TAP TO GO BACK TO KIOSK MENU

# SAS® GLOBAL FORUM 2020

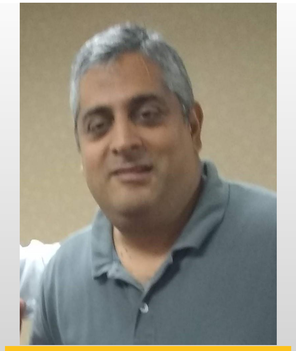
MARCH 29 – APRIL 1
WASHINGTON, DC

USERS PROGRAM

# If you need these OBS and these VARS, then drop IF, and keep WHERE

TAP TO GO BACK TO KIOSK MENU

Jay Iyengar
Data Systems Consultants LLC



Jay Iyengar

## Abstract

Reading data effectively in the DATA step requires knowing the implications of various methods, and DATA step mechanics; The Observation Loop, and the PDV. The impact is especially pronounced when working with large data sets. Individual techniques for subsetting data have varying levels of efficiency and implications for input/output time. Use of the WHERE statement/option to subset observations consumes less resources than the subsetting IF statement. Also, use of DROP and KEEP to select variables to include/exclude can be efficient depending on how they're used.



Please use the headings above to navigate through the different sections of the poster

## Introduction

The DATA step is the primary construct and tool for data manipulation in the BASE SAS® package. As such, it provides a litany of capabilities for processing data. Subsetting, conditional processing, merging, by-group processing, appending, summarizing, and deriving variables are all processes that the DATA step can execute. It's also a versatile tool which provides multiple methods to perform the same tasks and accomplish the same result. The basic structure of the DATA step reads an input data set, and writes an output data set. This e-poster primarily focuses on methods to select observations and variables, and the efficiency of different techniques using the basic form of the DATA step.

## The DATA Step

| OBS | ID | DOB | SEX | AGE |
|---|---|---|---|---|
| **1** | **52805** | **103179** | **M** | **34** |
| 2 | 52806 | 081776 | M | 28 |
| 3 | 52807 | 050684 | F | 30 |
| 4 | 52808 | 012969 | M | 35 |
| 5 | 52809 | 031273 | F | 41 |
| 6 | 52810 | 122575 | M | 38 |
| 7 | 52811 | 070476 | F | 38 |

```
Data TWO;
  Set ONE;
  If  SEX = 'M';

   Y=YEAR(DOB);
  M=MONTH(DOB);

  Keep ID SEX M Y;
  Format SEX 2.;

Run;
```

The DATA step has an automatic loop, which reads and writes data sequentially, or in order. Each loop is referred to as a 'DATA step Iteration'. It also processes data one observation at a time. In the first loop of the data step, the SET statement executes, and SAS reads the first record from the input data set. Consequently, each programming statement in the data step is executed on that observation. At the bottom of the loop, after its finished executing all statements, SAS writes the observation to the output data set. There's a default output at the bottom of the loop, when SAS encounters the RUN statement. Next, a new loop begins and the second observation is read from the input data set. This process continues until all observations have been read from the input data set(s). After all observations have been read, SAS encounters an end-of-file marker, and the DATA step stops executing.

# If you need these OBS and these VARS, then drop IF, and Keep WHERE

Jay Iyengar
Data Systems Consultants LLC



Please use the headings above to navigate through the different sections of the poster

## The PDV

The PDV stands for Program Data Vector. The PDV is an internal record in memory where data is held while it's being processed through the DATA step. When the DATA step is compiled, the PDV is created containing all variables on the input SAS data set. Any new variables created during the step are initialized to missing. When the DATA step executes, SAS reads the first observation from the input data set, and sends it to the PDV. The record is held in the PDV during the DATA step loop. when SAS reaches the end of the DATA step, the observation held in the PDV is released and written to the output data set

| SAS DATA SET ONE | | | | |
|---|---|---|---|---|
| OBS | ID | DOB | SEX | AGE |
| 1 | 52805 | 103179 | M | 36 |
| 2 | 52806 | 081776 | M | 28 |
| 3 | 52807 | 050684 | F | 30 |
| 4 | 52808 | 012969 | M | 35 |

| PDV – PROGRAM DATA VECTOR | | | | |
|---|---|---|---|---|
| OBS | ID | DOB | SEX | AGE |
| 1 | 52805 | 103179 | M | 36 |

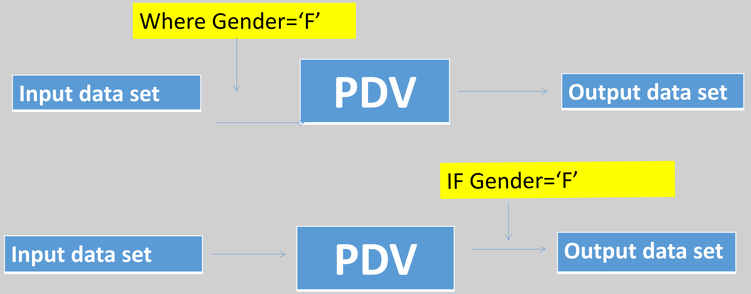
## IF VS. WHERE – Subsetting observations

```
**IF STATEMENT**

DATA NDF1;
  SET NDF;
  IF GENDER='F' AND STATE='AZ';
RUN;
```

```
**WHERE STATEMENT**

DATA NDF2;
  SET NDF;
  WHERE GENDER='F' AND STATE='AZ';
RUN;
```

