Statistical Analysis System: Class 25

Dated: 26/05/2018

Prog 1 : Appending data from multiple datasets into 1 dataset

```
%macro append;

data a b c;
set sasuser.admit;
run;

proc sql;
select distinct memname into:ds separated by " " from dictionary.tables
where lowcase(libname)="work";
quit;

%put *&ds*;

data all;
set &ds;
run;

%mend append;
%append;
```

Log Display:

```
*A B C*

NOTE: There were 21 observations read from the data set WORK.A.
NOTE: There were 21 observations read from the data set WORK.B.
NOTE: There were 21 observations read from the data set WORK.C.
NOTE: The data set WORK.ALL has 63 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Explained: working with dictionary.tables here, from “work library” we get 3 distinct values for “memname” that are stored in variable “ds”. The resulting dataset “all” has data from all of the values of “memname”.

Prog 2: Appending data from multiple sheets into 1 sheet within an excel file (dynamically)

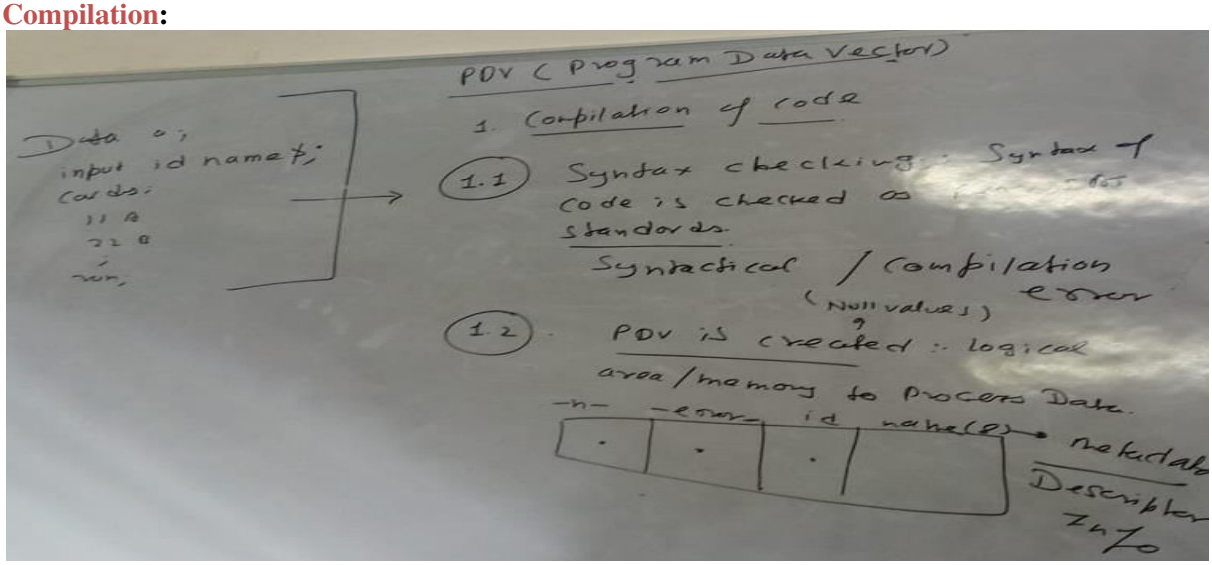
```
%macro kuchnahi;

libname ki"C:\Documents and Settings\sasadm\Desktop\2\cargorev.xls";

proc sql noprint;
select cats(libname,".","'",memname,"'n") into:ds separated by " " from
dictionary.tables where lowcase(libname)="ki";
quit;
```

```
%put *&ds*;

data ki.all;
set &ds;
run;

libname ki clear;

