

Step by step guide to Learn SAS

Venkat Reddy

Contents

- What is SAS?
- Step-1: SAS windowing environment
- Step-2: SAS Datasets and Variables
- Step-3: Importing data into SAS
- Step-4: Basic Procedures and Functions
- Step-5: Combining Datasets in SAS
- Next Steps

Before you start...

- Pre Requisites
 - Basic data base knowledge
 - Basic idea on Analytics and Applications
 - Windows OS with minimum 2 GB RAM
 - SAS software
- Disclaimer
 - This presentation is just class notes. The best way to treat this is as a high-level summary;
 - The actual session went more in depth and contained other information.
 - Most of this material was written as informal notes, not intended for publication

What is SAS?

Analysis Applications

- Predicting what will be the right segment for a marketing campaign – **Decision Trees**
- Predicting the loan repay capacity of a customer – **Logistic Regression**
- Forecasting the revenue numbers for a product - **ARIMA**
- Identifying fraud claims in healthcare insurance – **Cluster Analysis**
- The recommendation engine that predicts the best product that interests the user - **Neural networks**
- Predicting the sales numbers based on macro economic data – **Regression analysis**

What is Gauss markov theorem

Let $\tilde{\beta} = Cy$ be another linear estimator of β and let C be given by $(X'X)^{-1}X' + D$, where D is a $k \times n$ nonzero matrix. As we're restricting to *unbiased* estimators, minimum mean squared error implies minimum variance. The goal is therefore to show that such an estimator has a variance no smaller than that of $\hat{\beta}$, the OLS estimator.

The expectation of $\tilde{\beta}$ is:

$$\begin{aligned} E(Cy) &= E(((X'X)^{-1}X' + D)(X\beta + \varepsilon)) \\ &= ((X'X)^{-1}X' + D)X\beta + ((X'X)^{-1}X' + D)\underbrace{E(\varepsilon)}_0 \\ &= (X'X)^{-1}X'X\beta + DX\beta \\ &= (I_k + DX)\beta. \end{aligned}$$

Therefore, $\tilde{\beta}$ is unbiased if and only if $DX = 0$.

The variance of $\tilde{\beta}$ is

$$\begin{aligned} V(\tilde{\beta}) &= V(Cy) = CV(y)C' = \sigma^2 CC' \\ &= \sigma^2((X'X)^{-1}X' + D)(X(X'X)^{-1} + D') \\ &= \sigma^2((X'X)^{-1}X'X(X'X)^{-1} + (X'X)^{-1}X'D' + DX(X'X)^{-1} + DD') \\ &= \sigma^2(X'X)^{-1} + \sigma^2(X'X)^{-1}(\underbrace{DX}_0)' + \sigma^2 \underbrace{DX}_0(X'X)^{-1} + \sigma^2 DD' \\ &= \underbrace{\sigma^2(X'X)^{-1}}_{V(\hat{\beta})} + \sigma^2 DD'. \end{aligned}$$

Since DD' is a positive semidefinite matrix, $V(\tilde{\beta})$ exceeds $V(\hat{\beta})$ by a positive semidefinite matrix.

$$\hat{\beta} = (X'X)^{-1}X'y$$

The above theorem is used find the regression estimates

All analytics algorithms are complicated

- We need pre written libraries or codes to perform analytical operations
- Below tools will make analytics easy
 - SAS
 - R
SPSS
 - Matlab
 - Excel
 - Weka
 - RapidMiner
 - SAP
 - Mahout

Main Phases in learning analytics tool

1) Basics

- Introduction to the tool
- Coding and debugging
- Important features and limitations

2) Data Handling

- Importing the data
- Data manipulations & creating calculated fields
- Data Merging

3) Functions and Algorithms

- Basic and Advanced functions
- Statistical techniques and algorithms
- Basic Reporting and exporting the results

Step-1: SAS windowing environment

Contents

- Introduction to SAS
- Accessing SAS
- Explorer
- Results
- Program Editor or Editor
- Log
- Output windows
- My first SAS program
- SAS Libraries
- Rules if assigning a library

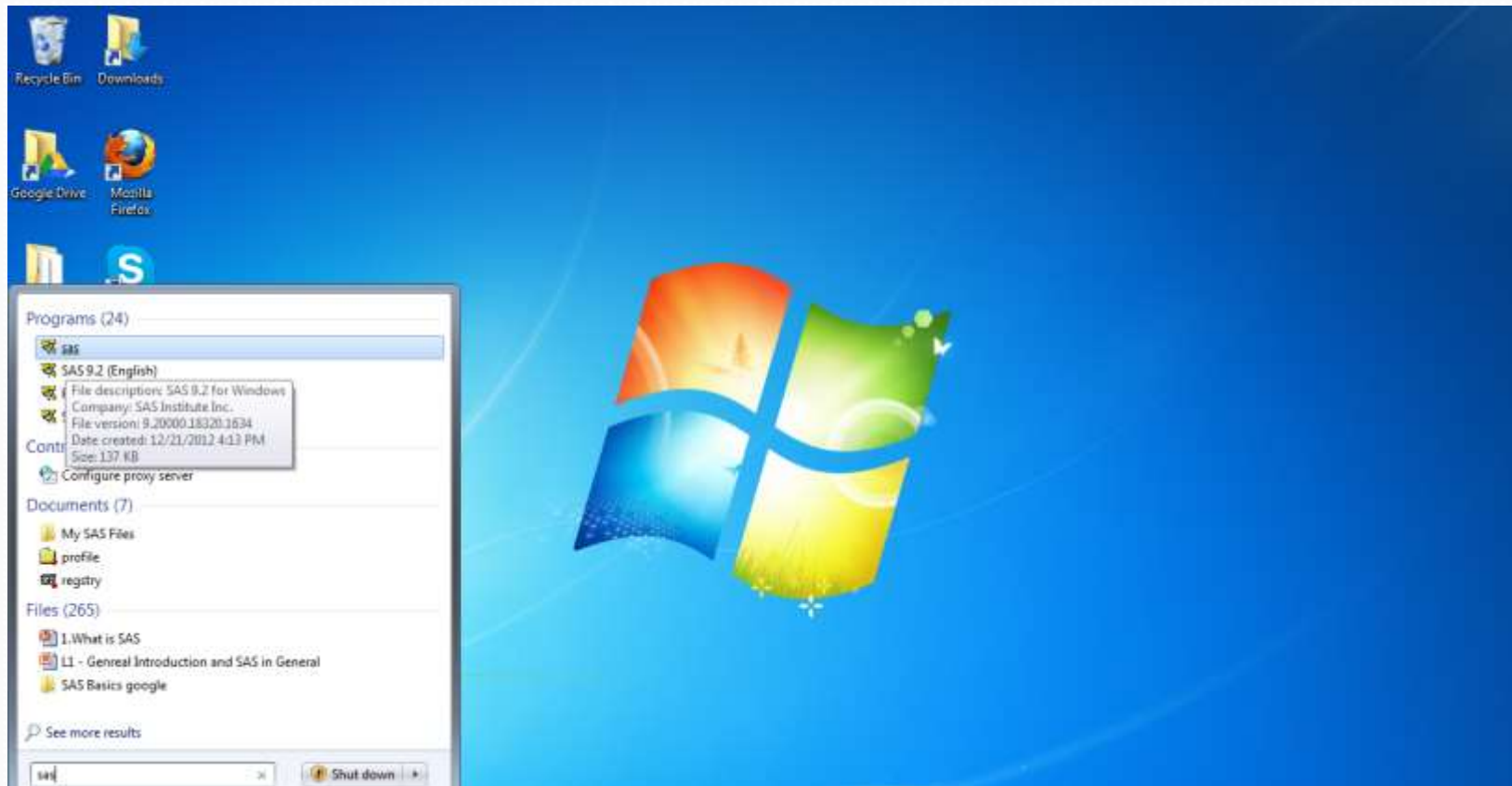
What is SAS?

- Collection of modules that are used to process and analyze data.
- Developed in the **early 1970s at North Carolina State University**
- Originally intended for management and analysis of agricultural field experiments
- Now the most widely used statistical software
- Used to stand for “Statistical Analysis System”, now it is not an acronym for anything
- Pronounced “sass”, not spelled out as three letters.

Various Industries use SAS

- Casinos
- Communications
- Education
- Financial Services
- Government
- Health Insurance
- Health Care Providers
- Hotels
- Insurance
- Life Sciences
- Manufacturing
- Media
- Oil & Gas
- Retail
- Travel & Transportation
- Utilities

