Conditional Processing in the SAS® Software by Example

Charu Shankar, SAS Institute Canada, Toronto, Canada
Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, California

Abstract

Conditional processing is at the heart & core of computer programming. The SAS software supports conditionally selecting result values from rows in a table (or view) in the form of DATA step subsetting IF, IF-Then-Else, Select-When-Otherwise, and the IFN/IFC statements, the powerful PROC SQL Case Expression, and PROC FORMAT. Learn about best practices while crafting your conditional statements and much more.

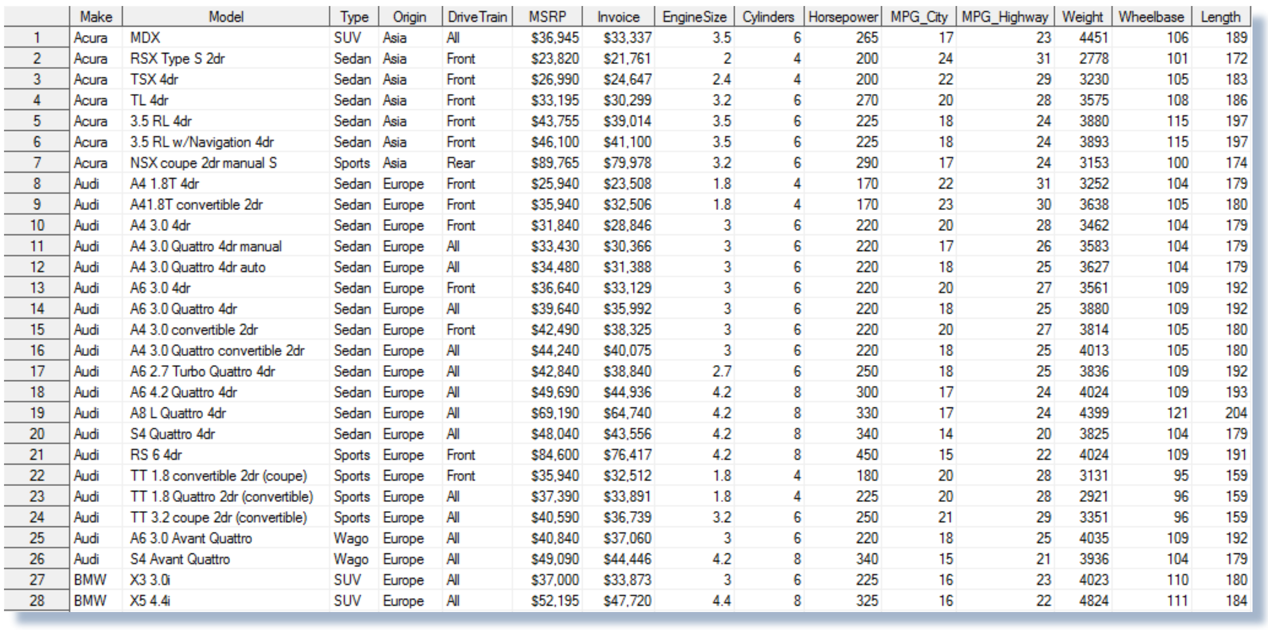
Introduction

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. Another important technique used in conditional processing is to reclassify (or restructure) data in SAS data sets. As with most things in the SAS software, users have a variety of options to choose from when performing conditional logic processing. From DATA step subsetting IF, IF-THEN-ELSE, SELECT-WHEN-OTHERWISE, and IFN/IFC statements; a Case expression in PROC SQL; and user-defined formats with PROC FORMAT.

This paper presents the power, and simplicity, of using the various conditional processing approaches found in the SAS software. We'll share guidelines, best practice scenarios, along with our experience using these powerful statements, expressions, and procedures using an assortment of examples. case expressions to perform conditional processing in the SQL procedure.

Table (Data Set) Used in Examples

The data set used in all the examples in this paper is the SASHELP.CARS. The SASHELP.CARS data set contains 428 observations and 15 variables, illustrated below.