%mend kuchnahi;
%kuchnahi;
```

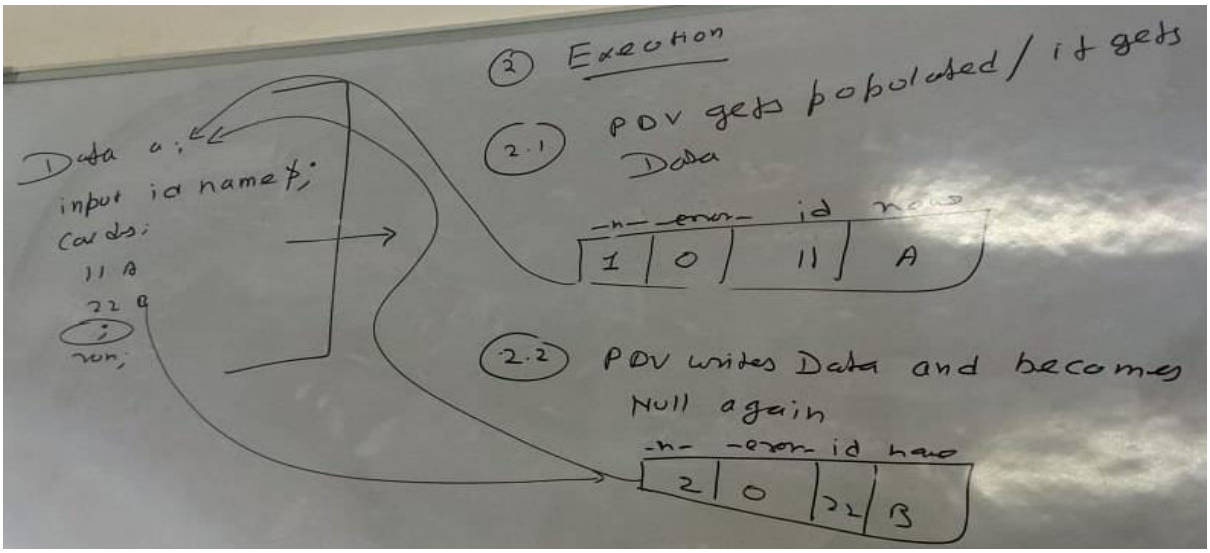
Explained: Just like code 1 above, this program also creates an appended sheet at the end within the same excel file using dictionary.tables. Here, library is “ki”, memname are “datasets created from multiple sheets” stored in “ds” and “all” is the new sheet appended at the end of the orginal excel file, with all the appended data therein.

Program Data Vector (PDV):

Compilation:



Execution:



- The Program Data Vector is a logical area of memory that is created during the data step processing.
- PDV is brain of SAS.
- The program data vector contains two types of variables.
    - Permanent (data set and computed variables)
    - Temporary (automatic and option defined) Automatic (_N_ and _ERROR_)

- Suppose we have a data set

```
data a;
input id name$;
cards;
11  A
12  B
;
run;
```

- **Step 1:-** **Compilation**
- **Step 1.1:-** syntax checking of the code, syntactical errors is compilation errors.
- **Step 1.2:-** PDV gets created. (Logical concepts where SAS builds data).
- When you submit a DATA step for execution, SAS checks the syntax of the SAS statements and compiles them, that is, automatically translates the statements into machine code. In this phase, SAS identifies the type and length of each new variable, and determines whether a variable type conversion is necessary for each subsequent reference to a variable. During the compilation phase, SAS creates the following three items:
- Input buffer is a logical area in memory into which SAS reads each record of raw data when SAS executes an INPUT statement. Note that this buffer is created only when the DATA step reads raw data. (When the DATA step reads a SAS data set, SAS reads the data directly into the program data vector.)
- Program data vector (PDV) is a logical area in memory where SAS builds a data set, one observation at a time. When a program executes, SAS reads data values from the input buffer or creates them by executing SAS language statements. The data values are assigned to the appropriate variables in the program data vector. From here, SAS writes the values to a SAS data set as a single observation.
- Along with data set variables and computed variables, the PDV contains two automatic variables, _N_ and _ERROR_. The _N_ variable counts the number of times the DATA step begins to iterate. The _ERROR_ variable signals the occurrence of an error caused by the data during execution. The value of _ERROR_ is either 0 (indicating no errors exist), or 1 (indicating that one or more errors have occurred). SAS does not write these variables to the output data set.
- Descriptor information is information that SAS creates and maintains about each SAS data set, including data set attributes and variable attributes. For example, it contains the name of the data set, its member type, the date and time that the data set was created, and the number, names, and data types (character or numeric) of the variables. The descriptor information also contains information about extended attributes (if defined on a data set). Extended attribute descriptor information includes the name of the attribute, the name of the variable, and the value of the attribute.

Compilation Phase

- **Input buffer**, an area of memory, is created to hold a record from the external file. It's a logical concept

Input Buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Note: The input buffer is created only when raw data is read, not when a

Program Data Vector

N	ERROR
1	0

- Then the **PDV** is created. The program data vector is the area of memory where SAS software builds a data set, one observation at a time.

Program Data Vector

- **Program Data Vector (PDV)**, a logical framework that the SAS System uses when creating SAS data sets.

Raw Data File Invent