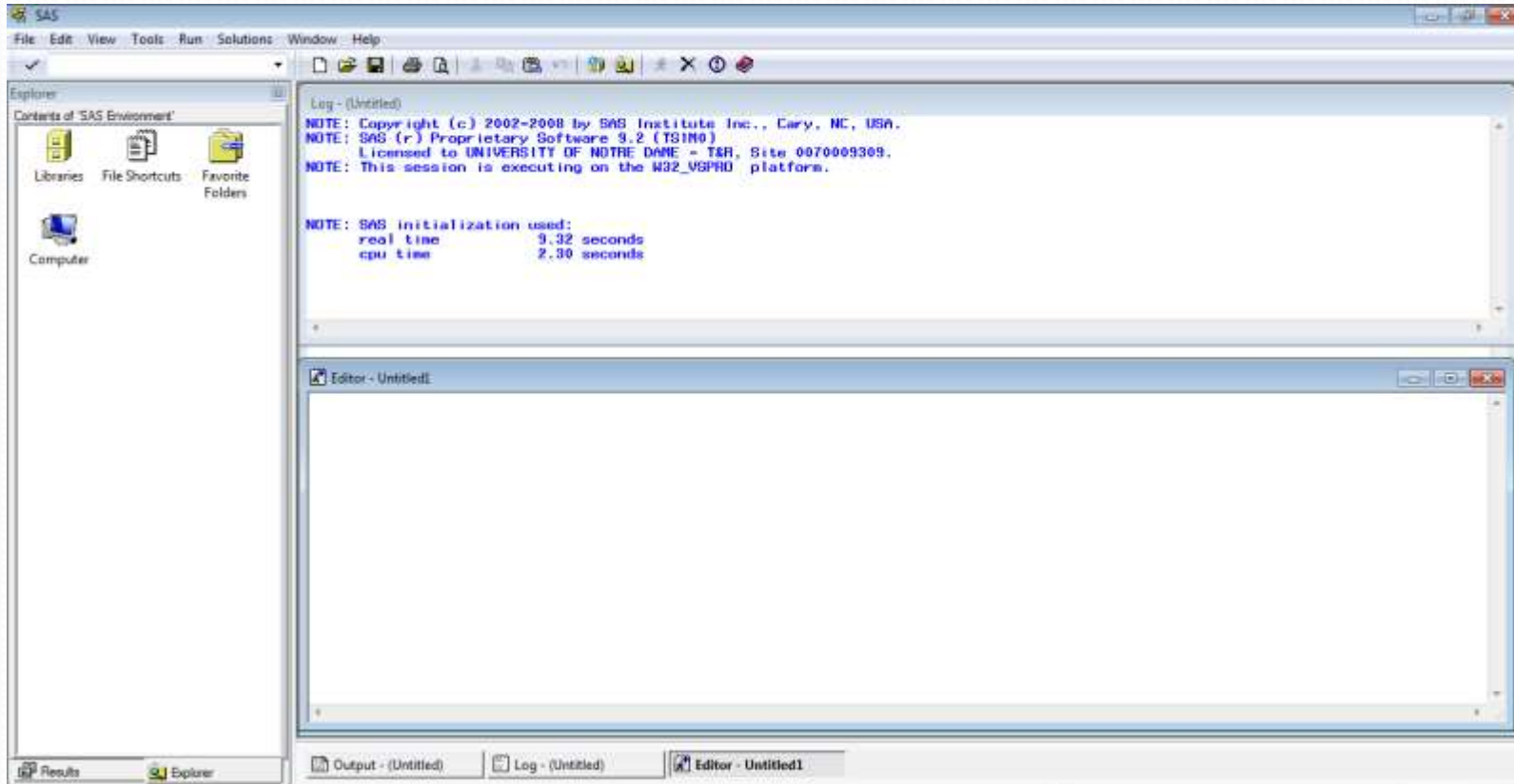
Accessing SAS



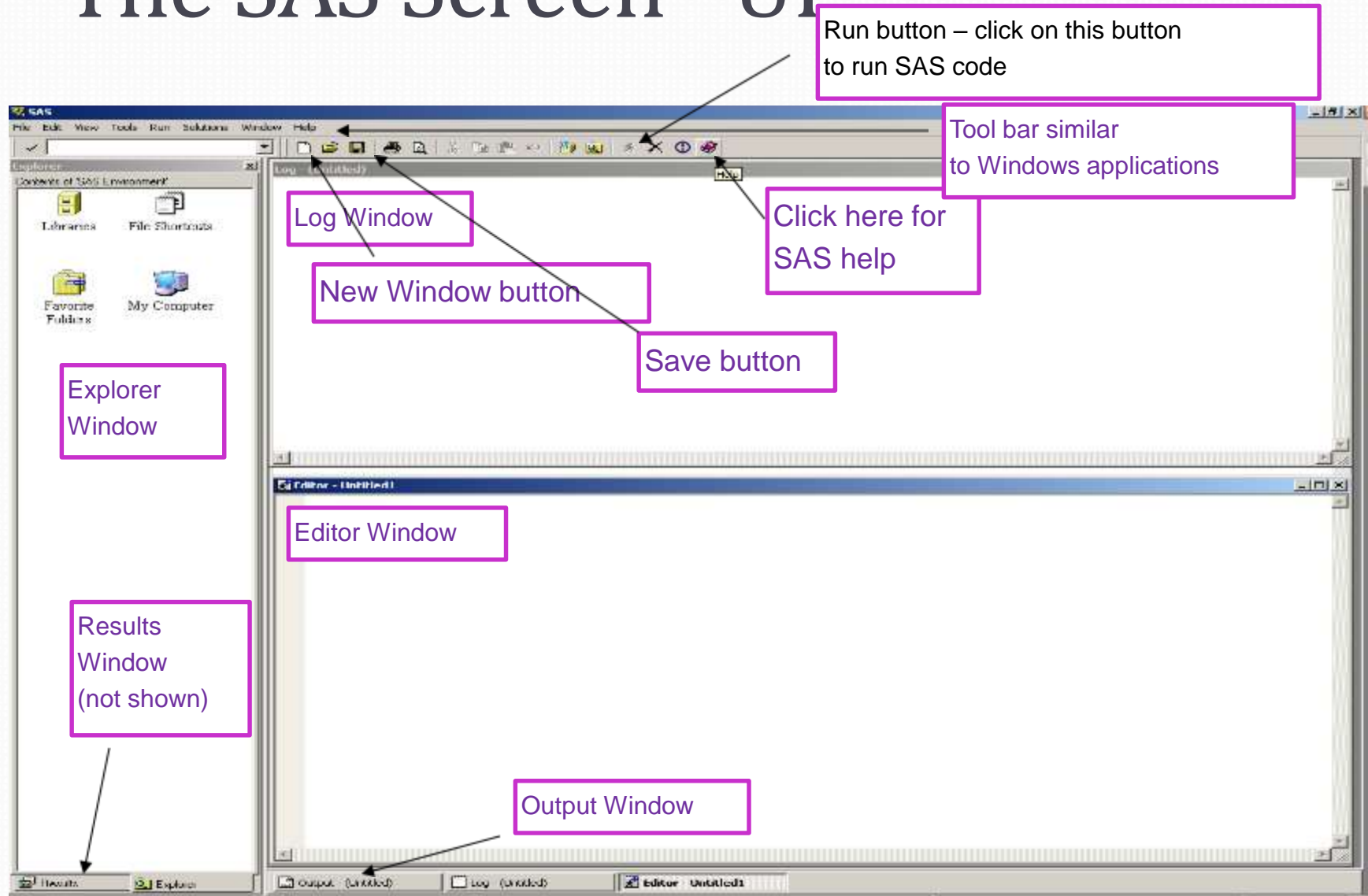
Accessing SAS – on linux(optional)

1. Type `sas` . This opens the SAS “display manager”, which consists of three windows (program, log, and output). Some procedures must be run from the display manager.
2. Type `sas -nodms` . You will be prompted for each SAS statement, and output will scroll by on the screen.
3. Type `sas -stdio` . SAS will act like a standard UNIX program, expecting input from standard input, sending the log to standard error, and the output to standard output;
4. Type `sas filename.sas` . This is the batch mode of SAS -your program is read from `filename.sas`, the log goes to `filename.log` and the output goes to `filename.lst`.

The SAS Screen



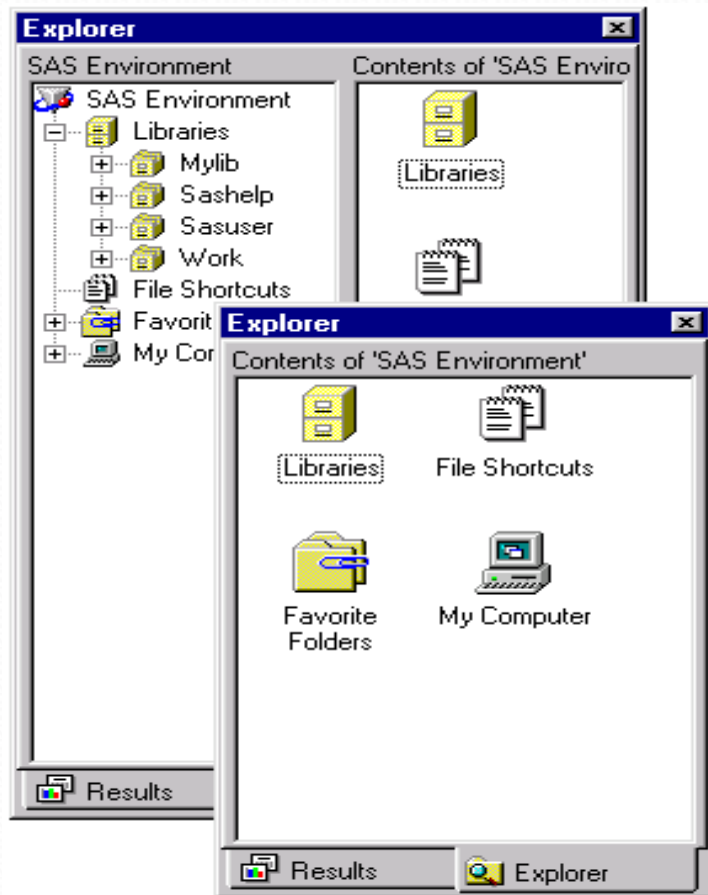
The SAS Screen - UI



Lab:

- Open SAS from start button. Identify below windows in SAS
- Explorer window
 - Where is it?
 - What is the symbol?
 - What does it contain?
- Results window
 - Where is it?
 - What is the symbol?
 - What does it contain?
- Program Editor or Editor
 - Where is it?
 - What is the symbol?
 - What does it contain?
- Log window
 - Where is it?
 - What is the symbol?
 - What does it contain?
- Output window
 - Where is it?
 - What is the symbol?
 - What does it contain?

Explorer

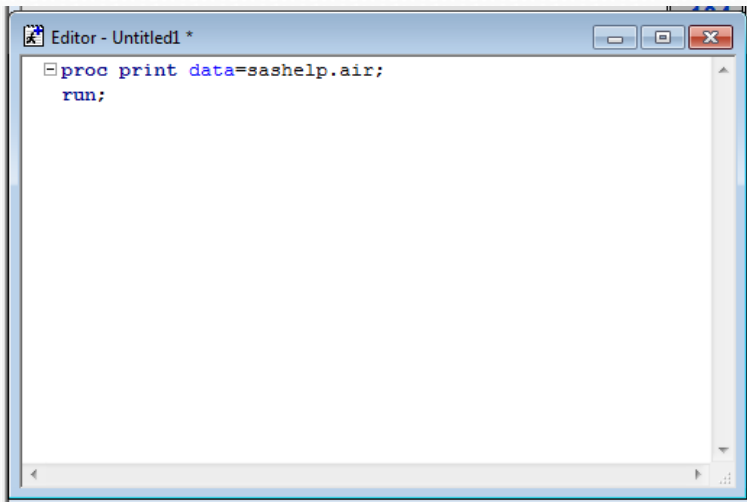


Where is Explorer in SAS?
What do you see in your SAS Explorer?