	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17	23	4451	106	189
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200	24	31	2778	101	172
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18	24	3893	115	197
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17	24	3153	100	174
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22	31	3252	104	179
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23	30	3638	105	180
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220	20	28	3462	104	179
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220	17	26	3583	104	179
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220	18	25	3627	104	179
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220	20	27	3561	109	192
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3	6	220	18	25	3880	109	192
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3	6	220	20	27	3814	105	180
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3	6	220	18	25	4013	105	180
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250	18	25	3836	109	192
18	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300	17	24	4024	109	193
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330	17	24	4399	121	204
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340	14	20	3825	104	179
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450	15	22	4024	109	191
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports	Europe	Front	$35,940	$32,512	1.8	4	180	20	28	3131	95	159
23	Audi	TT 1.8 Quattro 2dr (convertible)	Sports	Europe	All	$37,390	$33,891	1.8	4	225	20	28	2921	96	159
24	Audi	TT 3.2 coupe 2dr (convertible)	Sports	Europe	All	$40,590	$36,739	3.2	6	250	21	29	3351	96	159
25	Audi	A6 3.0 Avant Quattro	Wago	Europe	All	$40,840	$37,060	3	6	220	18	25	4035	109	192
26	Audi	S4 Avant Quattro	Wago	Europe	All	$49,090	$44,446	4.2	8	340	15	21	3936	104	179
27	BMW	X3 3.0i	SUV	Europe	All	$37,000	$33,873	3	6	225	16	23	4023	110	180
28	BMW	X5 4.4i	SUV	Europe	All	$52,195	$47,720	4.4	8	325	16	22	4824	111	184

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.

Subsetting IF

The subsetting IF construct in the DATA step allows users to subset rows of data

. . .

Code:

