Paper 213-2007

WHERE vs. IF Statements: Knowing the Difference in How and When to Apply

Sunil Gupta, Quintiles, Thousand Oaks, CA

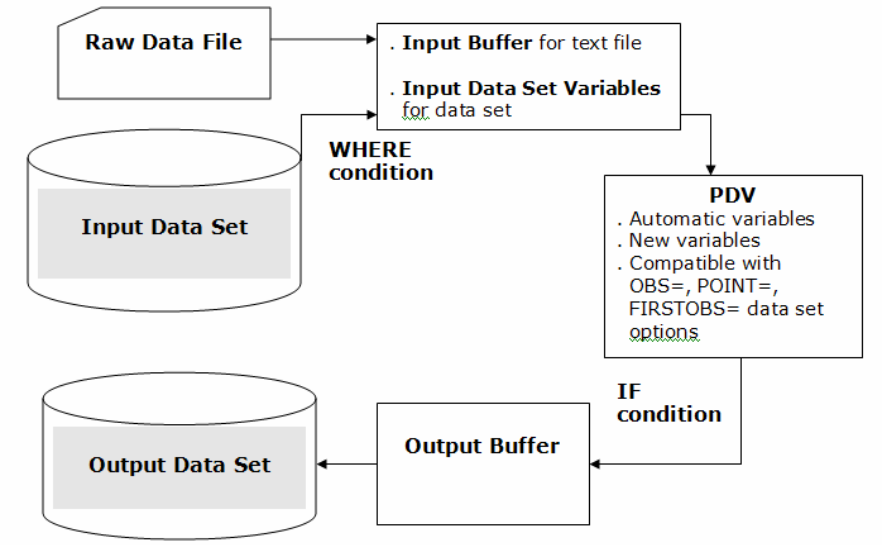
ABSTRACT

When programming in SAS, there is almost always more than one way to accomplish a task. Beginning programmers may think that there is no difference between using the WHERE statement and the IF statement to subset your data set. Knowledgeable programmers know that depending on the situation, sometimes one statement is more appropriate than the other. For example, if your subset condition includes automatic variables or new variables created within the DATA step, then you must use the IF statement instead of the WHERE statement. This paper shows you how and when to apply the WHERE and IF statements to get correct and reliable results. It also reviews the similarities as well as the differences between these two SAS programming approaches. Note that this paper does not focus on comparing the performance differences between the WHERE and IF statements.

INTRODUCTION

As shown in the figure below, WHERE conditions are applied *before* the data enters the input buffer while IF conditions are applied *after* the data enters the program data vector. This is the reason why the WHERE condition is faster because not all observations have to be read and because it can only be applied on variables that exist in the input data set. Each WHERE or IF expression must be consistent and contain all numeric or all character variables. WHERE conditions were introduced in SAS version 6.12.

Note that multiple WHERE conditions do not become cumulative subset conditions unless the 'ALSO' keyword is added. By default, the most recent WHERE condition replaces the previous WHERE condition. The IF conditions, however, are cumulative. In addition, it is not the scope of this paper to address efficiency issues between the two approaches to subset the data set. Note that there is no input buffer when reading a SAS data set.



The key differences, other than performance issues, between WHERE and IF Conditions can be summarized in the table below. The examples following this table show some of these differences.

Summary of Key Differences between WHERE and IF Conditions to Subset Data Sets

Subset Data set	WHERE	IF
(No Difference between WHERE and IF Conditions other than better performance when using WHERE)		
Using variables in data set ❶	**X**	**X**
Using SET, MERGE or UPDATE statement if within the DATA step*	**X**	**X**
(Must use IF Condition)		
Accessing raw data file using INPUT statement ❷		**X**
Using automatic variables such as _N_, FIRST.BY, LAST.BY ❸		**X**
Using newly created variables in data set or existing variables AFTER their value has been changed in the data step ❹		**X**
In combination with data set options such as FIRSTOBS =, OBS =**, and POINT = ,		**X**
To conditionally execute statement		**X**
(Must use WHERE Condition)		
Using special operators*** such as LIKE or CONTAINS ❺	**X**	
Directly using any SAS Procedure ❻	**X**	
More efficiently****	**X**	
Using index, if available	**X**	
When subsetting as a data set option	**X**	
When subsetting using the SQL procedure	**X**	
(Be careful which you use! In general, use IF Condition to subset after merging.)		
When merging data sets***** ❼	**SUBSET *BEFORE* MERGING**	**SUBSET *AFTER* MERGING**