1. View and manage SAS files stored in SAS data libraries
2. Create new libraries and SAS files Open SAS files
3. File management tasks such as moving, adding , deleting files
4. Create shortcuts to files

Details later

Editor Window



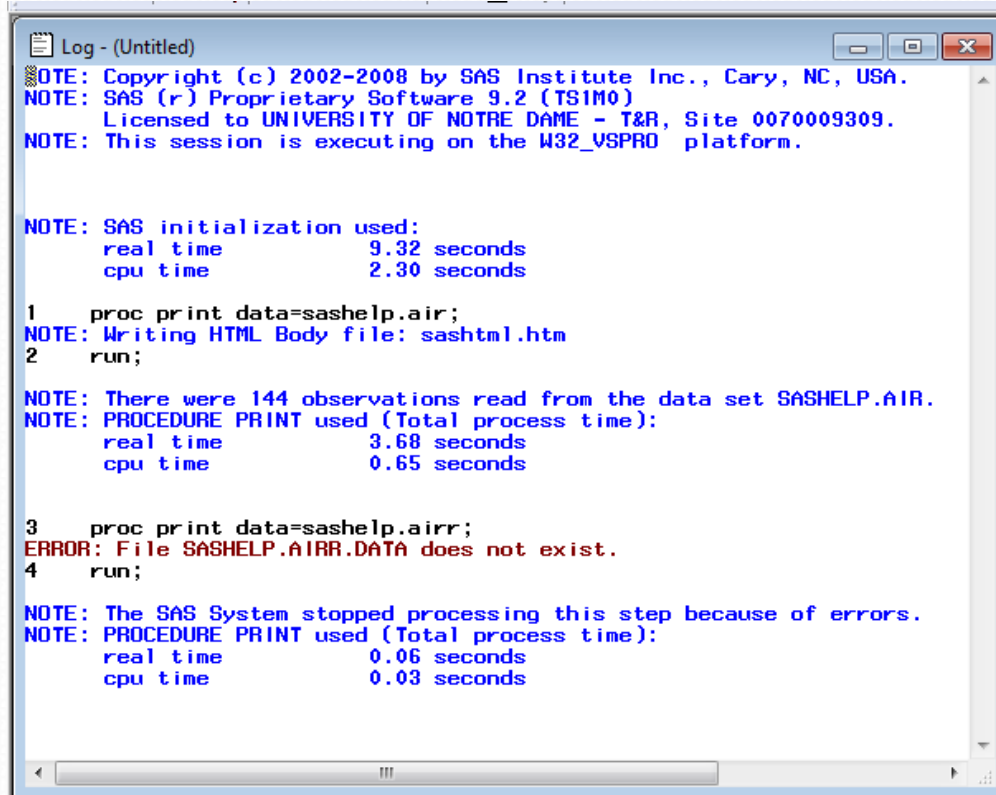
Where is editor in SAS?

1. Opening SAS program
2. Entering, editing and submitting SAS programs
3. Using the command line or menus
4. Clearing the contents
5. Support for keyboard shortcuts
6. Color coding and syntax checking of SAS language

Write this script in editor;

```
proc print data=sashelp.air;  
run;
```

Log window



The screenshot shows a SAS Log window titled "Log - (Untitled)". It contains the following text:

```
NOTE: Copyright (c) 2002-2008 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.2 (TS1M0)
      Licensed to UNIVERSITY OF NOTRE DAME - T&R, Site 0070009309.
NOTE: This session is executing on the W32_VSPRO platform.

NOTE: SAS initialization used:
      real time      9.32 seconds
      cpu time       2.30 seconds

1  proc print data=sashelp.air;
NOTE: Writing HTML Body file: sashtml.htm
2  run;

NOTE: There were 144 observations read from the data set SASHELP.AIR.
NOTE: PROCEDURE PRINT used (Total process time):
      real time      3.68 seconds
      cpu time       0.65 seconds

3  proc print data=sashelp.airr;
ERROR: File SASHELP.AIRR.DATA does not exist.
4  run;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
      real time      0.06 seconds
      cpu time       0.03 seconds
```

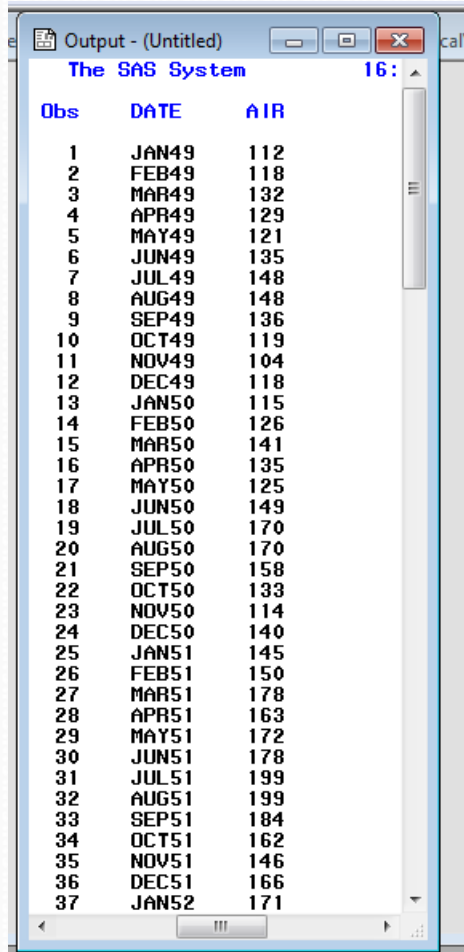
Where is log window in SAS?

1. The Debug window
2. Displays the messages about SAS session and the program that is submitted.

Write this script in editor and see the log file;

```
proc print
data=sashelp.airr;
run;
```

Outputs window



The screenshot shows a SAS window titled "Output - (Untitled)" with a subtitle "The SAS System". It displays a table with three columns: "Obs", "DATE", and "AIR". The data represents monthly values for the years 1949 through 1952. The window has standard Windows-style controls (minimize, maximize, close) and a scrollbar on the right.

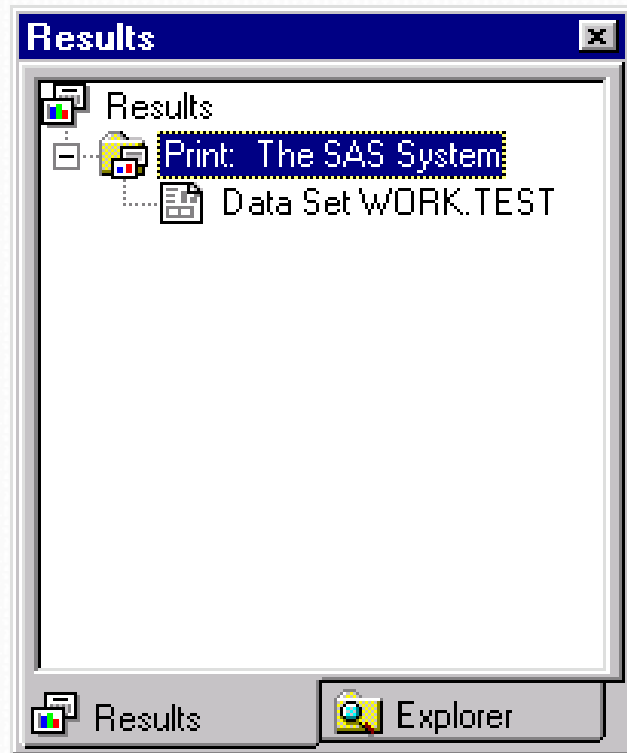
Obs	DATE	AIR
1	JAN49	112
2	FEB49	118
3	MAR49	132
4	APR49	129
5	MAY49	121
6	JUN49	135
7	JUL49	148
8	AUG49	148
9	SEP49	136
10	OCT49	119
11	NOV49	104
12	DEC49	118
13	JAN50	115
14	FEB50	126
15	MAR50	141
16	APR50	135
17	MAY50	125
18	JUN50	149
19	JUL50	170
20	AUG50	170
21	SEP50	158
22	OCT50	133
23	NOV50	114
24	DEC50	140
25	JAN51	145
26	FEB51	150
27	MAR51	178
28	APR51	163
29	MAY51	172
30	JUN51	178
31	JUL51	199
32	AUG51	199
33	SEP51	184
34	OCT51	162
35	NOV51	146
36	DEC51	166
37	JAN52	171

1. Displays the listing output in the output window
2. It automatically opens or moves to the front of display when output is created.

Write this script in editor and see output file;

```
proc print data=sashelp.air;  
run;
```

Results window



1. Helps navigate and manage output
2. View, save, and print individual items of output.

Comments and Help menu

There are two styles of comments you can use:

1. One starts with an asterisk (*) and ends with a semicolon (;).
2. The other style starts with a slash asterisk (/*) and ends with an asterisk slash (*/).



Other options: Results in html output

Lab: My first SAS Program

- My first SAS program– “Hello world”

```
proc print data=sashelp.air;  
run;
```

- Submit the program
- Write this SAS script in your editor

```
data income_data;  
input income expenses;  
cards;  
1200 1000  
9000 600  
;  
run;  
Proc print data=income_data;  
Run;
```


Lab

- Explorer window
 - What does it contain now ?
- Results window
 - What does it contain now ?
- Program Editor or Editor
 - What does it contain now ?
- Log window
 - What does it contain now ?
- Output window
 - What does it contain now ?

Lab: My first SAS program

- Run below code

```
proc print data=sashelp.airr;  
run;
```

- How to diagnose & correct errors

- Run below code

```
proc content data=sashelp.air;  
run;
```

- Identify the errors if any
- Print income_data
- Close SAS & open it again
- Print income_data

SAS Libraries

- By defining a library, you indicate the location of your SAS files to SAS
- Collection of SAS files such as SAS data sets and catalogs
- To access a library, you need to assign it a name (also known as a libref, or library reference)
- After assigning a library name you'll work with SAS data sets in a library
- In the Windows and Unix environments, SAS library is typically a group of SAS files in the same folder or directory.
- How to define a library?
 - Using Interactive window
 - Libname Statement

Temporary and permanent libraries

Temp Library

- Storing files temporarily
 - If you don't specify a library name when you create a file or
 - If you specify the library name WORK
 - Then the file is stored in the temporary SAS data library
- Last only for the current SAS session

Permanent library

- Storing files permanently
 - Specify a library name other than the default library name Work
- Available during subsequent SAS sessions

Defining a library -GUI

- Define a library name using interactive window
 - On the toolbar, click the New Library tool
 - Enter a Library Name
 - Browse to select the location
 - Enable at Startup check box -> OK.