Results

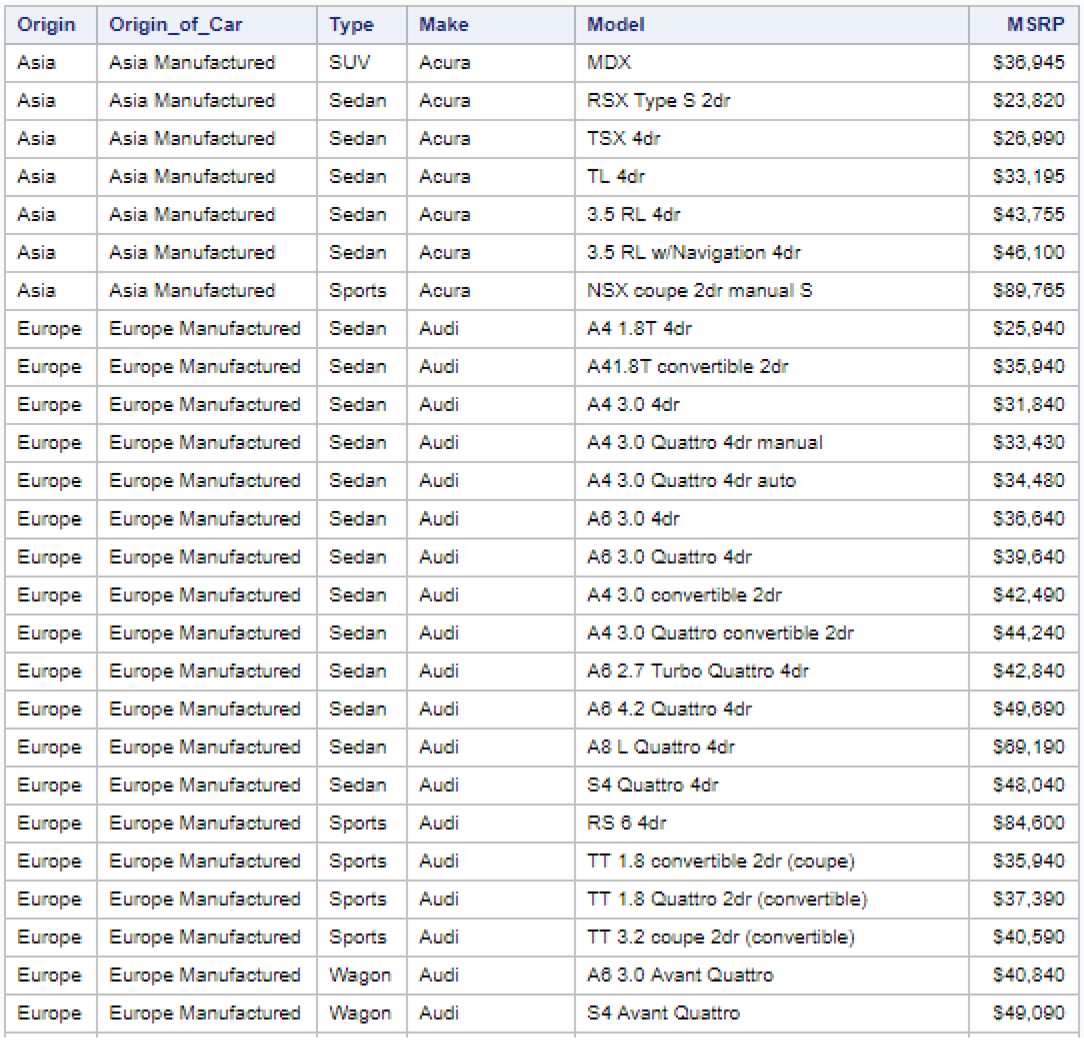
Conditional Logic with IF-THEN / ELSE

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 Orgin_of_Car being assigned a value of either "Asia Manufactured", "Europe Manufactured", "USA Manufactured" or "Unknown Manufacturer" based on the mutually exclusive conditions specified in the IF-THEN and ELSE conditions. Although not required, an ELSE condition serves as an effective "best practice" 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 Origin_of_Car LENGTH=$19 LABEL='Origin of Car' ;
  SET SASHELP.CARS ;
  IF UPCASE(Origin) = 'ASIA' THEN Origin_of_Car = 'Asia Manufactured' ;
  ELSE IF UPCASE(Origin) = 'EUROPE' THEN Origin_of_Car = 'Europe Manufactured' ;
  ELSE IF UPCASE(Origin) = 'USA' THEN Origin_of_Car = 'USA Manufactured' ;
  ELSE Origin_of_Car = 'Unknown Manufacturer' ;
RUN ;
PROC PRINT DATA=IF_THEN_EXAMPLE NOOBS ;
  VAR Origin Origin_of_Car Type Make Model MSRP ;
RUN ;
```

Results



Origin	Origin_of_Car	Type	Make	Model	MSRP
Asia	Asia Manufactured	SUV	Acura	MDX	$36,945
Asia	Asia Manufactured	Sedan	Acura	RSX Type S 2dr	$23,820
Asia	Asia Manufactured	Sedan	Acura	TSX 4dr	$26,990
Asia	Asia Manufactured	Sedan	Acura	TL 4dr	$33,195
Asia	Asia Manufactured	Sedan	Acura	3.5 RL 4dr	$43,755
Asia	Asia Manufactured	Sedan	Acura	3.5 RL w/Navigation 4dr	$46,100
Asia	Asia Manufactured	Sports	Acura	NSX coupe 2dr manual S	$89,765