* WHERE condition requires one of these statements if used within the DATA step. In addition, the variable specified in the WHERE condition must exist in all data sets because SAS subsets each data set before merging them. If using the WHERE = data set option, then variable specified in the WHERE condition does NOT have to exist in all data sets, just in the specified data set.

** FIRSTOBS= and OBS = data set option is compatible with the WHERE statement in SAS version 8.1 and higher. When FIRSTOBS= is used, SAS starts reading the data set from that observation number. When OBS = is used with the IF statement, SAS first subsets the data set based on the number of observations in the OBS = option and then applies the IF subset condition. When OBS = is used with the WHERE statement, SAS first applies the WHERE subset condition and then restricts the output data set to contain the maximum of observations as specified in the OBS = option.

*** The Colon Modifier (:) works with the IF statement to compare the starting of the longer text of the variable to the shorter text by truncating the longer text to the length of the shorter text.

**** WHERE condition may be more efficient because SAS is not required to read all observations from the input data set.

***** Results may be different depending on the data sets being merged. In general, use the IF condition to subset the data set after merging the data sets.

Source: Sharpening Your SAS Skills, CRC Press (www.crcpress.com, www.sas.com), April 2005

The seven examples in this paper show how to correctly apply the WHERE and IF statements based on the variables and the subset condition. In addition, because key words are very important, error messages are displayed when specifying the incorrect subset statement.

EXAMPLES

❶ Using variables in data set, using SET, MERGE, or UPDATE statement if within the DATA step – No difference other than better performance with WHERE statement
❷ Accessing raw data file using INPUT statement – Must use IF statement
❸ Using automatic variables such as _N_, FIRST.BY, LAST.BY – Must use IF statement
❹ Using newly created variables in data set – Must use IF statement
❺ Using special operators such as LIKE or CONTAINS – Must use WHERE statement
❻ Directly using any SAS Procedure or as a data set option – Must use WHERE condition
❼ When merging data sets – Be careful when subsetting; in general, use IF statement to subset after merging

SAMPLE DATA SET

Below is the sample data set that will be used in the examples. The data set contains test scores of three classes from three students for a total of nine records.

```
data exam;
  input name $ class $ score ;
  cards;
Tim math 9
Tim history 8
Tim science 7
Sally math 10
Sally science 7
Sally history 10
John math 8
John history 8
John science 9
;
run;
```

EXAMPLE ❶

Using variables in data set, using SET, MERGE, or UPDATE statement if within the DATA step – Although WHERE or IF statement can be used, for better performance, use the WHERE statement.

In this example, the DATA step contains a WHERE statement based on a variable in the EXAM data set. The SET statement is used to access the EXAM data set. The records in the new data set, STUDENT1, will contain only those records that meet the WHERE condition.

```
data student1;
  set exam;

  * Can use WHERE condition because NAME variable is a data set variable;
  * WHERE condition requires all data set variables;
  where name = 'Tim' or name = 'Sally'; ❶

run;
```

In this case, you can apply the IF statement instead of the WHERE statement to get the same results. Note that in the basic form, the syntax of WHERE and IF statements are the same except for the keyword.

```
  if name = 'Tim' or name = 'Sally'; ❶
```

As you can see below, all six records meet the condition, which required Tim's or Sally's scores. Since either the WHERE or IF statement could have been used, using the WHERE statement would be more efficient.