Create a library name 'mylib' using interactive window

Libname Statement

Write a statement in a editor window

Syntax

```
LIBNAME libref 'SAS-library' ;
```

Example:

```
LIBNAME sales 'c:\salesdata\sas\2002';
```

- Create a library named 'mylib2' using the above syntax
- Create a library named '2mylib' using the above syntax
- Create a library named 'mylib2mylib' using the above syntax
- Create a library named '\$mylib2' using the above syntax

Rules for assigning a Library

1. Limited to 8 characters long
2. Must start with letter or underscore only
3. Can be a combination of letters, numbers and underscore

Referencing files in SAS Libraries

As we know, to reference a SAS file we need to assign a **libref** (library reference) to the SAS library in which the file is stored.

- Referencing files from permanent library:
 - To reference permanent SAS data set, you use two-level name. **libref.filename**
 - The **libref** is name of the SAS data library that contains the file. The **filename** is the name of the file itself.
 - A period separates the **libref** and **filename**.
 - Example: **libref.filename**
- Referencing files from temporary library:
 - Specify default libref WORK, a period and the filename. e.g. work.ecg1
 - Alternatively, you can use one level name- the filename only.
e.g. ecg1

SAS default libraries (Self)

Following libraries are automatically assigned each time you start SAS

Sashelp:

- Permanent library
- Contains sample data and other files that control how SAS works at our site.
- This is a read-only library

Sasuser:

- Permanent library
- Contains SAS files that store personal settings/ our own files

Work:

- Temporary library
- Last only for the current SAS session

Lab

- Print a data file from SAS default library
`Proc print data= <> ; run;`
- Print a data file from SASuser library
- Print a dataset from work library.
- Try work. & without work.
- Create a new library “_data_”
- Create a new library “3data_”
- Create a new library “data”

Step-2: SAS datasets and variables

Contents

- SAS programs
- Data step
- PROC Step
- Writing a SAS program and debugging it
- SAS Data sets
- Data sets properties
- Variables in a data set
- Types of variables
- Attributes of variables
- Create data using Data statement

SAS Program

- Sequence of statements executed in order
- Used to access, manage, analyze and present the data
- Contains statements, expressions, functions and CALL routines, options, formats, and informats
- Simplified programming with built in programs known as **SAS Procedures** (pre written codes)
 - PROC REG, PROC PRINT, PROC ARIMA etc.,
- All programs are already written in SAS, we just need to call the right procedure
- There are a few rules to follow when writing SAS programs.

Characteristics and layout

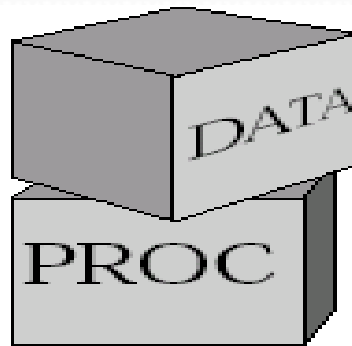
- SAS programs are made up of SAS statements.
- A SAS statement has two important characteristics:
 1. It usually begins with a SAS keyword.
 2. It always ends with a semicolon
- Layout of SAS program:
 1. They can begin and end anywhere on a line
 2. One statement can continue over several lines
 3. Several statements can be on a line.
 4. SAS statements are NOT case sensitive.
 5. Blanks or special characters separate the "words" in a SAS statement.

Lab

- Print a dataset in SAS help library
- Proc print data=<>; run;
- Try these properties of SAS statements
 1. They can begin and end anywhere on a line - Try it
 2. One statement can continue over several lines - Try it
 3. Several statements can be on a line. - Try it
 4. SAS statements are NOT case sensitive. - Try it

Components of SAS Programs

- SAS programs are constructed from two basic building blocks:
 1. DATA steps
 2. PROC steps
- These two types of steps, alone or combined, form all SAS programs.
- Generally, a step ends with a RUN statement or when a new DATA or PROC step begins.



DATA step

- Begins with the keyword DATA
- Creates or modifies the data set
- Produces custom designed reports
- Using data step we can
 - Put our data into a SAS data set
 - Compute values & create new fields
 - Check for and correct errors in our data
 - Produce new SAS data sets by sub setting, merging, and updating existing data sets

Creating a SAS data

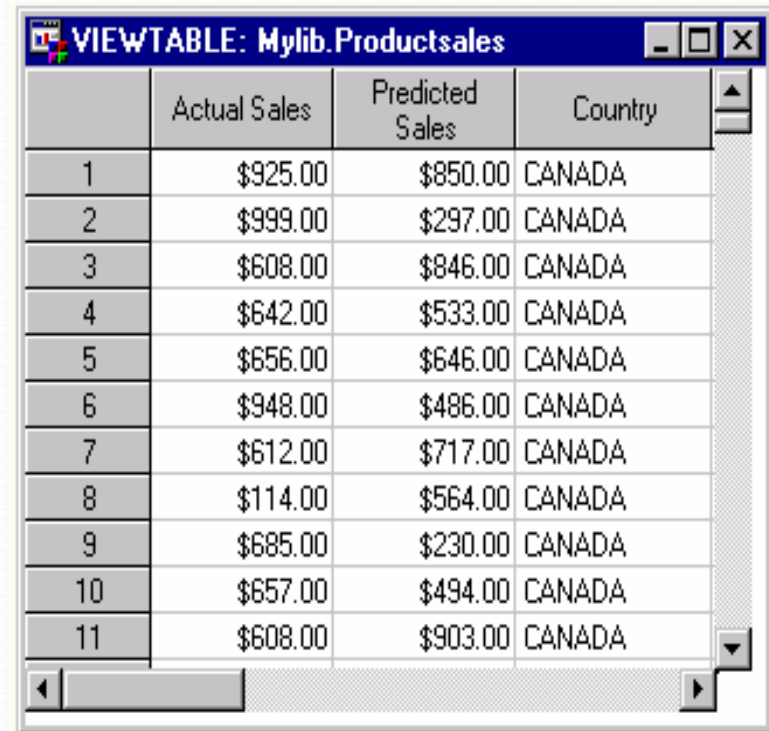
```
data <libname>.<datasetname>;  
Input <var1> <var2>....<varp>;  
Cards;  
v11 v21 ....vp1  
.....  
v1n v2n ....vpn  
;  
run;
```

Creating a SAS data

```
data mylib.stock;  
input volume price;  
cards;  
1200 14  
1400 14.5  
1250 16  
1470 13.5  
1290 18  
1609 19.5  
1809 17  
1123 16.5  
;  
run;
```

Data portion of SAS data set

- Observations (Rows):
 - Collections of data values that usually relate to a single object
 - Can store any number of observations.
- Variables (Columns):
 - Collections of values that describe a particular characteristic
 - Can store thousands of variables.



	Actual Sales	Predicted Sales	Country
1	\$925.00	\$850.00	CANADA
2	\$999.00	\$297.00	CANADA
3	\$608.00	\$846.00	CANADA
4	\$642.00	\$533.00	CANADA
5	\$656.00	\$646.00	CANADA
6	\$948.00	\$486.00	CANADA
7	\$612.00	\$717.00	CANADA
8	\$114.00	\$564.00	CANADA
9	\$685.00	\$230.00	CANADA
10	\$657.00	\$494.00	CANADA
11	\$608.00	\$903.00	CANADA

Lab

- Use data step and create a dataset named “reallybigdatasetnamethis12345678”;
- Use data step and create a dataset named “123data”;
- Use data step and create a dataset named “\$data”;
- Use data step and create a dataset named “_data”;
- Use data step and create a dataset named “_data123”;

SAS Dataset name rules

1. Can be 1 to 32 characters long
2. Must begin with a letter (A–Z, either uppercase or lowercase) or an underscore (_)
3. Can be a combination of numbers, letters, or underscores.

Examples: admit2, _test_1, CLINTRIAL

Lab

- Use data step and create a variable named “reallybigvarnamethis123456789101112”; in a dataset
- Use data step and create a variable named “123var”;
- Use data step and create a variable named “\$var”;
- Use data step and create a variable named “_var”;
- Use data step and create a variable named “_var123”;

SAS Variables

Columns in the SAS datasets are called SAS variables.

Attributes of SAS variables: Attribute information includes the variable's

1. Type
2. Length
3. Format
4. Informat
5. Label.

SAS variable name rules: (Same as SAS dataset name)

1. Limited to 32 characters long
2. Must start with letter or underscore
3. Can be a combination of letters (Uppercase / lowercase, A-Z) , numbers and underscore

Lab: Creating a sample dataset using script

- Run below script

```
Data student;  
input height weight;  
Cards;  
120 55  
133 70  
140 78  
135 69  
;  
Run;
```

- See the properties of the created data, proc contents
- Add a new variable 'name' to the above dataset
- What is the error? (Use \$)
- See proc contents data
- Print data

Main Attributes of a variable

- Length:
 - Character variables can be up to 32,767 bytes long.
 - All numeric variables have a default length of 8.
- Label:
 - Consists of descriptive text up to 256 characters long
 - Labels are used instead of variable names in some reports
 - Labels are used for the column headings in the VIEWTABLE window
 - By default, variable name is assigned as a label
- Formats:
 - Affect the way data values are written
 - SAS offers a variety of character, numeric, and date and time formats
 - Can create and store your own formats
 - Storing
- Informats:
 - Read data values in certain forms into standard SAS values
 - Determine how data values are read into a SAS data set
 - Printing

Main types of SAS variables

Character variables:

1. Can contain any value
2. Use Blank to represent the missing values
3. Can be up to 32K

Numeric variables:

1. Can contain only numeric values
(digits 0 through 9, +,-,., E or scientific notations)
2. Use a single period (.) to represent the missing values
3. Have a default length of 8.
4. Numeric values are stored as floating point number as 8 bytes of storage, unless you specify another length.