```
>----+----1----+----2----+
Bird Feeder     LG088  3 20
6 Glass Mugs    SB082  6 12
Glass Tray      BQ049 12  6
Padded Hangrs   MN256 15 20
Jewelry Box     AJ498 23  0
Red Apron       AQ072  9 12
Crystal Vase    AQ572 27  0
Picnic Basket   LS930 21  0
Brass Clock     AN910  2 10
```

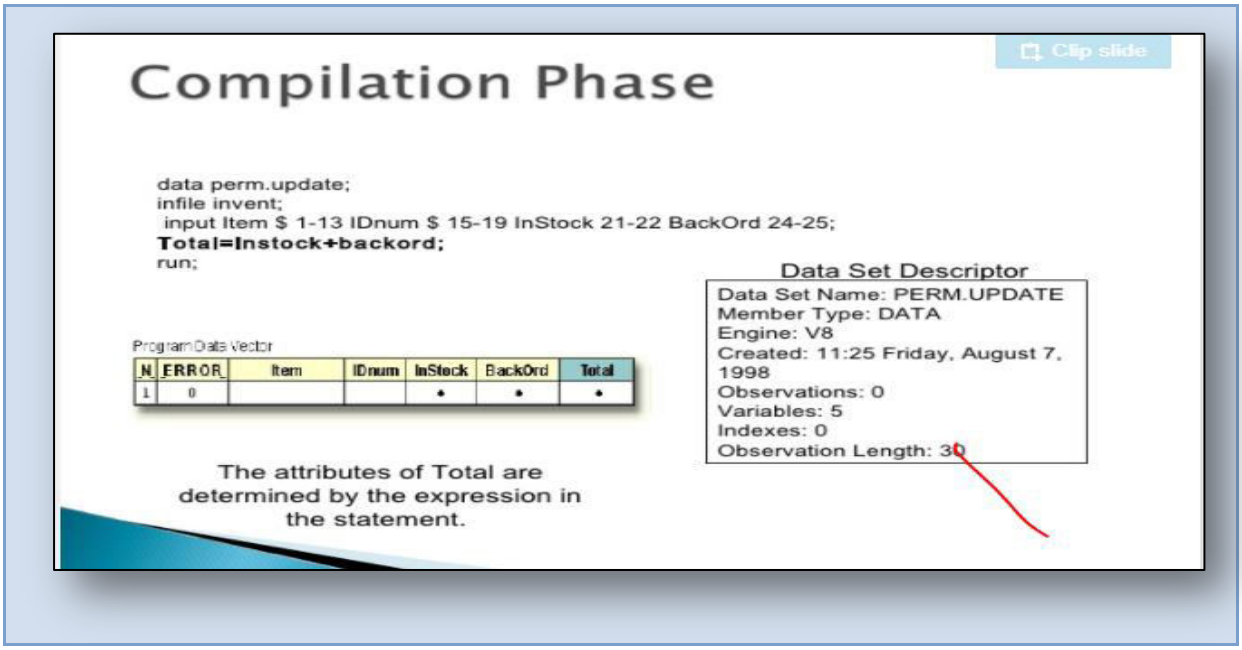
```
data perm.update;
   infile invent;
   input Item $ 1-13 IDnum $ 15-19
         Instock 21-22 BackOrd 24-25;
   Total=instock+backord;
run;
```

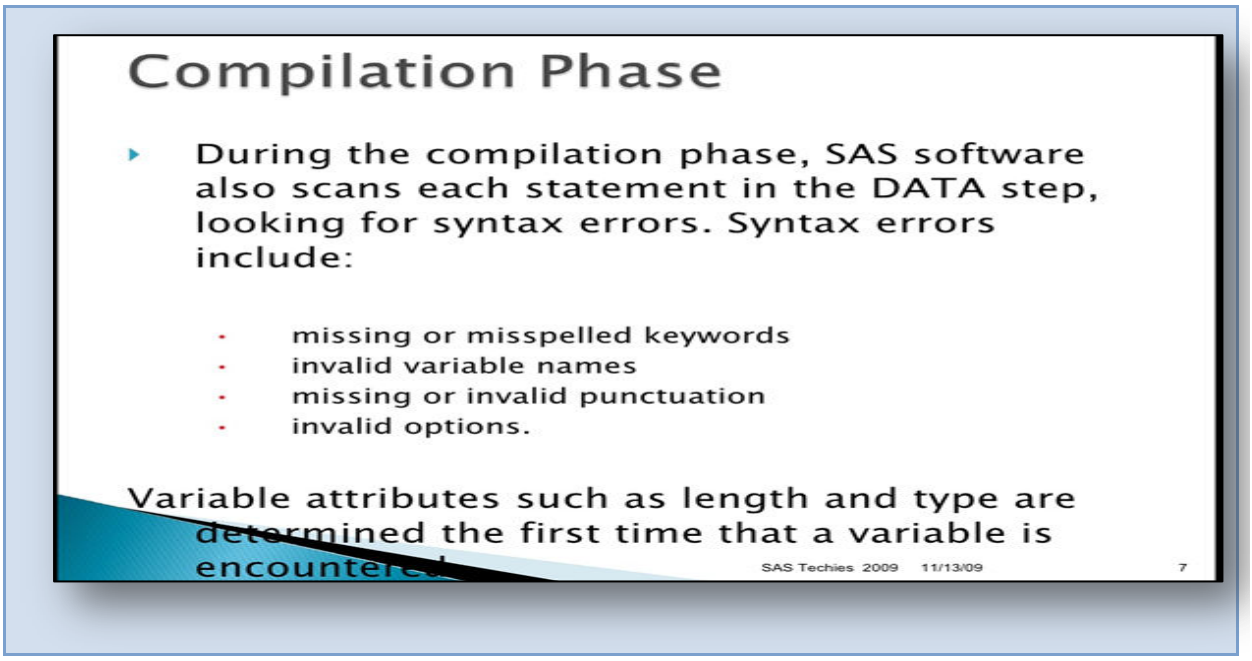
Program Data Vector

N	ERROR	Item	IDnum	InStock	BackOrd	Total
2	0	6 Glass Mugs	SB082	6	12	18

SAS Data Set Perm.Update

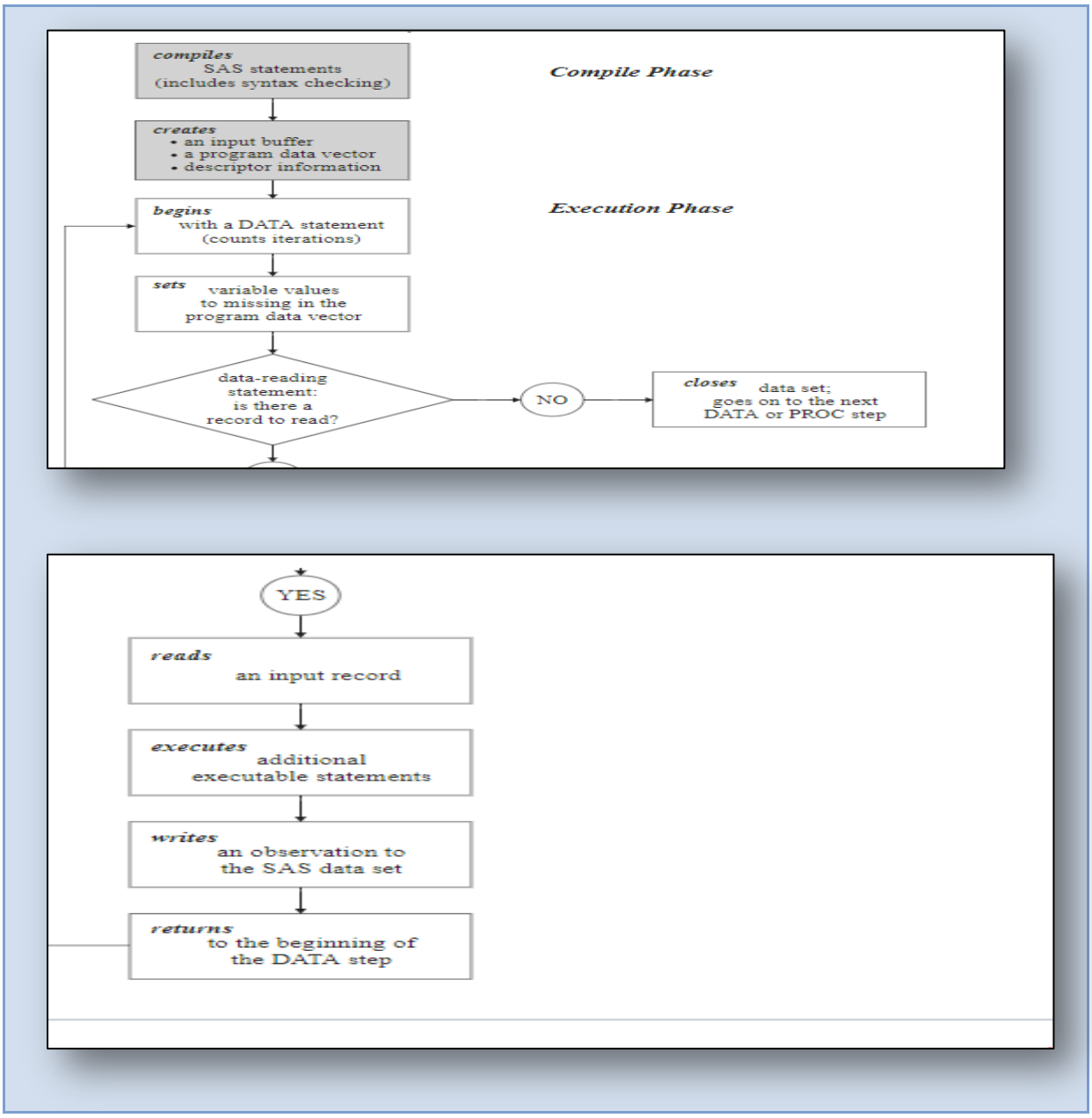
Item	IDnum	InStock	BackOrd	Total
Bird Feeder	LG088	3	20	23
6 Glass Mugs	SB082	6	12	18





- **Step 2:-** **Execution Phase**.
- **Step 2.1:-** PDV gets populated, it writes data to dataset, and it iterate again.
- By default, a simple DATA step iterates once for each observation that is being created. The flow of action in the Execution Phase of a simple DATA step is described as follows:
- The DATA step begins with a DATA statement. Each time the DATA statement executes, a new iteration of the DATA step begins, and the _N_ automatic variable is incremented by 1.