Europe	Europe Manufactured	Sedan	Audi	A4 1.8T 4dr	$25,940
Europe	Europe Manufactured	Sedan	Audi	A41.8T convertible 2dr	$35,940
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 4dr	$31,840
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro 4dr manual	$33,430
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro 4dr auto	$34,480
Europe	Europe Manufactured	Sedan	Audi	A6 3.0 4dr	$36,640
Europe	Europe Manufactured	Sedan	Audi	A6 3.0 Quattro 4dr	$39,640
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 convertible 2dr	$42,490
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro convertible 2dr	$44,240
Europe	Europe Manufactured	Sedan	Audi	A6 2.7 Turbo Quattro 4dr	$42,840
Europe	Europe Manufactured	Sedan	Audi	A6 4.2 Quattro 4dr	$49,690
Europe	Europe Manufactured	Sedan	Audi	A8 L Quattro 4dr	$69,190
Europe	Europe Manufactured	Sedan	Audi	S4 Quattro 4dr	$48,040
Europe	Europe Manufactured	Sports	Audi	RS 6 4dr	$84,600
Europe	Europe Manufactured	Sports	Audi	TT 1.8 convertible 2dr (coupe)	$35,940
Europe	Europe Manufactured	Sports	Audi	TT 1.8 Quattro 2dr (convertible)	$37,390
Europe	Europe Manufactured	Sports	Audi	TT 3.2 coupe 2dr (convertible)	$40,590
Europe	Europe Manufactured	Wagon	Audi	A6 3.0 Avant Quattro	$40,840
Europe	Europe Manufactured	Wagon	Audi	S4 Avant Quattro	$49,090

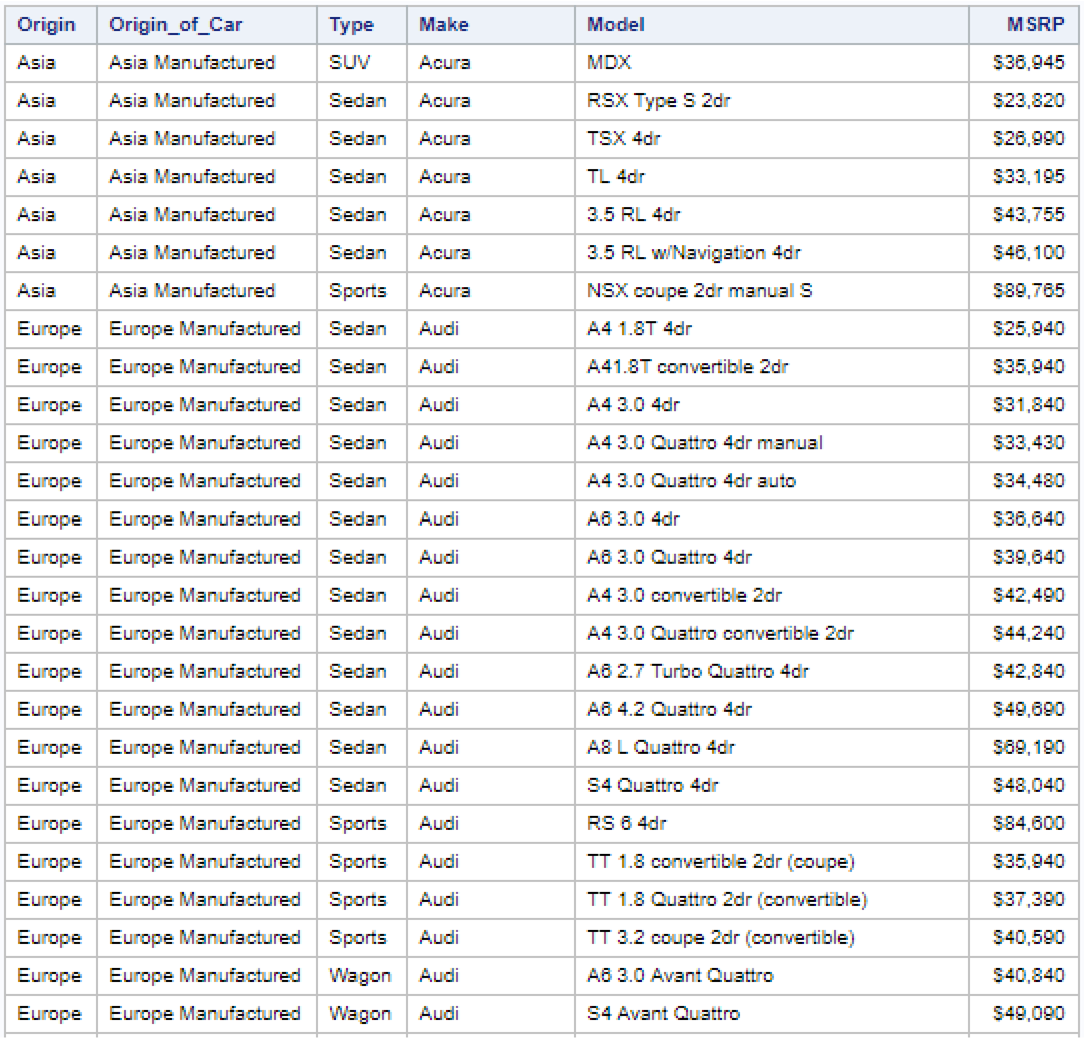
Conditional Logic with SELECT

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.

<u>**Code:**</u>

