statistics are saved and the variable names are renamed. The NWAY option prevents calculation of missing P_GROUP or SALES_CAT values. Only the _TYPE_ = 3 records are saved. No ALL PRODUCTS and ALL SALES or ALL SALES records exist in the data set. Notice that the sample size is 38 instead of 60 as in Model 2.

Data Presentation

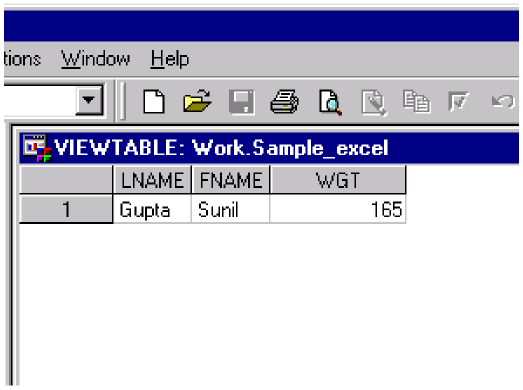
Example Task: Create an Excel file

Creating Excel, RTF, PDF and HTML files with the Output Delivery System

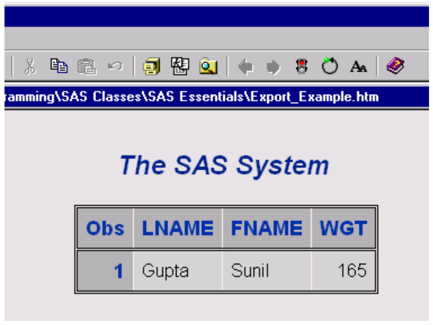
ODS can be used to convert SAS data sets to HTML files.

ODS options exist to change style attributes: Color, Font, Background, etc.
(See Customized Reports with SAS Output Delivery System Course Outline)

Input – Data Set **Output – HTML File**



	LNAME	FNAME	WGT
1	Gupta	Sunil	165



The SAS System

Obs	LNAME	FNAME	WGT
1	Gupta	Sunil	165

Example

```
* Create an HTML file;
ODS HTML FILE= "C:\SAS Essentials\Export_Example.htm";

proc print data=sample_excel;
run;

ods html close;
```

Observations

Use the HTML destination to create HTML files.

Use the .HTM file name extension.

ODS prints the Sample_excel_data set and creates the Export_Example HTML file.

The variable names in the data set are used to define the first row in the HTML file. Ex. LNAME, FNAME, WGT.

It is very important to CLOSE the HTML file before accessing it.

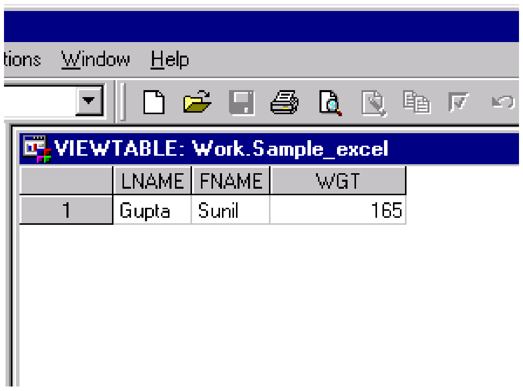
Exporting data to Excel With ODS

ODS can be used to convert SAS data sets to Excel files.

Options exist to remove font color, background color, header, etc.

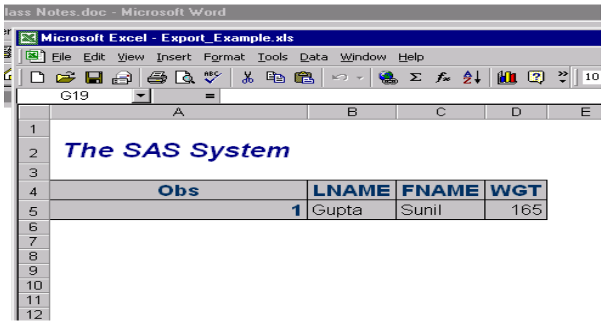
Alternative is to use PROC EXPORT to save SAS data set as an Excel file.

Input – Data Set **Output – Excel File**



VIEWTABLE: Work.Sample_excel

	LNAME	FNAME	WGT
1	Gupta	Sunil	165


lass Notes.doc - Microsoft Word
Microsoft Excel - Export_Example.xls
File Edit View Insert Format Tools Data Window Help
G19

The SAS System

Obs	LNAME	FNAME	WGT
1	Gupta	Sunil	165

Example

```
* Create an Excel file;

ODS HTML FILE= "C:\SAS Essentials\Export_Example.xls";

proc print data=sample_excel;
run;

ods html close;
```

Observations

Use the HTML destination to create Excel files.