In the DATA step there are multiple methods available to subset observations. There's the IF statement, also known as the subsetting IF, and the WHERE statement. Both statements subset data based on a condition. If the condition evaluates to true, then processing continues, and the observation is read from or written to a SAS data set. If the condition is false, the record is deleted or excluded, and the pointer moves to the next observation. Using the IF statement, you control which observations are written to the output data set. With the WHERE statement you control which observations are read from the input data set. WHERE can be also used as a data set option, either input or output.

## IF VS. WHERE – DATA step processing




The key difference between IF and WHERE is in DATA step mechanics, and the impact of each technique for efficient processing of data. This involves understanding the PDV. The WHERE statement or input data set option applies its condition before a record is read and loaded to the PDV. Only those records meeting the WHERE condition are read from the input data set. The IF statement applies its condition after a record is read and loaded to the PDV, but before its written to the output data set. All records in the input data set are read using the subsetting IF. Only those records meeting the IF condition are written to the output data set.

# If you need these OBS and these VARS, then drop IF, and keep WHERE

Jay Iyengar
Data Systems Consultants LLC

## IF and WHERE – Limitations on their use



Please use the headings above to navigate through the different sections of the poster

There are different places and situations within your SAS code where WHERE and IF can be used. WHERE can be used both in a DATA step, and SAS Procs, while IF can only be used in a DATA step. Both IF and WHERE can be used to subset data based on a SAS data set. Some programming tasks involve reading data in a text, ascii, or flat file format, and converting it to a SAS data set. To subset an external file, use Subsetting IF, or the WHERE output data set option. WHERE or WHERE= input data set option cannot subset an external file.

| | IF | WHERE |
|---|---|---|
| Data Steps | X | X |
| SAS data sets | X | X |
| Proc Steps | | X |
| External Files with INFILE/INPUT | X | |
| Automatic/Temporary Variables | X | |
| Computed/Derived Variables | X | |

## IF VS. WHERE – Comparison example

```
56                  DATA NDF_WI1;
57                      SET NEWFILE.NDF;
58                      IF STATE = 'WI';
59                  RUN;

NOTE: There were 1048575 observations read from the data set NEWFILE.NDF.
NOTE: The data set WORK.NDF_WI1 has 22518 observations and 36 variables.

61                  DATA NDF_WI2;
62                      SET NEWFILE.NDF;
63                      WHERE STATE = 'WI';
64                   RUN;

NOTE: There were 22518 observations read from the data set NEWFILE.NDF  WHERE STATE='WI';
NOTE: The data set WORK.NDF_WI2 has 22518 observations and 36 variables.
```

The example shows using IF to produce a subset of Maryland physicians, the entire data sets was read (1048575 records). The subset was applied to the output data set . Using WHERE to produce the same subset, only the subset is read from the input data set (22518  records). All else being equal, WHERE is more efficient because smaller amounts of data are processed. Both the WHERE statement and WHERE input data set control reading observations , and have the same efficiency impact. The subsetting IF has the same effect as the WHERE output data set option, and the same impact on efficiency.

# If you need these OBS and these VARS, then drop IF, and keep WHERE

Jay Iyengar
Data Systems Consultants LLC



Please use the headings above to navigate through the different sections of the poster

## Optimal placement of IF statement

```
DATA MedU16;
   SET MedUtiliz16;
   IF CITY='Bethesda';

   IF PROVIDER_CITY='Baltimore' THEN PROVIDER_STATE='MD';
   IF PROVIDER_CITY='Arlington' THEN PROVIDER_STATE='VA';

   IF BILLTYPE = 13 THEN BILLTYPEDSC = 'Hospital Outpatient';
   ELSE IF BILLTYPE = 11 THEN BILLTYPEDSC = 'Hospital Inpatient';
   ELSE IF BILLTYPE = 33 THEN BILLTYPEDSC = 'Home Health Agency';
RUN;
```

If you're using the Subsetting IF in a DATA step, It's best to position it just below the SET statement, and before the rest of the code in the step. This way IF will be executed before any remaining code in the step. The remaining code will only be executed for observations which meet the subsetting criteria. Time won't be wasted processing code for observations which don't meet the subsetting condition. Positioning IF at the top of the step saves CPU or processing time.

## Selecting variables with KEEP and DROP

To subset variables in a DATA step, you use the KEEP or DROP option/statement. With KEEP and DROP you specify the variables to be included or excluded for processing. In a DATA step, KEEP and DROP can be specified either as a statement or a data set option in parentheses. KEEP/DROP can be coded either as an input or output data set option. The KEEP/DROP input data set option controls which variables are read or not read from an input SAS data set. The KEEP/DROP output data set option controls which variables are written or not written to an output data set.

