

Conditional Processing Using the Case Expression in PROC SQL

Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, California

Abstract

The SQL[®] procedure supports conditionally selecting result values from rows in a table (or view) in the form of a case expression. Similar to an IF-THEN construct in the DATA step, a case expression uses one or more WHEN-THEN clause(s) to conditionally process some but not all the rows in a table. An optional ELSE expression can be specified to handle an alternate action should none of the expression(s) identified in the WHEN condition(s) be satisfied. This paper will illustrate the basic syntax associated with the two forms of case expression along with examples of each.

Introduction

It is frequently necessary to test and evaluate one or more conditions as true or false. From a programming perspective, the evaluation of a condition determines which of the alternate paths a program will follow. Conditional logic in the SQL procedure is traditionally implemented in a WHERE clause to reference constants and relationships among columns and values. The SQL procedure allows the identification and assignment of data values in a SELECT statement using case-expressions.

This paper presents the power, and simplicity, of using case expressions to perform conditional processing in the SQL procedure. Two forms of case expressions will be illustrated for users to consider when confronted with conditional logic scenarios: simple and searched along with brief explanations, guidelines and “simple” coding techniques.

Example Tables

The data used in the examples consist of a selection of movies that I’ve viewed over the years. The MOVIES table contains four character columns: title, category, studio, and rating, and two numeric columns: length and year, and is illustrated in Figure 1.

	Title	Length	Category	Year	Studio	Rating
1	Brave Heart	177	Action Adventure	1995	Paramount Pictures	R
2	Casablanca	103	Drama	1942	MGM / UA	PG
3	Christmas Vacation	97	Comedy	1989	Warner Brothers	PG-13
4	Coming to America	116	Comedy	1988	Paramount Pictures	R
5	Dracula	130	Horror	1993	Columbia TriStar	R
6	Dressed to Kill	105	Drama Mysteries	1980	Filmways Pictures	R
7	Forrest Gump	142	Drama	1994	Paramount Pictures	PG-13
8	Ghost	127	Drama Romance	1990	Paramount Pictures	PG-13
9	Jaws	125	Action Adventure	1975	Universal Studios	PG
10	Jurassic Park	127	Action	1993	Universal Pictures	PG-13
11	Lethal Weapon	110	Action Cops & Robber	1987	Warner Brothers	R
12	Michael	106	Drama	1997	Warner Brothers	PG-13
13	National Lampoon's Vacation	98	Comedy	1983	Warner Brothers	PG-13
14	Pollgeist	115	Horror	1982	MGM / UA	PG
15	Rocky	120	Action Adventure	1976	MGM / UA	PG
16	Scarface	170	Action Cops & Robber	1983	Universal Studios	R
17	Silence of the Lambs	118	Drama Suspense	1991	Orion	R
18	Star Wars	124	Action Sci-Fi	1977	Lucas Film Ltd	PG
19	The Hunt for Red October	135	Action Adventure	1989	Paramount Pictures	PG
20	The Terminator	108	Action Sci-Fi	1984	Live Entertainment	R
21	The Wizard of Oz	101	Adventure	1939	MGM / UA	G
22	Titanic	194	Drama Romance	1997	Paramount Pictures	PG-13

Figure 1. MOVIES Table

The ACTORS table consists of thirteen rows and three columns: title, actor_leading, and actor_supporting. Each column is defined as character, as shown in Figure 2.

	Title	Actor_Leading	Actor_Supporting
1	Brave Heart	Mel Gibson	Sophie Marceau
2	Christmas Vacation	Chevy Chase	Beverly D'Angelo
3	Coming to America	Eddie Murphy	Arsenio Hall
4	Forrest Gump	Tom Hanks	Sally Field
5	Ghost	Patrick Swayze	Demi Moore
6	Lethal Weapon	Mel Gibson	Danny Glover
7	Michael	John Travolta	Andie MacDowell
8	National Lampoon's Vacation	Chevy Chase	Beverly D'Angelo
9	Rocky	Sylvester Stallone	Talia Shire
10	Silence of the Lambs	Anthony Hopkins	Jodie Foster
11	The Hunt for Red October	Sean Connery	Alec Baldwin
12	The Terminator	Arnold Schwarzenegger	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet

Figure 2. ACTORS Table

Conditional Logic Scenarios

A powerful and necessary programming technique in the SAS® software is its ability to perform different actions depending on whether a programmer-specified condition evaluates to true or false. The method used to accomplish this is to use one or more conditional statements, expressions, and constructs to build a level of intelligence in a program or application. Conditional logic scenarios in the DATA step are frequently implemented using IF-THEN / ELSE and SELECT statements. The SQL procedure also supports logic scenarios and is implemented with a coding technique known as a CASE expression. The remaining topics presented in this paper will illustrate the implementation of logic scenarios in the DATA step and SQL procedure.