Step-3: Importing data into SAS

Contents

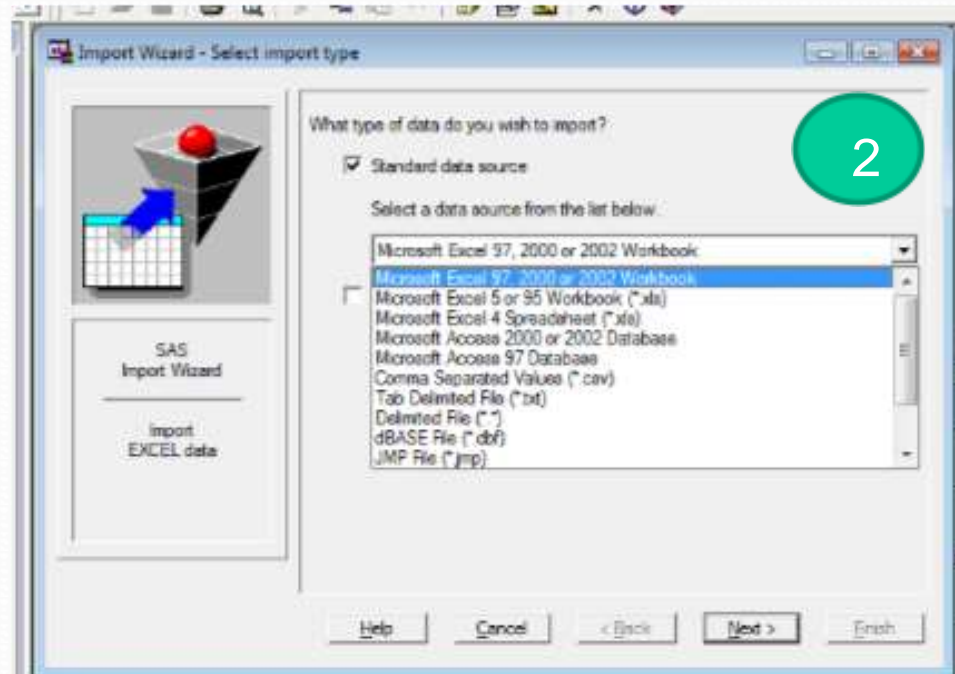
- Data set creation using SAS program(data step)
- Using import wizard
 - Importing Excel file
 - Importing CSV file
 - Importing delimited file
- Converting files from other packages such as dbf, xls, wkn via proc import.

Reading data using import wizard

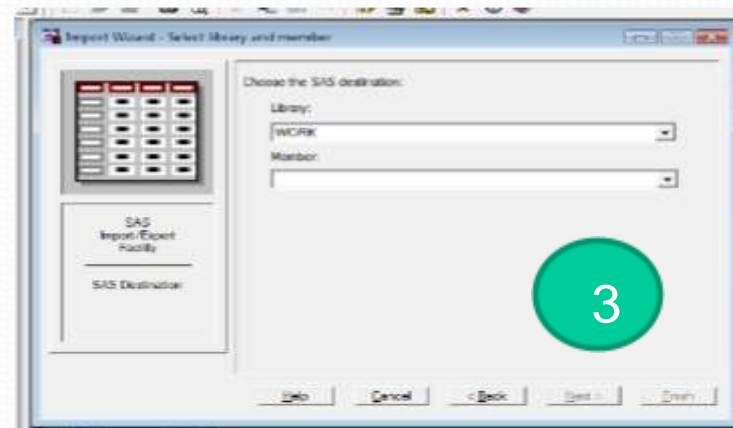
1



2



3

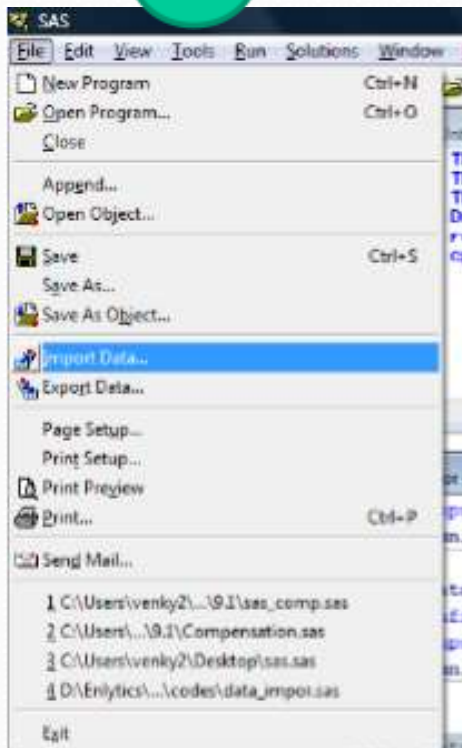


Lab

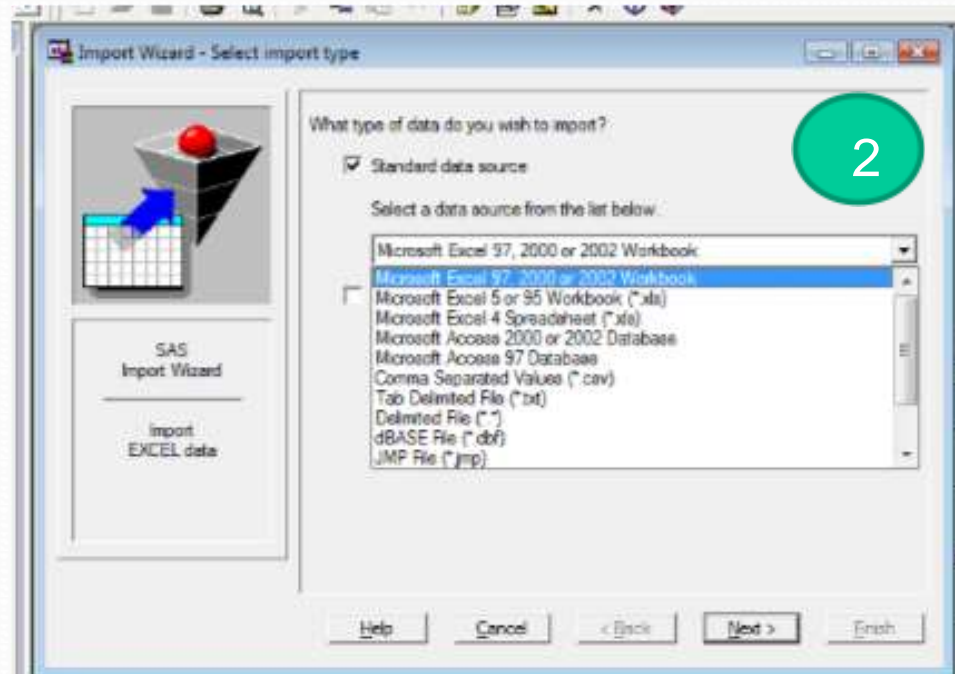
- Import Survey_data from Survey_data.xls file
- Name the output file as survey_2012;
- Import the file into work library & current library
- Print the contents of the data
- Are all the variables present in the SAS dataset?
- Import 'Cust_survey_old' data from Survey_data1.xls

Import CSV and TAB files

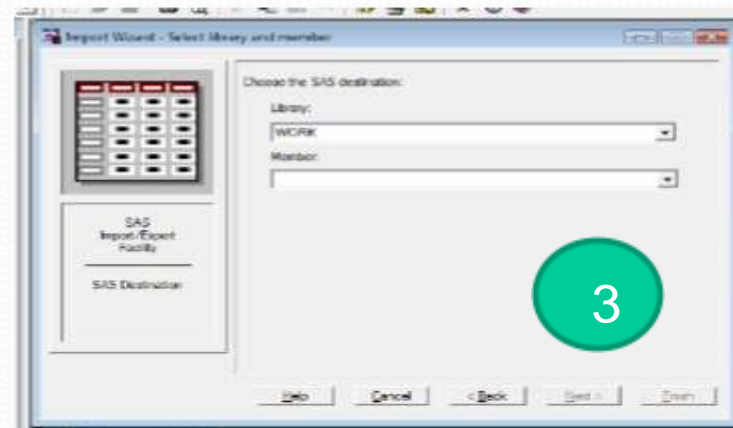
1



2



3



Lab

- Import client_data.csv file
- Name the output file as client_2012;
- Import the file into work library & current library
- Print the contents of the data
- Are all the variables present in the SAS dataset?
- Import clinet_manager.txt file
- Import price web.csv file
- See the contents
- **Save the import code in local folder**

Reading data sets using 'PROC import'

```
PROC IMPORT DATAFILE= "<location of file>\sample_data.csv"  
OUT= testdat.excelSAS1  
DBMS=CSV  
REPLACE;  
RUN;
```

Where

DATAFILE is the name and the location of file you want to read.

OUT is the name of the SAS data set you want to create

DBMS is *identifier*; specifies the type of data to import. Valid identifiers for delimited files are DLM, JMP, and CSV.