STUDENT1 data set

Obs	name	class	score
1	Tim	math	9
2	Tim	history	8
3	Tim	science	7
4	Sally	math	10
5	Sally	science	7
6	Sally	history	10

What would happen if you incorrectly specified the IF statement by mixing a numeric variable in a character expression as shown below?

```
data student1;
  set exam;

  if score = '9';
run;
```

With IF conditions, SAS will automatically converts variable types.

```
70   data student1;
71     set exam;
72
73     if score = '9';
74
75   run;

NOTE: Character values have been converted to numeric values at the places given by:
(Line):(Column).
      73:14
NOTE: There were 9 observations read from the data set WORK.EXAM.
NOTE: The data set WORK.STUDENT1 has 2 observations and 3 variables.
```

With WHERE conditions, however, SAS does not automatically convert variable types. It generates an ERROR message because operator requires compatible variables.

```
24   data student1;
25     set exam;
26
27      where score = '9';
ERROR: Where clause operator requires compatible variables.
28   run;

NOTE: The SAS System stopped processing this step because of errors.

WARNING: THE DATA SET WORK.STUDENT1 MAY BE INCOMPLETE.  WHEN THIS STEP WAS STOPPED
THERE WERE 0 OBSERVATIONS AND 3 VARIABLES.
```

EXAMPLE ❷

Accessing raw data file using INPUT statement – Must use IF statement

To save processing time, you can subset the data as you are reading the records while creating the EXAM data set. Since the variables in the INPUT statement exist only in the program data vector, you must specify the IF statement.

```
data exam;
  input name $ class $ score ;
  if name = 'Tim' or name = 'Sally'; ❷
  cards;
Tim math 9
Tim history 8
Tim science 7
Sally math 10
Sally science 7
Sally history 10
John math 8
John history 8
John science 9
;
run;
```

The result is the same data set as in example 1 because the subset condition is the same.

EXAM data set

Obs	name	class	score
1	Tim	math	9
2	Tim	history	8
3	Tim	science	7
4	Sally	math	10
5	Sally	science	7
6	Sally	history	10

If you replace the IF statement with a WHERE statement, then you will get the following error message. This is because the WHERE statement requires variables from a data set and can not be used when specifying an INPUT statement.

```
42   data exam;
43     input name $ class $ score ;
44     where name = 'Tim' or name = 'Sally';
ERROR: No input data sets available for WHERE statement.
45     * if name = 'Tim' or name = 'Sally';
46     cards;

NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.EXAM may be incomplete.  When this step was stopped there
were 0
         observations and 3 variables.
WARNING: Data set WORK.EXAM was not replaced because this step was stopped.
NOTE: DATA statement used:
      real time           0.10 seconds
      cpu time            0.01 seconds

56   ;
57   run;
```

EXAMPLE ❸

Using automatic variables such as _N_, FIRST.BY, LAST.BY – Must use IF statement

When specifying a condition based on automatic or temporary variables within a DATA step, you must use the IF statement. In this example, the EXAM data set is sorted by the NAME variable. The DATA step uses the IF statement to keep the FIRST.NAME record, because these temporary variables exist only in the program data vector. Variables in the WHERE statement must exist in the data set.

Note that if the dot is missing between the FIRST and NAME, then SAS creates the FIRST variable as a numeric variable, converts the NAME variable from a character to a numeric variable and then expects an operator between FIRST and NAME.

```
data exam;
  input name $ class $ score ;
  cards;
Tim math 9
Tim history 8
Tim science 7
Sally math 10
Sally science 7
Sally history 10
John math 8
John history 8
John science 9
;
run;

proc sort data = exam out=student2;
 by name;
run;

data student2;
  set student2;
  by name;

  * Use IF condition because NAME is the BY variable;
  if first.name; ❸

run;
```

As you can see below, there is only one record for each student. This meets the subset condition to keep only the first record for each unique value of the student's name. Since the EXAM data set stored the math test scores as the first record for each student, this is the only subject in the STUDENT2 data set.