```
DATA SELECT_WHEN_EXAMPLE ;
  SET SASHELP.CARS ;
  SELECT ;
    WHEN (UPCASE(Origin) = 'ASIA')   Origin_of_Car = 'Asia Manufactured' ;
    WHEN (UPCASE(Origin) = 'EUROPE') Origin_of_Car = 'Europe Manufactured' ;
    WHEN (UPCASE(Origin) = 'USA')    Origin_of_Car = 'USA Manufactured' ;
    OTHERWISE Origin_of_Car = 'Unknown Manufacturer' ;
  END ;
RUN ;
PROC PRINT DATA=SELECT_WHEN_EXAMPLE NOOBS ;
  VAR Origin Origin_of_Car Type Make Model MSRP ;
RUN ;
```

Results



Origin	Origin_of_Car	Type	Make	Model	MSRP
Asia	Asia Manufactured	SUV	Acura	MDX	$36,945
Asia	Asia Manufactured	Sedan	Acura	RSX Type S 2dr	$23,820
Asia	Asia Manufactured	Sedan	Acura	TSX 4dr	$26,990
Asia	Asia Manufactured	Sedan	Acura	TL 4dr	$33,195
Asia	Asia Manufactured	Sedan	Acura	3.5 RL 4dr	$43,755
Asia	Asia Manufactured	Sedan	Acura	3.5 RL w/Navigation 4dr	$46,100
Asia	Asia Manufactured	Sports	Acura	NSX coupe 2dr manual S	$89,765
Europe	Europe Manufactured	Sedan	Audi	A4 1.8T 4dr	$25,940
Europe	Europe Manufactured	Sedan	Audi	A41.8T convertible 2dr	$35,940
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 4dr	$31,840
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro 4dr manual	$33,430
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro 4dr auto	$34,480
Europe	Europe Manufactured	Sedan	Audi	A6 3.0 4dr	$36,640
Europe	Europe Manufactured	Sedan	Audi	A6 3.0 Quattro 4dr	$39,640
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 convertible 2dr	$42,490
Europe	Europe Manufactured	Sedan	Audi	A4 3.0 Quattro convertible 2dr	$44,240
Europe	Europe Manufactured	Sedan	Audi	A6 2.7 Turbo Quattro 4dr	$42,840
Europe	Europe Manufactured	Sedan	Audi	A6 4.2 Quattro 4dr	$49,690
Europe	Europe Manufactured	Sedan	Audi	A8 L Quattro 4dr	$69,190
Europe	Europe Manufactured	Sedan	Audi	S4 Quattro 4dr	$48,040
Europe	Europe Manufactured	Sports	Audi	RS 6 4dr	$84,600
Europe	Europe Manufactured	Sports	Audi	TT 1.8 convertible 2dr (coupe)	$35,940
Europe	Europe Manufactured	Sports	Audi	TT 1.8 Quattro 2dr (convertible)	$37,390
Europe	Europe Manufactured	Sports	Audi	TT 3.2 coupe 2dr (convertible)	$40,590
Europe	Europe Manufactured	Wagon	Audi	A6 3.0 Avant Quattro	$40,840
Europe	Europe Manufactured	Wagon	Audi	S4 Avant Quattro	$49,090

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