Use the .XLS file name extension.

ODS prints the Sample_excel_data set and creates the Export_Example Excel file.

The variable names in the data set are used to define the first row in the Excel file. Ex. LNAME, FNAME, WGT.

Notice the additional items in the excel file: header – The SAS System, Obs as the first column, and the shade color of the header and the rows. These things can be removed with additional options.

E-mailing files to a distribution list (UNIX Execution)

Example

```
* Create filename statement with list of all e-mail address and excel file as an
attachement;

options emailsys=smtp;

Filename mymail email "JillE@amgen.com"
                    To = ("JillE@amgen.com" "LMao@amgen.com")
                    Subject = "SDS : :  Sales Sample Excel File"
                    Attach = '/public/sgupta/sales_sample.xls';

Data _null_;
    File mymail;
    Put 'Hi  ,';
    Put 'Find Attached a Sales Sample Excel File.';
Run;
```

Observations

You will usually place this code at the program creating the excel file. Any file created from SAS – data set, excel, etc. can be sent to an e-mail distribution list with a text message.

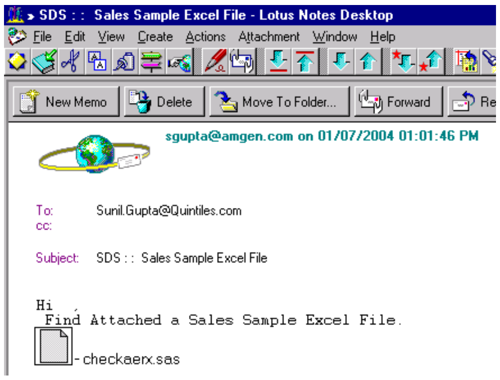
Assure the **EMAILSYS** option is set to **SMTP**;

Construct the filename statement with list of all e-mail address, subject line and the attached file (using full path name). The **MYMAIL** file name can be any name.

Add the DATA _NULL_ block of SAS code to create the text message.

Use can create the code from PC SAS but execute from UNIX using Reflection X.

Output


SDS : : Sales Sample Excel File - Lotus Notes Desktop
File Edit View Create Actions Attachment Window Help
New Memo
Delete
Move To Folder...
Forward
Re

sgupta@amgen.com on 01/07/2004 01:01:46 PM

To: Sunil.Gupta@Quintiles.com
cc:

Subject: SDS : : Sales Sample Excel File

```
Hi  ,
 Find Attached a Sales Sample Excel File.
```

- checkaerx.sas

SAS Essentials: Summary

Understanding SAS and the Data Step

Working in the Sales and Marketing Environment

Understanding typical tasks to perform using SAS Software

Understanding how SAS works

Using SAS Windows: Results, Explore, Editor, Output, Log, On-line Help

Working with SAS Files

Data Access

- Accessing Amgen data sets – approach(PC/UNIX), SAS Viewer (LIBNAME)
- Viewing your data – numeric, character, dates
- Importing data from Excel (PROC IMPORT)

Data Management

- Using the Data Step to create data sets (DATA STEP)
- Selecting Variables to restrict information (KEEP/DROP)
- Selecting Observations to subset data – numeric, character, dates (WHERE)
- Creating Variables – numeric, character, dates (LENGTH)
- Merging Data sets together to create new data sets
  - By statement (common variables)

Using SAS Procedures

Reviewing the Requirements

- Tasks, Data set, Variables, Subset condition, Report Layout

Data Analysis

- PROC Step Elements
- Understanding the data set structure with PROC CONTENTS
- Displaying data with PROC PRINT
- Sorting data with PROC SORT
- Summarizing data with PROC FREQ
- Displaying descriptive statistics with PROC MEANS

Data Presentation

- Creating Excel, RTF, PDF and HTML files with Output Delivery System (ODS)
- E-mailing files to a distribution list
- Exporting data to Excel with ODS

SAS Software Training

Sunil Gupta, Gupta Programming

Author of Quick Results with the Output Delivery System

- **SAS Essentials**
- **Sharpening Your SAS Skills**
- **Sharpening Your SAS Programming Techniques**
- **Customized Reports with SAS Output Delivery System**
- **Preparing the SAS Software Programming Environment for Regulatory Submission**

Contact Information: Sunil@GuptaProgramming.com
www.GuptaProgramming.com

sas® Certified Professional *Version 6*

Gupta Programming