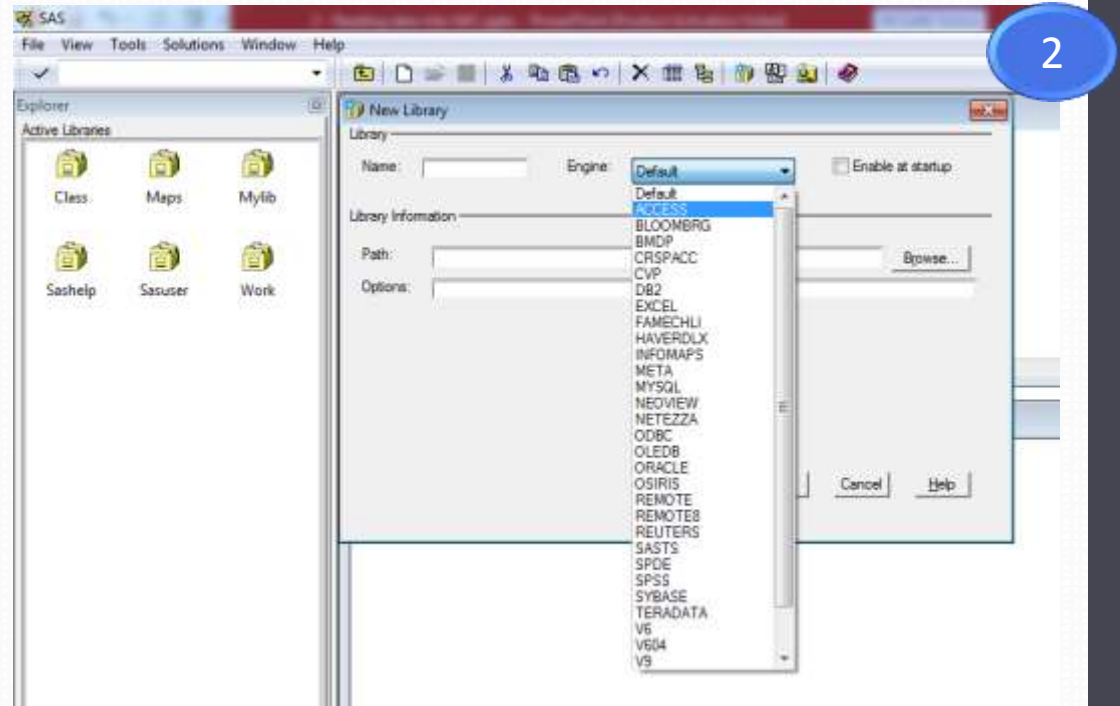
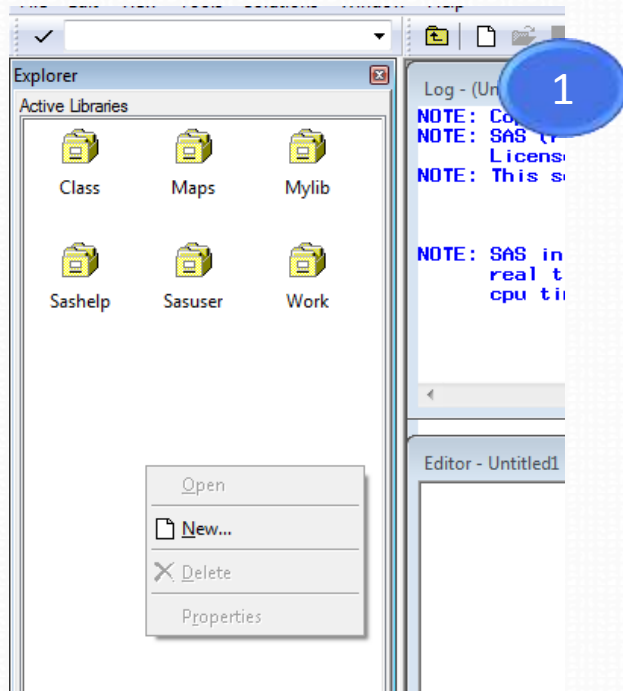
REPLACE overwrites an existing SAS data set. If you do not specify the REPLACE option

Lab: Using Proc Import script

- Import Healthcare data
- Print the contents of the data
- Are all the variables present in the SAS dataset?
- Try with and without replace options
- Import cars data
- Import credit risk.txt data

Working with DB files

What if we have all these tables in one database?



Lab

- Create a library out of All_Survey_DB.mdb
- Print the contents of the files
- Copy relevant tables into your library
- Print client_manager data
- Try opening access file using access now

Reading data from SAS datasets

- How to create a new SAS data set out of SAS data set?
- How do we create survey2 from survey data?

Data libname.new;

Set libname.old;

Run;

Lab: Creating new variables

- Create a new dataset and add a new variable score1 (Response + Quality)

```
data work.survey_v2;  
set work.survey;  
score1=(response+quality)/2;  
run;
```

- Create a new variable diff, the difference between overall rating and score1
- Update the same dataset with a new variable inter_score which is average of Communication, Response, Quality score
- Print all the observations with overall score greater than or equal to 4

Manipulating SAS data sets & Variables

Contents

- Recap
- Creating and managing variables
- Drop keep statements
- Conditional data processing
- SAS functions

Lab: Creating new variables

- Create a new dataset and add a new variable score1 (Response + Quality)

```
data work.survey_v2;  
set work.survey;  
score1=(response+quality)/2;  
run;
```

- Create a new variable diff, the difference between overall rating and score1
- Update the same dataset with a new variable inter_score which is average of Communication, Response, Quality score
- Print all the observations with overall score greater than or equal to 4

Recap

- Create a new library
- What are the already existing datasets in your library?
- Import **market_data_one** data into your library using import script
- Print the contents of the data
 - How many observations are there in the dataset?
 - How many variables?
 - Identify non numeric variables
- Print first ten observations of the data
- Create a new field `general_questions`. If we subtract `num_custom_questions` from `num_form_fields`, we get `general_questions`

Creating new variable - using if then else

Creating a new variable using if then else

```
data datalib.market_one_v1;  
set datalib.market_one;  
if budget <= 3000 then budget_ind='low';  
else if budget >= 100000 then budget_ind='high';  
else budget_ind='medium';  
run;
```

Lab: Creating a variable- using if then else

- Create a new variable `asset_ind`.
 - If the number of assets are greater than zero then `asset_ind` takes the value yes otherwise NO
- See proc contents
- Print first ten observations
- Create `reach_ind` which takes values 1, 2 and 3 when reach is less than 33, 33 to 66 and 67 to 100.
- Print first ten observations

Drop and keep variables

- What if I'm not interested in all the fields?
- Creating a new data set with less number of fields.

Using drop statement

```
data datalib.market_one_v2;  
set datalib.market_one (drop=name start_date end_date) ;  
run;
```

Using Keep statement

```
data datalib.market_one_v3;  
set datalib.market_one (keep=id name budget) ;  
run;
```

Lab: Drop and keep variables

- Import market_dk data
- See the contents
- Var1, var2, var3 are of no use for the analysis
- Create a new data by dropping the unnecessary variables by using drop statement
- Create a new data by keeping only necessary variables by using keep statement

Sub setting the data

- Sometimes we need a subset of the data for further analysis

```
data datalib.market_one_v4;  
set datalib.market_one;  
where vertical='Technology';  
run;
```

or

```
data datalib.market_one_v4;  
set datalib.market_one;  
if vertical='Technology';  
run;
```


Lab Sub-setting the data

- Create a data set named “good_reach_camp” contains all the camps where reach is more than 50 percent
- Create using where statement
- Create the same using if statement
- Is there any difference between where and if?
- Print a subset of data:
 - Try to print the data where reach is more than 90
 - Use if and print again
 - if
 - NOTE: There were 7843 observations read from the data set DATALIB.MARKET_ONE.
 - NOTE: The data set DATALIB.MARKET_ONE_V4 has 4679 observations and 12 variables.
 - **Where**
 - NOTE: There were 4679 observations read from the data set DATALIB.MARKET_ONE. WHERE vertical='Technology';
 - NOTE: The data set DATALIB.MARKET_ONE_V4 has 4679 observations and 12 variables.

Major differences between where and if

Where statement	IF statement
Selects observations before they are brought into the program data vector, making it a more efficient programming technique.	Works on observations after they are read into the program data vector.
Cannot be executed conditionally as part of an IF statement	The sub setting IF statement can be executed conditionally
Can be used in SAS procedures	Cannot be used in SAS procedures to subset observations for browsing or editing.

SAS functions - Numeric

- Sum, min, max, avg functions

```
data DATALIB.MARKET_two_v1;  
set DATALIB.MARKET_two;  
sum_two=Sum(Webinar,White_Paper);  
min_two=min(Webinar,White_Paper);  
max_two=max(Webinar,White_Paper);  
Avg_two=mean(Webinar,White_Paper);  
run;
```

Lab: Functions

1. There is some error in the number of assets field.
2. Create a variable num_asset to verify number of assets formula. Number of assents is nothing but the sum of White_Paper,Webinar,Software_Download,Free_Offer,Live_Event,Case_Study
3. Find the absolute difference between two variables using $\text{diff}=\text{abs}(x2-x1)$;
4. Print all observations where diff is not equal to zero

String functions

```
data datalib.market_one_v5;  
set datalib.market_one;  
vertical_ind=substr(vertical,1,1);  
length_name=length(name);  
trim_name=trim(name);  
run;
```

Lab: String functions

- Create name_new by taking first 20 characters of the name variable
- Create a new variable by converting the name of the campaign into uppercase (use UPCASE function)
- Create a flag variable that takes value 1 If campaign starting month is not equal to campaign ending month

Date Functions

```
data datalib.market_one_v5;  
set datalib.market_one;  
start_month=month(start_date);  
start_date1=day(start_date);  
Duration_days=INTCK('day',start_date,end  
_date);  
run;
```

Lab: Date functions

- How many campaigns started in January?
- How many campaigns started on Saturday or Sunday?
- What is the duration of each campaign? Create three variables
 - Duration_days
 - Duration_months
 - Duration_weeks
- How many market campaigns run more than 6 months

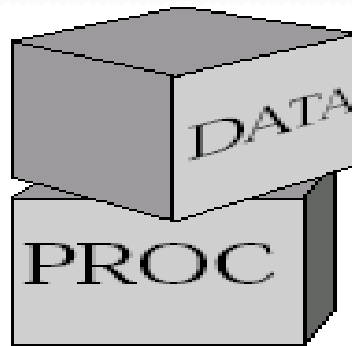
Step-4: Basic Procedures and Functions

Contents

- The PROC step
- PROC CONTENTS data
- PROC SORT and options
- PROC SQL
- PROC Gplot

Components of SAS Programs

- SAS programs are constructed from two basic building blocks:
 1. DATA steps
 2. PROC steps
- These two types of steps, alone or combined, form all SAS programs.
- Generally, a step ends with a RUN statement or when a new DATA or PROC step begins.