STUDENT2 data set

Obs	name	class	score
1	John	math	8
2	Sally	math	10
3	Tim	math	9

If you replace the IF statement with a WHERE statement, then you will get the following error message. This is because the WHERE statement does not recognize the temporary variable FIRST.NAME.

```
106  data student2;
107    set student2;
108    by name;
109
110    * Use IF condition because NAME is the BY variable;
NOTE: SCL source line.
111    where first.name;
```

```
              ----------
              180
ERROR: Syntax error while parsing WHERE clause.
ERROR 180-322: Statement is not valid or it is used out of proper order.

112    *if first.name;
113
114  run;
```

EXAMPLE ❹

Using newly created variables in data set – Must use IF statement

When specifying conditions based on variables created within the same DATA step, you must use the IF statement because that variable exists only in the program data vector. In this example, the CLASSNUM variable is created from the CLASS variable. The records in the new data set, STUDENT3, will contain only those records that meet the IF condition.

```
data student3;
  set exam;

  * Create CLASSNUM variable;
  if class = 'math' then classnum = 1;
  else if class = 'science' then classnum = 2;
  else if class = 'history' then classnum = 3;

  * Use IF condition because CLASSNUM variable was created within the DATA step;
  if classnum = 2; ❹
run;
```

As you can see below, all three records meet the condition which requires the CLASSNUM variable to equal 2 or any science score.

STUDENT3 data set

Obs	name	class	score	classnum
1	Tim	science	7	2
2	Sally	science	7	2
3	John	science	9	2

If you replace the IF statement with a WHERE statement, then you will get the following error message. This is because the WHERE statement requires variables to exist in the EXAM data set.

```
156  data student3;
157    set exam;
158
159    * Create CLASSNUM variable;
160    if class = 'math' then classnum = 1;
161    else if class = 'science' then classnum = 2;
162    else if class = 'history' then classnum = 3;
163
164    * Use IF condition because CLASSNUM variable was created within the DATA step;

165    where classnum = 2;
ERROR: Variable classnum is not on file WORK.EXAM.
166    *if classnum = 2;
167  run;
```

EXAMPLE ❺

Using special operators such as LIKE or CONTAINS – Must use WHERE statement

If you could not remember the exact spelling of a student's name, then you can take advantage of special operators such as LIKE or CONTAINS in the WHERE condition. The condition below is similar to the condition in example 1 except that it does not specify the full first name of the students to locate. Notice that this syntax is unique to the WHERE statement. In addition, when working with character variables, consider case sensitivity by using the upcase() function and be aware of embedded blanks.

```
data student4;
  set exam;

  * Can use WHERE condition because NAME variable is a data set variable;
  * WHERE condition requires all data set variables;
  where name =: 'T' or name contains 'ally'; ❺

run;
```

As seen in example 1, all six records meet the condition, which require student names starting with the letter 'T' or that contain the letters 'ally'.

STUDENT4 data set

Obs	name	class	score
1	Tim	math	9
2	Tim	history	8
3	Tim	science	7
4	Sally	math	10
5	Sally	science	7
6	Sally	history	10

Below is a list of special operators available in the WHERE statement:

OPERATOR	DESCRIPTION
BETWEEN … AND	Includes values defined in the range of numeric variables
COLON MODIFIER (:)*	Compares shorter text with longer text by truncating the longer text to the length of the shorter text; default is to pad shorter text with trailing blanks in order to make the comparison
CONTAINS or ?	Used to search for a specific text in character variables
IS NULL or IS MISSING	Includes all missing values including special missing values. Note that the variable can be character or numeric
LIKE 'PATTERN'	In character variables, used to search for similar match. Use with the underscore (_) or the percent sign (%) operator
PERCENT SIGN (%)	Any number of characters are possible, similar to a wildcard character
SOUNDS-LIKE (SOUNDEX) "=*"	Includes all similar character values that sound alike. This does not require the exact spelling of the character value
UNDERSCORE (_)	Each underscore represents any single character

* Colon Modifier can also be used with the IF statement.

If you replace the WHERE statement with an IF statement, then you will get the following error message. This is because SAS does not recognize the special operator 'contains' in the IF statement. Note that is possible to use the colon modifier (:) with the IF statement. No error message is identified under the name variable or the colon modifier.

```
196  data student4;
197    set exam;
198
199    * Can use WHERE condition because NAME variable is a data set variable;
200    * WHERE condition requires all data set variables;
NOTE: SCL source line.
201     if name =: 'T' or name contains 'ally';
                              -------- ------
                              388      200
ERROR 388-185: Expecting an arithmetic operator.

ERROR 200-322: The symbol is not recognized and will be ignored.

202    * where name =: 'T' or name contains 'ally';
203
204  run;
```

EXAMPLE ❻

Directly using any SAS Procedure or as a data set option – Must use WHERE condition

A big advantage of using the WHERE condition is the ability to apply the subset condition directly in SAS Procedures. This approach not only avoids a DATA step to first subset the data set, but also prevents the need to create a separate data set.

```
proc print data = exam;
 where name = 'Tim' or name = 'Sally'; ❻
run;
```

The result is the same as in example 1 because the subset condition is the same.