### Keep/Drop Statements

```
DATA TWO;
  SET ONE;
  KEEP DOB AGE;
RUN;
```

```
DATA TWO;
  SET ONE;
  DROP SEX AGE;
 RUN;
```

### Keep/Drop Input/Output data set option(s)

```
DATA TWO;
 SET ONE (KEEP=ID GENDER);
RUN;
```

```
DATA TWO (DROP=SEX AGE);
 SET ONE;
RUN;
```

# If you need these OBS and these VARS, then drop IF, and keep WHERE

Jay Iyengar
Data Systems Consultants LLC



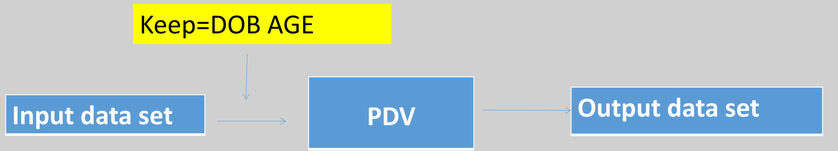
As we've seen, KEEP/DROP can be used in a DATA step, either as an input data set option, output data set option, or SAS statement. In SAS Procs, KEEP/DROP can be used as a data set option. The KEEP/DROP statement can only be used within a DATA step. All forms of KEEP and DROP can list variables to select or exclude from a SAS data set. Only the KEEP/DROP statement or output data set option can be used in a DATA step that reads an external file, such as a text or flat file

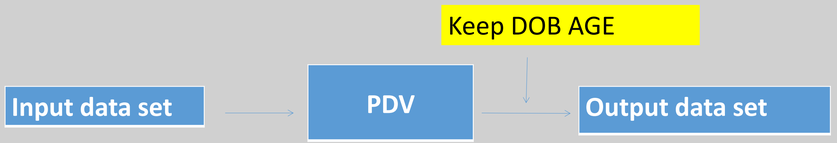
## KEEP\DROP step and file restrictions

| | DROP Option/Statement | KEEP Option/Statement |
|---|---|---|
| DATA Step | X | X |
| PROC Step | X (option only) | X (option only) |
| SAS Data Sets | X | X |
| External Files | X (statement only) | X (statement only) |

Please use the headings above to navigate through the different sections of the poster

## KEEP\DROP and efficient DATA step processing




In deciding which form of KEEP/DROP to use, its key to understand where they're executed during DATA step processing. The KEEP/DROP input data set option is applied before variables are loaded to the PDV. Using this option, only the variables that are listed (KEEP) or not listed (DROP) are read from the input data set. The KEEP/DROP statement or output data set option is applied after variables have been loaded into the PDV, before they're written to the output data set. All variables are read from the input data set, using either of these methods.

# If you need these OBS and these VARS, then drop IF, and keep WHERE

Jay Iyengar
Data Systems Consultants LLC



Please use the headings above to navigate through the different sections of the poster

## Output data set option - data set specific variable lists

```
Data HOSPOUT(KEEP=PATID PROVID AGE)  HOSPIN (KEEP=PATID PROVID LOS)
        HHA (KEEP=PATID PROVID SEX);
    Set MEDUTIL16;
                IF BILLTYPE=13 THEN OUTPUT HOSPOUT;
                ELSE IF BILLTYPE=11 THEN OUTPUT HOSPIN;
                ELSE IF BILLTYPE=33 THEN OUTPUT HHA;
  Run;
```

The KEEP/DROP statement and output data set option both control which variables are written to the output data set. Both of these methods have the same processing efficiency. The output data set option is used effecitively when outputting multiple data sets. Using the data set option instead, variable lists custom to each data set can be created.

## Input data set option and DATA step MERGE

By default, If you're merging two files with same-named variables, the variable in the second file will overwrite the variable in the first. The variable in the output data set will then have the values of the variable in the second file. Using KEEP or DROP as an input data set option can be handy in DATA step merges in preventing variables from overwriting each other.

```
DATA THREE;
  MERGE ONE(IN=A KEEP=ID AGE)  TWO(IN=B DROP=AGE);
    BY ID;
  IF A AND B;
RUN;
```

## Conclusion

In the DATA step, to decrease CPU (processing time) and i/o (input\output), I recommend using WHERE input data set option to subset from a SAS data set. Use the subsetting IF when subsetting external files or the results of a DATA step merge. Using these methods is necessary when processing large data sets. I recommend using KEEP/DROP input data set option to select only necessary variables from a SAS data set, and in a DATA step merge to prevent overwriting common variables. Use KEEP\DROP Output data set option to create variable specific lists for multiple output data sets. Its important to do comparative testing with these methods because results are environment and data dependent.


MARCH 29 – APRIL 1 | WASHINGTON, DC | #SASGF