PROC step

- Begins with the keyword PROC
- Pre-written routines that enable you to analyze and process the data
- Sometimes create SAS data sets that contain the results of the procedure
- Present the data in the form of a report

For example

- Create a report that lists the data
- Produce descriptive statistics
- Create a summary report
- Produce plots and charts
- Run regression analysis

Proc Contents

Proc contents describes the structure of the SAS data set. Gives following information -

- Data set level
 - Details of data like Name, creation date, Number of observations (weighted and un-weighted), Number of variables, File size and access permissions
- Variable level
 - Provides descriptive stats across each variable such as Name, Type (Character vs. Numeric), Length, Formats, Position and Labels
- Proc contents data= <<data name>>; run;
- Useful options :
 - **Short** – Outputs the list of variables in a row by row format.
Code : `proc contents data=test short;run;`
 - **Out=filename** - Creates a data set wherein each observation is a variable from the original data set.
Code : `proc contents data=test1 out=test2 noprint;run;`

Lab: Proc Contents

- Import Price_web_data.csv; See the data description.pdf file
- Print the contents
- Are all the variables in expected format?
- Print only field names of the dataset

```
proc contents data=class1.market_one short;  
run;
```

- Take the output of contents into a sas file, name it contents_out

```
proc contents data=class1.market_one out=con_out noprint;  
run;
```

Proc Sort

- Primarily used to sort the observation of your data by a certain variable or collection of variables.
- However, it can also be used to create a new data set, subset your data, rename, drop, or keep variables, and format or label variables.

```
proc sort data=class1.market_one out=new;  
by start_date ;  
run;
```

- Almost always a good idea to use the OUT= option when using proc sort to do anything except for a simple sort. Because Proc Sort automatically writes over your data set!

Lab: Proc Sort

- Sort market_one data by budget
- Sort market_one data by budget descending
- Sort data by date and reach
- Use market_one table and create a new table campan_names, sort name in alphabetical

Proc Sort

- Sorting and removing the duplicates

```
proc sort data=class1.market_one out=class1.names_table nodupkey;  
by name ;  
run;
```

- Sorting and removing the duplicate records

```
proc sort data=class1.market_one out=class1.names_table nodup;  
by name ;  
run;
```

Lab: Proc Sort

- Sort the dataset by date; don't overwrite the original data
- Create a new dataset which contains all the unique brands on a given date(A dataset where the brand name does not repeat on a given day)
- How many unique products are there in the dataset? Create a new data set with this unique list

Proc Sql

```
proc sql;  
  
create table class1.market_sql as  
select *  
from class1.Market_two  
where num_assets>0;  
  
quit;
```

Quit instead of run?

Lab: Proc Sql

- Import market_data_three
- Create a new table where budget is greater than zero
- In market_one data find the average budget and average reach for each vertical
- Create a new table by inner joining market_one, market_two join them on id

```
proc sql;  
create table class1.market_one_two as  
select *  
from class1.Market_one as a inner join class1.Market_two as b  
on a.id=b.id;  
quit;
```

- Create a new table by inner joining above table to market_three table, save the final table as market_final_data

Lab: Proc Sql

- Find the frequency of each brand(count number of times a brand appears),also average list price for each brand. Save this data as brand_data

```
proc sql;  
create table brand_data as  
select brand, count(rowid) as freq, mean(listprice) as  
avg_list_price  
from class1.price_data  
group by brand;  
quit;
```

- For each product find the number of appearances, average rating and total review count, name this data as product_data

Proc Gplot

- Scatter plot

```
proc gplot data= class1.market_final_data;  
plot budget_leads*budget;  
where budget < 100000;  
run;
```

Lab: Proc Gplot

- If Average price is less, then frequency will be more, if price is more then frequency will be less.
- Verify the above statement by drawing a scatter plot graph between freq and average list price in brand_data

Proc Gchart

- Vertical Bar chart

```
proc gchart data= class1.market_final_data;  
vbar vertical / type=sum sumvar=total_leads;  
Run;
```

- Vbar → X axis category
- Type → type of aggregation
- sumvar → Y axis variable
- The above script shows sum of leads for each vertical

```
proc gchart data= class1.market_final_data;  
vbar vertical / type=mean sumvar=total_leads;  
Run;
```


Lab: Proc Gchart

- Draw a vertical bar graph which shows all the brands and the average number of reviews(Use original price_data)
- Draw a horizontal bar graph for the same
- Draw a bar chart for number of products by site name
 - You may want to use count
 - Try freq if it doesn't work

Pie chart

```
PROC GCHART DATA=class1.price_data;  
    PIE category;  
RUN;
```

Lab: Pie chart

- There was a complaint that one in every four items listed on the site is out of stock. Is that true? Draw a pie chart to verify
- Draw a pie which shows the percentage of products with free shipping availability & non availability

Step-5: Combining Datasets in SAS

Combining data set

Combining datasets in a DATA step

- Concatenate Append

- SET

- Interleave

- SET ... BY

- Merge

- MERGE

- MERGE ... BY

Concatenation/Appending

Simply appending/staking of the data

- Eg: Month by month data concatenated to form year end data
- Concatenating data from various customers to make a master dataset

DATA1		DATA2		COMBINED
Year		Year		Year
1991		1991		1991
1992		1992		1992
1993	+	1993	=	1993
1994		1994		1994
1995		1995		1995
				1991
				1992
				1993
				1994
				1995

Demo: Appending Datasets

```
data store1;  
input customer $;  
cards;  
Mr-X  
Mr-A  
Mr-Z  
;  
data store2;  
input customer $;  
cards;  
Mr-B  
Mr-Y  
Mr-C  
;
```

```
data overall;  
set store1 store2;  
run;
```

```
proc print data=overall  
run;
```

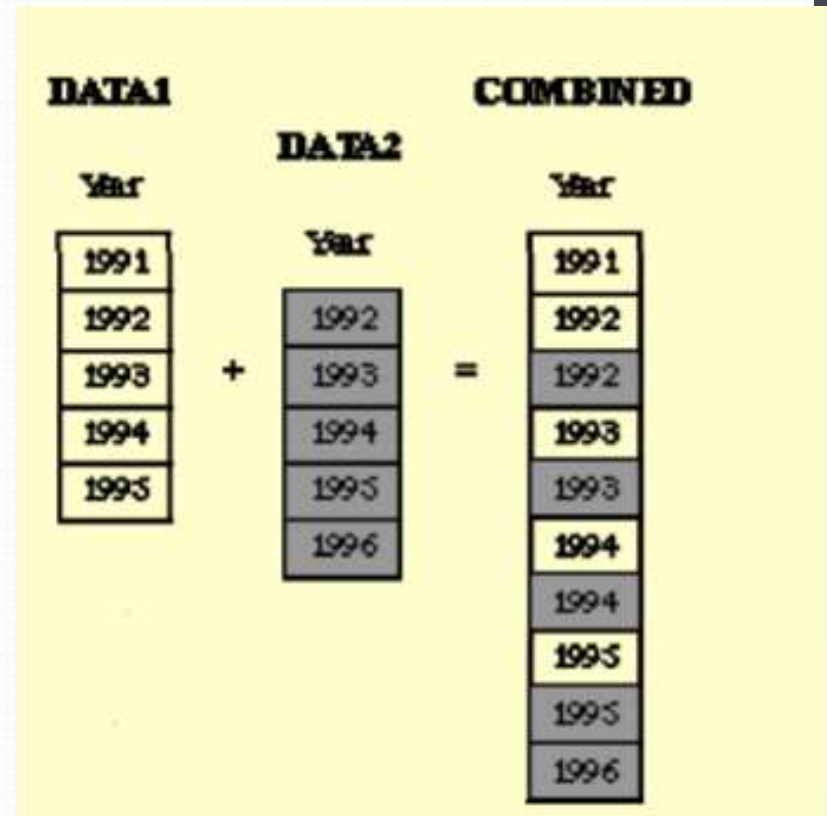
Lab: Concatenation

- Create three datasets out of price web data, based on brand Samsung, apple & others
- Create a new variable in apple dataset, disc_price_apple which is 20% of actual cost
- Concatenate the three datasets
- How many fields are there in the new dataset?
- Create four datasets out of telecom complaints data, based on status closed, open, Temporary Closed and others
- Keep only del number, customer email id and status
- Concatenate all these four datasets to form a new_overall dataset

```
data tech_mark ener_mark;  
set datalib.market_one;  
if vertical='Technology' then output tech_mark ;  
else if vertical='Energy' then output ener_mark;  
run;
```


Interleave

- Appending simply appends
- We may want to maintain the order rather than just appending
- Interleave – Ordered stacking



Demo: Interleave

```
data store1;  
input customer $;  
cards;  
Mr-X  
Mr-A  
Mr-Z  
;  
data store2;  
input customer $;  
cards;  
Mr-B  
Mr-Y  
Mr-C  
;
```

```
data overall;  
set store1 store2;  
by name;  
run;  
  
proc print data=overall;  
run;
```

Demo: Interleave

```
data store1;  
input customer $;  
cards;
```

Mr-X

Mr-A

Mr-Z

;

```
data store2;  
input customer $;  
cards;
```

Mr-B

Mr-Y

Mr-C

;

```
proc sort data= store1;  
by customer;  
run;
```

```
proc sort data= store2;  
by customer;  
run;
```

```
data overall;  
set store1 store2;  
run;
```

```
proc print data=overall;  
run;
```

LAB Inter leaving Dataset

- Crate four datasets out of telecom complaints data, based on status closed, open, Temporary Closed and others
- Keep only del number, customer email id and status.
- Interleave by all these four datasets by del number to form a new_overall dataset
- Create there datasets out of price web data, based on brand Samsung, apple & others
- Crate a new variable in apple dataset, disc_price_apple which is 20% of actual cost
- Concatenate the three datasets based on date
- How many fields are there in the new dataset?