RESULT

Obs	name	class	score
1	Tim	math	9
2	Tim	history	8
3	Tim	science	7
4	Sally	math	10
5	Sally	science	7
6	Sally	history	10

If you replace the WHERE statement with an IF statement, then you will get the following error message. The IF statement is not valid outside the DATA step. When possible, specify WHERE statements in SAS Procedures to avoid unnecessary DATA step processing.

```
209  proc print data = exam;
NOTE: SCL source line.
210    if name = 'Tim' or name = 'Sally';
       --
       180
ERROR 180-322: Statement is not valid or it is used out of proper order.
211    *where name = 'Tim' or name = 'Sally';
212  run;
```

Note that multiple WHERE conditions are not cumulative unless the 'ALSO' keyword is used. By default, the most recent WHERE condition replaces the previous WHERE condition.

```
proc print data = exam;
 where name = 'Tim';
 where also class = 'math';
run;
```

As an alternative, both a WHERE data set option and condition may be applied for cumulative effect.

```
proc print data = exam (where =(name = 'Tim'));
 where class = 'math';
run;
```

When applying multiple IF conditions within a DATA step, however, the conditions become cumulative.

```
data student1;
 set exam;
 if name = 'Tim';
 if class = 'math';
run;
```

In addition, mixing IF and WHERE conditions within a DATA step also become cumulative.

```
data student1;
 set exam;
 if name = 'Tim';
 where class = 'math';
run;
```

The result is only one record that meets both conditions.

RESULT

Obs	name	class	score
1	Tim	math	9

EXAMPLE ❼

When merging data sets – Be careful when subsetting; in general, use IF statement to subet after merging data sets.

Previous examples applied conditions on one data set. This example merges the following two sample data sets to show the difference in using WHERE and IF statements. While either WHERE or IF statement can be used when merging data sets, you need to be aware that different results may appear depending on your source data sets.

```
data school;
  input name $ class $ score ;
  cards;
A math 10
B history 10
C science 10
;
run;
```

```
data school_data;
  input name $ class $ score ;
  cards;
A math 10
B history 8
C science 7
;
run;
```

Below are two DATA steps using WHERE or IF statement respectively. The condition is to select records when score = 10.

```
data school_where;
   merge school school_data;
   by name;

   * subsets BEFORE merging;
   where score = 10;          ❼
run;
```

```
data school_if;
   merge school school_data;
   by name;

   * subsets AFTER merging;
    if score = 10;
run;
```

Since the WHERE statement applies the subset condition *before* merging the data sets, all records from the SCHOOL data set are selected and only the first record from the SCHOOL_DATA data set is selected. All records in the SCHOOL data set have score = 10 and only the first record in the SCHOOL_DATA data has score = 10. After subsetting the data sets, the merge process will replace the score from the first record in the SCHOOL data set with

the score from the first record in the SCHOOL_DATA data set, which in this case is the same value, 10. Because of this, all records from the first data set are kept in the SCHOOL_WHERE data set.

From the SAS log below for the WHERE statement, you can see that the WHERE condition was applied to the SCHOOL data set to read in three observations and to the SCHOOL_DATA data set to read in one observation before merging the two data sets. The two data sets are then merged to result in three observations.

```
51   data school_where;
52      merge school school_data;
53      by name;
54
55      * subsets BEFORE merging;
56      where score = 10;
57   run;

NOTE: There were 3 observations read from the data set WORK.SCHOOL.
      WHERE score=10;
NOTE: There were 1 observations read from the data set WORK.SCHOOL_DATA.
      WHERE score=10;
NOTE: The data set WORK.SCHOOL_WHERE has 3 observations and 3 variables.
```