- SAS sets the newly created program variables to missing in the program data vector (PDV).
- SAS reads a data record from a raw data file into the input buffer, or it reads an observation from a SAS data set directly into the program data vector. You can use an INPUT, MERGE, SET, MODIFY, or UPDATE statement to read a record.
- SAS executes any subsequent programming statements for the current record.

- At the end of the statements, an output, return, and reset occur automatically. SAS writes an observation to the SAS data set, the system automatically returns to the top of the DATA step, and the values of variables created by INPUT and assignment statements are reset to missing in the program data vector. Note that variables that you read with a SET, MERGE, MODIFY, or UPDATE statement are not reset to missing here.
- SAS counts another iteration, reads the next record or observation, and executes the subsequent programming statements for the current observation.
- The DATA step terminates when SAS encounters the end-of-file in a SAS data set or a raw data file.
- **Note:** The figure shows the default processing of the DATA step. You can place data-reading statements (such as INPUT or SET), or data-writing statements (such as OUTPUT), in any order in your program.



Below mentioned codes shows the various other ways of using the _N_ .

Code	**Explanation**
data a; set sasuser.admit; put _all_; run;	Put _all_ is used to display the entire data of the DATASTEP. Output: 21 observation & 9 variables
data a; set sasuser.admit; if _n_=5; run;	If _n_=5, copies only the 5th observation as the output. Output: 1 observation & 9 variables
data a; set sasuser.admit; if _n_=5 then delete; run;	If _n_=5, deletes the 5th observation from the output. Output: 20 observation & 9 variables
data a; set sasuser.admit; if _n_ IN (5:9); run;	Gets the observation from 5 to 9 including both extremes Output: 5 observation & 9 variables
data a; set sasuser.admit; if mod(_n_,2)=0; run;	Gets all even observations for which (_n_ mod 2 =0) Output: 10 observation & 9 variables
data a; set sasuser.admit; if mod(_n_,7)=0; run;	Gets all observations for which (_n_ mod 7 =0) Output: 3 observation & 9 variables

data a;
set sasuser.admit;
x=_N_;
run;
 	
proc sort data=a out=b(drop=x);
by descending x;
run;
	Inverting data or contents in a table- upside down.
	
	Sorting data in descending order.

data a; set sasuser.admit; where _n_ gt 5; run;	Does not execute because where is executed pre –PDV, causing failure of the condition for pointer _n_.

-