From set to Merge

- Merging is similar to joins but not exactly same as joins
- Options need to be used in proper order to form the desired dataset

```
data students1;  
input name $ maths;  
cards;  
Ram 78  
Robert 90  
Raheem 80  
Gopi 75  
Anil 60
```

```
;  
proc sort data=students1; by name; run;  
proc sort data=students2; by name; run;
```

```
data two_sub;  
set students1 students2;  
by name;  
run;  
Proc print data=two_sub; run;
```

```
data students2;  
input name $ english;  
cards;  
Ram 55  
Robert 70  
Raheem 60  
Fred 75  
Alex 50
```

Replace Set by Merge

Matched Merging

```
data two_sub;  
set students1 students2;  
by name;  
run;
```

Obs	name	maths	english
1	Alex	.	50
2	Anil	60	.
3	Fred	.	75
4	Gopi	75	.
5	Raheem	80	.
6	Raheem	.	60
7	Ram	78	.
8	Ram	.	55
9	Robert	90	.
10	Robert	.	70

```
data two_sub;  
Merge students1 students2;  
by name;  
run;
```

Obs	name	maths	english
1	Alex	.	50
2	Anil	60	.
3	Fred	.	75
4	Gopi	75	.
5	Raheem	80	60
6	Ram	78	55
7	Robert	90	70

Merging

Merge looks for the matching variables and created a consolidated merged dataset

```
DATA output-SAS-data-set;  
  MERGE SAS-data-set-1 SAS-data-set-2;  
  BY variable(s);  
RUN;
```

Where,

- Output–SAS data set names the data set to be created.
- SAS-data-set-1 and SAS-data-set-2 specify the data sets to be read.
- Variable(s) specifies one or more variables that are used to match observations.

LAB: Matched Merging

- Import Orders data & spot data into two different data sets from TV commercial data
- Sort both data sets based on iSCI/AD-iD
- Merge orders and slots based on iSCI/AD-iD
- How many observations are there in the consolidated dataset

```
data two_sub;  
Merge students1 students2;  
by name;  
run;
```


Merge With Conditions

```
data students1;  
input name $ maths;  
cards;  
Ram 78  
Robert 90  
Raheem 80  
Gopi 75  
Anil 60  
;
```

```
data students2;  
input name $ english;  
cards;  
Ram 55  
Robert 70  
Raheem 60  
Fred 75  
Alex 50  
;
```

- What if we want complete details from dataset-1 and matching details from dataset-2
- Marks of only students from data set1 and fetch their English marks from data set 2
- The other way around - Marks of only students from data set2 and fetch their Maths marks from data set 1

Demo: Merge with condition

```
data twosub_student1;  
merge students1(in=a) students2(in=b);  
by name;  
if a;  
run;
```

```
data twosub_student2;  
merge students1(in=a) students2(in=b);  
by name;  
if b;  
run;
```

```
proc print data= twosub_student1; run;  
proc print data= twosub_student2; run;
```

Merge Condition Statements

Must be in 1st dataset;

`if in1; * Same as: if in1 = 1;`

Must be in 2nd dataset;

`if in2;`

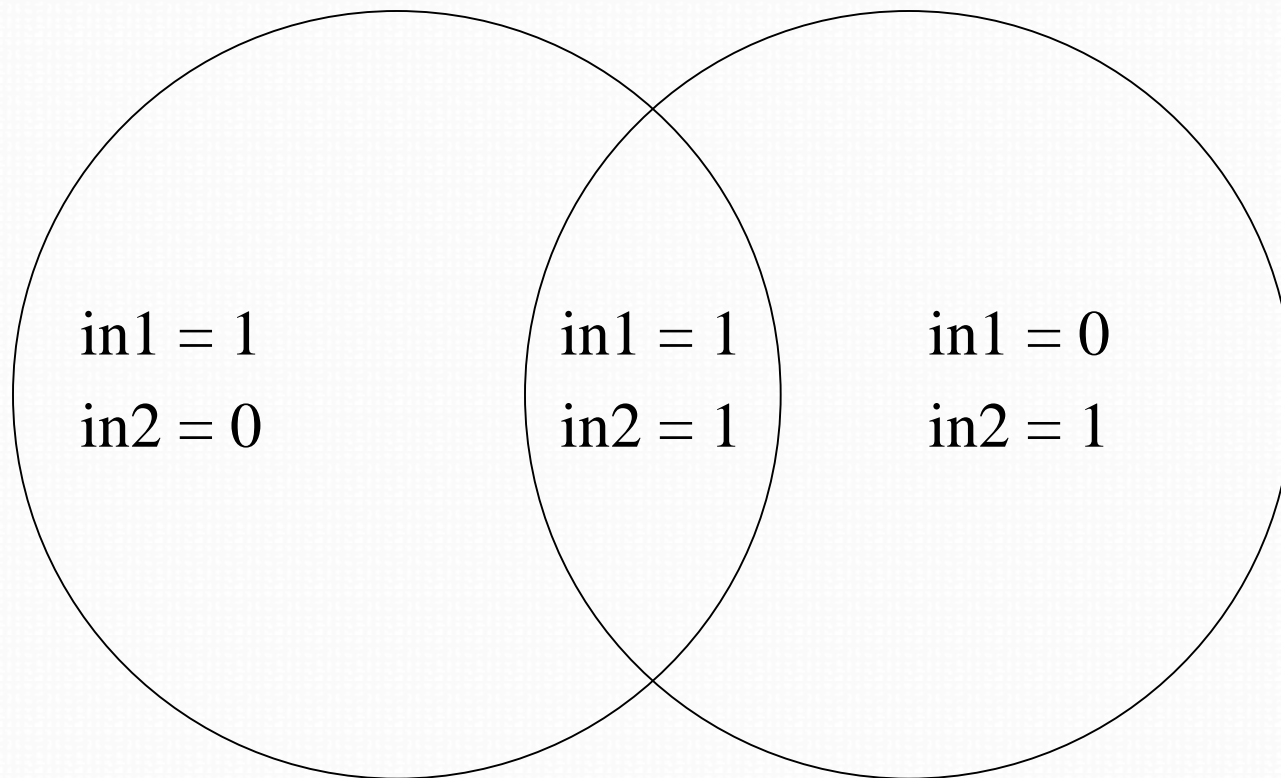
Must be in both datasets;

`if in1 and in2;`

Merge Condition Statements

DATASET 1

DATASET 2



LAB: Merge with Condition

- Merge the data sets orders and spot data
- Create a new data set with all orders data along with matching spot data fields
- Create a new data set with all spot data along with matching orders data fields

```
data twosub_student1;  
merge students1 (in=a)  students2 (in=b) ;  
by name;  
if a;  
run; b
```

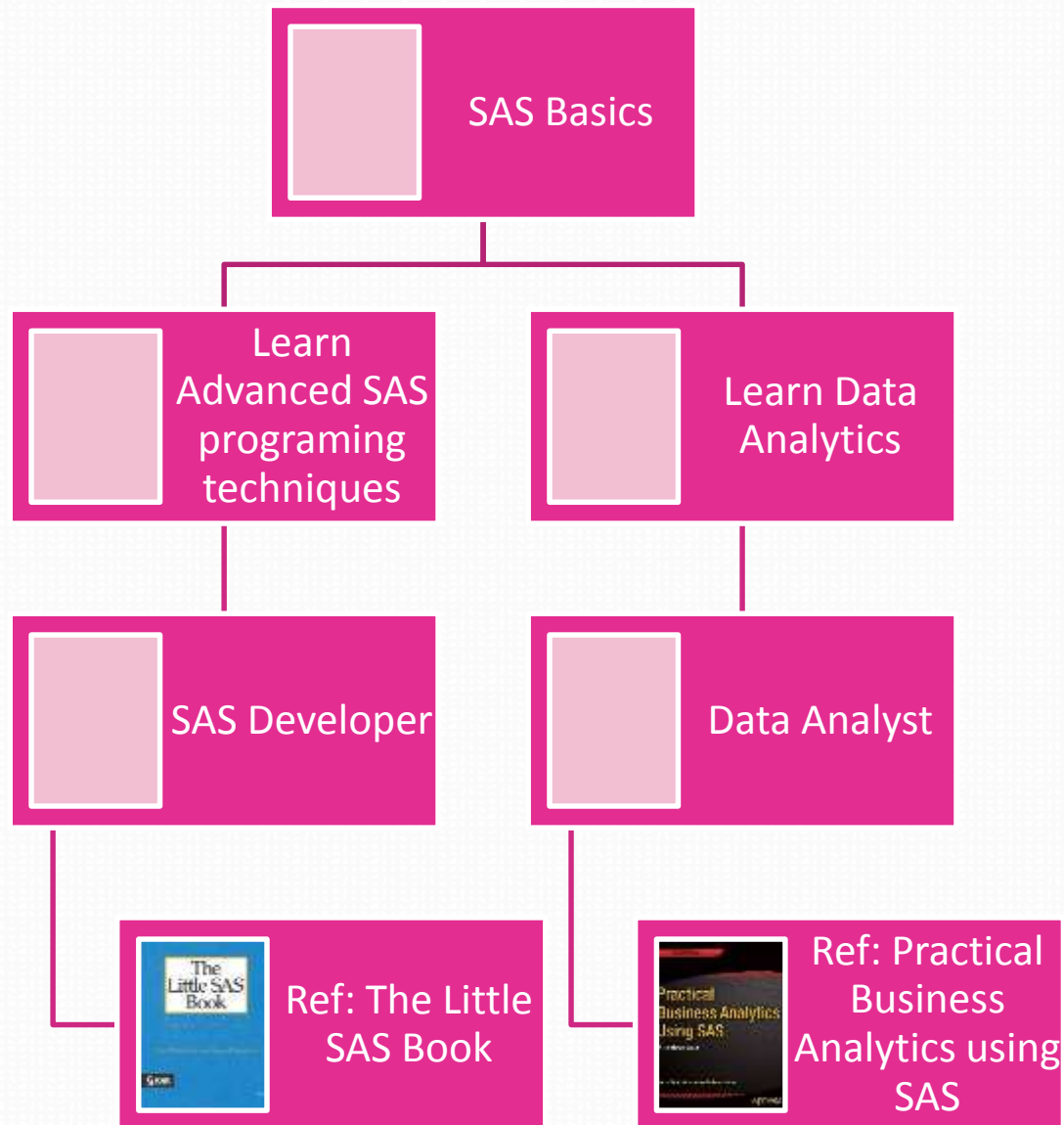
LAB: Merge with Condition

- Sort both bill and complaints data on del_no. Remove duplicates using nodupkey on del_no
- Attach complaint details of the customers to the billing data, all customers might not have complaints
- Attach billing details to complaints data, if available
- Create a data set and print all the customers whose billing and complaints details are available

```
data twosub_student1;  
merge students1 (in=a)  students2 (in=b) ;  
by name;  
if a;  
run;
```

Next Steps

What next??



Thank you