Conditional Logic with IF-THEN / ELSE

The IF-THEN / ELSE construct in the DATA step enables a sequence of conditions to be assigned that when executed proceeds through the sequence of logic conditions until a match in an expression is found or until all conditions are exhausted. The example shows a character variable `Movie_Length` being assigned a value of either “Shorter Length”, “Average Length”, or “Longer Length” based on the mutually exclusive conditions specified in the IF-THEN and ELSE conditions. Although not required, an ELSE condition serves as an effective technique for continuing processing to the next specified condition when a match is not found. An ELSE condition can also be useful as a “catch-all” to prevent a missing value from being assigned.

Code:

```
DATA IF_THEN_EXAMPLE;
  ATTRIB Movie_Length LENGTH=$14 LABEL='Movie Length';
  SET MOVIES;
  IF LENGTH < 120 THEN Movie_Length = 'Shorter Length';
  ELSE IF LENGTH > 160 THEN Movie_Length = 'Longer Length';
  ELSE Movie_Length = 'Average Length';
RUN;
PROC PRINT DATA=IF_THEN_EXAMPLE NOOBS;
  VAR TITLE LENGTH Movie_Length;
RUN;
```

Results

Title	Length	Movie_Length
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

Conditional Logic with SELECT

Another form of conditional logic available to users is a **SELECT** statement. Its purpose is to enable a sequence of logic conditions to be constructed in a DATA step by specifying one or more **WHEN** conditions and an optional **OTHERWISE** condition. When executed, processing continues through each WHEN condition until a match is found that satisfies the specified expression. Typically one or more WHEN conditions are specified in descending frequency order representing a series of conditions. The next example shows a value based on the mutually exclusive conditions specified in the sequence of logic conditions of “Shorter Length”, “Average Length”, or “Longer Length” being assigned to the character variable `Movie_Length`. Although not required, the **OTHERWISE** condition can be useful in the assignment of a specific value or as a “catch-all” to prevent a missing value from being assigned.

Code:

```
DATA SELECT_EXAMPLE;
  ATTRIB Movie_Length LENGTH=$14 LABEL='Movie Length';
  SET MOVIES;
  SELECT;
    WHEN (LENGTH < 120) Movie_Length = 'Shorter Length';
    WHEN (LENGTH > 160) Movie_Length = 'Longer Length';
    OTHERWISE Movie_Length = 'Average Length';
  END;
RUN;
PROC PRINT DATA=SELECT_EXAMPLE NOOBS;
  VAR TITLE LENGTH Movie_Length;
RUN;
```

Results

Title	Length	Movie_Length
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

Conditional Logic with CASE Expressions

Another form of conditional logic available to users is a case expression. Its purpose is to provide a way of conditionally selecting result values from each row in a table (or view). Similar to an IF-THEN/ELSE or SELECT construct in the DATA step, a case expression can only be specified in the SQL procedure. It supports a WHEN-THEN clause to conditionally process some but not all the rows in a table. An optional ELSE expression can be specified to handle an alternative action should none of the expression(s) identified in the WHEN condition(s) not be satisfied. A case expression must be a valid SQL expression and conform to syntax rules similar to DATA step SELECT-WHEN statements. Even though this topic is best explained by example, a quick look at the syntax follows.

```
CASE <column-name>  
  WHEN when-condition THEN result-expression  
  <WHEN when-condition THEN result-expression> ...  
  <ELSE result-expression>  
END
```

A column-name can optionally be specified as part of the CASE-expression. If present, it is automatically made available to each when-condition, and is classified as a simple case expression. When it is not specified, the column-name must be coded in each when-condition, and is classified as a searched case expression. If a when-condition is satisfied by a row in a table (or view), then it is considered “true” and the result-expression following the THEN keyword is processed. The remaining WHEN conditions in the case expression are skipped. If a when-condition is “false”, the next when-condition is evaluated. SQL evaluates each when-condition until a “true” condition is found or in the event all when-conditions are “false”, it then executes the ELSE expression and assigns its value to the CASE expression’s result. A missing value is assigned to a case expression when an ELSE expression is not specified and each when-condition is “false”.

A simple case expression provides a handy way to perform the simplest type of comparisons. The syntax requires a column name from an underlying table to be specified as part of the case expression. This not only eliminates having to continually repeat the column name in each WHEN condition, it also reduces the number of keystrokes, making the code easier to read (and support).

In the next example, a simple case expression is illustrated that shows a character variable `Movie_Length` being assigned with the AS keyword. A value of “Shorter Length” for movie lengths less than 120 minutes, “Longer Length” for movie lengths greater than 160 minutes, or “Average Length” for all other movie lengths is assigned to the newly created column. Although not required, an ELSE condition can be useful in the assignment of a specific value or as a “catch-all” to prevent a missing value from being assigned, as shown below.