```
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
```

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

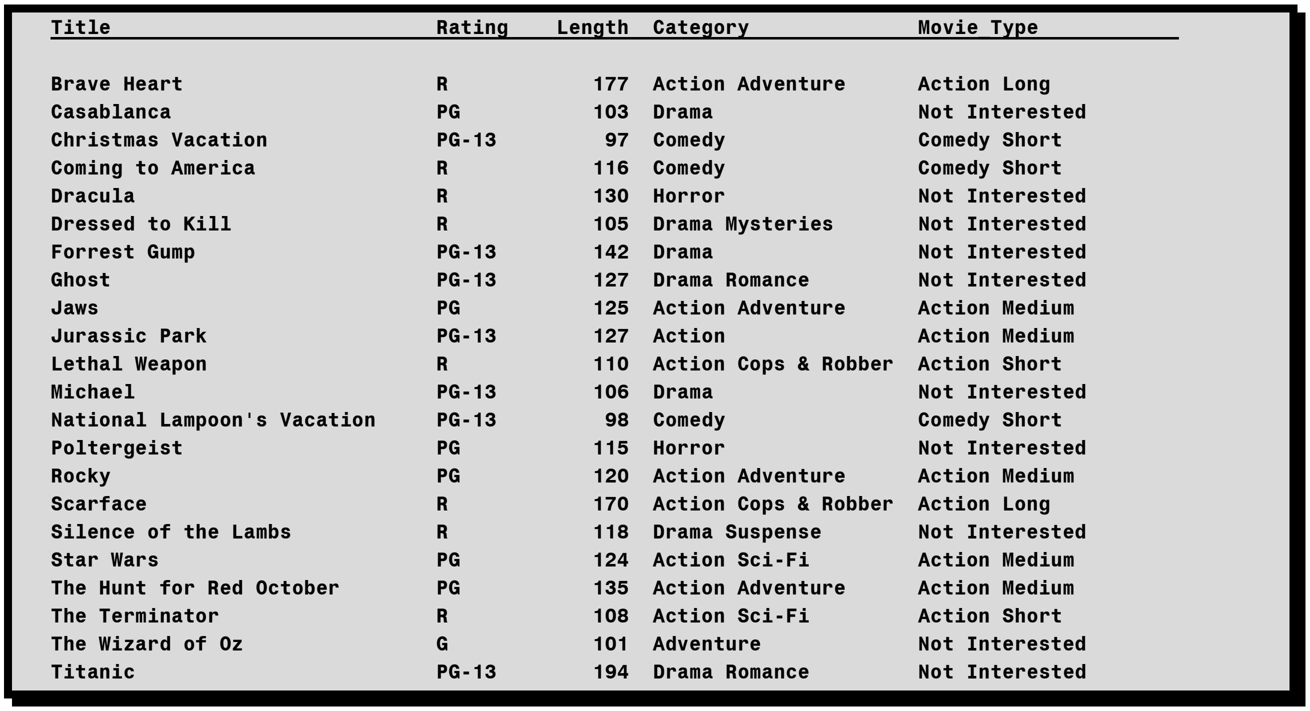
```
Title                          Length     Movie Length

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
```

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.

Trademark Citations

About The Authors

Charu Shankar is a technology trainer, wellness coach, writer and public speaker with proven ability in delivering top quality training resulting in a high degree of satisfaction among clients. As a SAS instructor Charu helps individuals and organizations leverage SAS to learn and use SAS creatively to solve practical business problems. Charu is a popular speaker at SAS user group conferences and has helped train thousands of SAS users. She also helps individuals land their dream SAS job to meet their technology goals and make a contribution to society with their skills. Charu's blog posts are found at blogs.sas.com/content/author/charushankar.

Kirk Paul Lafler is consultant, entrepreneur, and founder at Software Intelligence Corporation and has been using SAS since 1979. He is a SAS Certified Professional, mentor, provider of SAS consulting and training services, educator to SAS users around the world, and an emeritus sasCommunity.org 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 hundreds of papers and articles; served as an Invited speaker, trainer, keynote and section leader at SAS International, regional, special-interest, local, and in-house user group conferences and meetings; and is the recipient of 25 "Best" contributed paper, hands-on workshop (HOW), and poster awards.

Comments and suggestions can be sent to:

Charu Shankar
Technical Trainer, SAS Institute, Canada
E-mail: charu_shankar@hotmail.com
LinkedIn: https://www.linkedin.com/in/charushankar
Twitter: CharuSAS

~ ~ ~ ~ ~ ~ ~ ~

Kirk Paul Lafler
SAS® Consultant, Application Developer, Programmer, Data Analyst, Educator and Author
Software Intelligence Corporation
E-mail: KirkLafler@cs.com
LinkedIn: https://www.linkedin.com/in/KirkPaulLafler/
LinkedIn: https://www.linkedin.com/in/Order-of-Magnitude-Analytics/
Twitter: @sasNerd

https://pdfs.semanticscholar.org/a73a/d95270d0cedfd893ad9a598a1915bed04bf1.pdf