From the SAS log below for the IF statement, you can see that all three observations from both SCHOOL and SCHOOL_DATA data sets are read before merging the two data sets. The IF statement is applied to the merged data set to result with one observation.

```
58   data school_if;
59      merge school school_data;
60      by name;
61
62      * subsets AFTER merging;
63      if score = 10;
64   run;

NOTE: There were 3 observations read from the data set WORK.SCHOOL.
NOTE: There were 3 observations read from the data set WORK.SCHOOL_DATA.
NOTE: The data set WORK.SCHOOL_IF has 1 observations and 3 variables.
```

In general, you will want to use the IF statement to apply the subset condition *after* merging the data sets. This approach will first merge the two data sets as shown in the intermediate data set below. The score values from the SCHOOL_DATA data set override score value from the SCHOOL data set for the same corresponding NAME values. Notice that once the condition is applied to the intermediate data set, it is easy to determine that only the first record will be selected.

Intermediate data set *before* subsetting when merge order is school and school_data

obs	name	class	score
1	A	math	10
2	B	history	8
3	C	science	7

Below are the two different data sets from using WHERE and IF statements. For the SCHOOL_WHERE data set, Be careful to use the correct subset method since the results could be very different.

SCHOOL_WHERE data set

obs	name	class	score
1	A	math	10
2	B	history	10
3	C	science	10

SCHOOL_IF data set

obs	name	class	score
1	A	math	10

SUMMARY

Knowing the similarities and differences in WHERE and IF conditions are important in taking advantage of these two programming approaches.

Make sure you apply the following rules when determining which approach to take when subsetting your data set using the DATA step. If your subset condition does not meet the requirements below, then the WHERE and IF statements should produce identical results. For cases such as this, use the WHERE statement since it is more efficient. Note that having both WHERE and IF statements within the same DATA step has a cumulative effect.

- Can use WHERE statement when only specifying data set variables
- Use IF statement when specifying automatic variables or new variables created within DATA step
- Use IF statement when specifying FIRST.BY or LAST. BY variables
- Use IF statement when specifying data set options such as OBS = , POINT = or FIRSTOBS =
- In general, use IF statement when merging data sets to apply subset condition *after* merging data set
- Use WHERE statement when specifying indexes

REFERENCES

Grant, Paul, "Simplifying Complex Character Comparisons by Using the IN Operator and the Colon (:) Operator Modifier", Coder's Corner,

Gupta, Sunil and Curt Edmonds, *Sharpening Your SAS Skills*, Boca Raton, FL: Chapman & Hall/CRC Press, 2005

Ma, Juliana Meimei and Schlotzhauer, Sandra , "How and When to Use WHERE", Beginning Tutorials

Scerbo, Marge, "Tips for Manipulating Data", Beginning Tutorials", SUGI 28

Suligavi, Raj and Yellanki Jyotheeswar, "Advance Tips for Manipulating Data in commonly used SAS Procedures", Technical Techniques, PharmaSUG 2005

YOUR SAS TECHNOLOGY REPORT for June 14, 2005 - FAQ: When do I use a WHERE statement versus an IF statement to subset a data set? http://support.sas.com/faq/042/FAQ04278.html?ETS=2799&PID=94063

CONTACT INFORMATION

The author welcomes your comments and suggestions.

Sunil K. Gupta
Associate Director of Statistical Programming
Quintiles
325 E. Hillcrest Dr., Suite 200
Thousand Oaks, CA 91360
Phone: (805)-557-7700 E-mail: Sunil.Gupta@Quintiles.com

Sunil is the Associate Director of Statistical Programming at Quintiles. He has been using SAS® software for over 14 years and is a SAS Base Certified Professional. He has participated in over 6 successful FDA submissions. His projects with pharmaceutical companies include the development of a Macro-Based Application for Report Generation and Customized Plots and Charts. He is also the author of *Quick Results with the Output Delivery System*, developer of over five SAS programming classes, developer of Clinical Trial Reporting Templates for quick generation of tables, lists and graphs and was a SAS Institute Quality Partner™ for over 5 years. Most recently, he released his new book, *Data Management and Reporting Made Easy with SAS Learning Edition 2.0,* and has co-authored the book *Sharpening Your SAS Skills.*