SQL Code

```
PROC SQL;  
  SELECT TITLE,  
         LENGTH,  
         CASE LENGTH  
           WHEN < 120 THEN 'Shorter Length'  
           WHEN > 160 THEN 'Longer Length'  
           ELSE 'Average Length'  
         END AS Movie_Length  
  FROM MOVIES;  
QUIT;
```

Results

<u>Title</u>	<u>Length</u>	<u>Movie Length</u>
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

In the next example, a searched case expression is illustrated. A searched case expression in the SQL procedure provides users with the capability to perform more complex comparisons. Although the number of keystrokes can be more than with a simple case expression, the searched case expression offers the greatest flexibility and is the primary form used by SQL'ers. The noticeable absence of a column name as part of the case expression permits any number of columns to be specified from the underlying table(s) in the WHEN-THEN/ELSE logic scenarios.

The next example shows a searched case expression being used to assign the character variable `Movie_Length` with the `AS` keyword. A value of "Shorter Length" for movie lengths less than 120 minutes, "Longer Length" for movie lengths greater than 160 minutes, or "Average Length" for all other movie lengths is assigned to the newly created column. Although not required, an ELSE condition can be useful in the assignment of a specific value or as a "catch-all" to prevent a missing value from being assigned.

SQL Code

```
PROC SQL;
  SELECT TITLE,
         LENGTH,
         CASE
           WHEN LENGTH < 120 THEN 'Shorter Length'
           WHEN LENGTH > 160 THEN 'Longer Length'
           ELSE 'Average Length'
         END AS Movie_Length
  FROM MOVIES;
QUIT;
```

Results

<u>Title</u>	<u>Length</u>	<u>Movie Length</u>
Brave Heart	177	Longer Length
Casablanca	103	Shorter Length
Christmas Vacation	97	Shorter Length
Coming to America	116	Shorter Length
Dracula	130	Average Length
Dressed to Kill	105	Shorter Length
Forrest Gump	142	Average Length
Ghost	127	Average Length
Jaws	125	Average Length
Jurassic Park	127	Average Length
Lethal Weapon	110	Shorter Length
Michael	106	Shorter Length
National Lampoon's Vacation	98	Shorter Length
Poltergeist	115	Shorter Length
Rocky	120	Average Length
Scarface	170	Longer Length
Silence of the Lambs	118	Shorter Length
Star Wars	124	Average Length
The Hunt for Red October	135	Average Length
The Terminator	108	Shorter Length
The Wizard of Oz	101	Shorter Length
Titanic	194	Longer Length

As previously mentioned, searched case expressions provide users with the capability to perform more complex logic comparisons. Combined with logical and comparison operators, searched case expressions along with their WHERE clause counterparts, provide the capabilities to construct complex logic scenarios. In the next example a listing of “Action” and “Comedy” movies are displayed. Using a searched case expression, a value of “Shorter Length” for movie lengths less than 120 minutes, “Longer Length” for movie lengths greater than 160 minutes, or “Average Length” for all other movie lengths is assigned to the newly created column. A column heading of Movie_Type is assigned to the new column with the AS keyword.

SQL Code

```
PROC SQL;
  SELECT TITLE, RATING, LENGTH, CATEGORY,
    CASE
      WHEN UPCASE(CATEGORY) CONTAINS 'ACTION' AND LENGTH < 120 THEN 'Action Short'
      WHEN UPCASE(CATEGORY) CONTAINS 'ACTION' AND LENGTH > 160 THEN 'Action Long'
      WHEN UPCASE(CATEGORY) CONTAINS 'ACTION' AND
        LENGTH BETWEEN 120 AND 160 THEN 'Action Medium'
      WHEN UPCASE(CATEGORY) CONTAINS 'COMEDY' AND LENGTH < 120 THEN 'Comedy Short'
      WHEN UPCASE(CATEGORY) CONTAINS 'COMEDY' AND LENGTH > 160 THEN 'Comedy Long'
      WHEN UPCASE(CATEGORY) CONTAINS 'COMEDY' AND
        LENGTH BETWEEN 120 AND 160 THEN 'Comedy Medium'
      ELSE 'Not Interested'
    END AS MOVIE_TYPE
  FROM MOVIES
  WHERE UPCASE(CATEGORY) CONTAINS 'ACTION' OR 'COMEDY';
QUIT;
```

Results

Title	Rating	Length	Category	Movie Type
Brave Heart	R	177	Action Adventure	Action Long
Casablanca	PG	103	Drama	Not Interested
Christmas Vacation	PG-13	97	Comedy	Comedy Short
Coming to America	R	116	Comedy	Comedy Short
Dracula	R	130	Horror	Not Interested
Dressed to Kill	R	105	Drama Mysteries	Not Interested
Forrest Gump	PG-13	142	Drama	Not Interested
Ghost	PG-13	127	Drama Romance	Not Interested
Jaws	PG	125	Action Adventure	Action Medium
Jurassic Park	PG-13	127	Action	Action Medium
Lethal Weapon	R	110	Action Cops & Robber	Action Short
Michael	PG-13	106	Drama	Not Interested
National Lampoon's Vacation	PG-13	98	Comedy	Comedy Short
Poltergeist	PG	115	Horror	Not Interested
Rocky	PG	120	Action Adventure	Action Medium
Scarface	R	170	Action Cops & Robber	Action Long
Silence of the Lambs	R	118	Drama Suspense	Not Interested
Star Wars	PG	124	Action Sci-Fi	Action Medium
The Hunt for Red October	PG	135	Action Adventure	Action Medium
The Terminator	R	108	Action Sci-Fi	Action Short
The Wizard of Oz	G	101	Adventure	Not Interested
Titanic	PG-13	194	Drama Romance	Not Interested

Conclusion

The SQL procedure is a wonderful language for SAS users to explore and use in a variety of application situations. This paper has presented code examples, explanations, guidelines and “simple” techniques for users to consider when confronted with conditional logic scenarios. The author encourages you to explore these and other techniques to make your PROC SQL experience a more exciting one.

References

- Lafler, Kirk Paul (2013). *PROC SQL: Beyond the Basics Using SAS*, Second Edition, SAS Institute Inc., Cary, NC, USA.
- Lafler, Kirk Paul (2012), “Conditional Processing Using the Case Expressions in PROC SQL,” Pharma SAS Users Group (PharmaSUG) 2012 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2012), “Conditional Processing Using the Case Expressions in PROC SQL,” Iowa SAS Users Group (IowaSUG) 2012 Half-Day User Group Meeting, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2011), “Powerful and Sometimes Hard-to-find PROC SQL Features,” PharmaSUG 2011 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2010), “DATA Step and PROC SQL Programming Techniques,” Ohio SAS Users Group (OSUG) 2010 One-Day Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2009), “DATA Step and PROC SQL Programming Techniques,” South Central SAS Users Group (SCSUG) 2009 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul (2009), “DATA Step versus PROC SQL Programming Techniques,” Sacramento Valley SAS Users Group 2009 Meeting, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul, *Advanced SAS® Programming Tips and Techniques*; Software Intelligence Corporation, Spring Valley, CA, USA; 1987-2007.
- Lafler, Kirk Paul (2007), “Undocumented and Hard-to-find PROC SQL Features,” Proceedings of the PharmaSUG 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, USA.
- Lafler, Kirk Paul and Ben Cochran (2007), “A Hands-on Tour Inside the World of PROC SQL Features,” Proceedings of the SAS Global Forum (SGF) 2007 Conference, Software Intelligence Corporation, Spring Valley, CA, and The Bedford Group, USA.
- Lafler, Kirk Paul (2006), “A Hands-on Tour Inside the World of PROC SQL,” Proceedings of the 31st Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2005), *"Manipulating Data with PROC SQL,"* Proceedings of the 30th Annual SAS Users Group International Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2004). *PROC SQL: Beyond the Basics Using SAS*, SAS Institute Inc., Cary, NC, USA.

SAS[®] SQL Procedure User's Guide, Version 9.3; SAS Institute Inc., Cary, NC, USA; 2012.

Acknowledgments

The author thanks Alejandro Farias and John Taylor, SCSUG 2014 Conference Chairs, for accepting my abstract and paper; as well as SAS Institute Inc.; and the SCSUG Executive Committee for organizing a great conference!

Trademark Citations

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

About The Author

Kirk Paul Lafler is consultant and founder of Software Intelligence Corporation and has been using SAS since 1979. He is a SAS Certified Professional, provider of IT consulting services, trainer to SAS users around the world, and sasCommunity.org emeritus Advisory Board member. As the author of six books including Google[®] Search Complete (Odyssey Press. 2014); PROC SQL: Beyond the Basics Using SAS, Second Edition (SAS Press. 2013); PROC SQL: Beyond the Basics Using SAS (SAS Press. 2004); Kirk has written more than five hundred papers and articles, been an Invited speaker and trainer at four hundred-plus SAS International, regional, special-interest, local, and in-house user group conferences and meetings, and is the recipient of 23 "Best" contributed paper, hands-on workshop (HOW), and poster awards.

Comments and suggestions can be sent to:

Kirk Paul Lafler
Senior SAS[®] Consultant, Application Developer, Data Scientist, Trainer and Author
Software Intelligence Corporation
E-mail: KirkLafler@cs.com
LinkedIn: <http://www.linkedin.com/in/KirkPaulLafler>
Twitter: @sasNerd