Paper 1653-2018

Data-driven Programming Techniques Using SAS® Metadata

Kirk Paul Lafler, Software Intelligence Corporation

Abstract

Data-driven programming, or data oriented programming (DOP), is a specific programming paradigm where the data, or data structures, itself controls the flow of a program and not the program logic. Often, data-driven programming approaches are applied in organizations with structured data for filtering, aggregating, transforming and calling other programs. SAS® users can easily access metadata content to capture valuable information about the librefs that are currently assigned, the names of the tables available in a libref, whether a data set is empty, how many observations are in a data set, how many character versus numeric variables are in a data set, a variable's attributes, the names of variables associated with simple and composite indexes, and much more. The value of accessing the content of the contents of these read-only SAS metadata data sets called DICTIONARY tables or their counterparts, SASHELP views, is limitless. This paper and presentation explores how SAS metadata can be dynamically created using data-driven programming techniques.

Introduction

The SAS System collects and populates valuable information ("metadata") about SAS libraries, data sets (tables), catalogs, indexes, macros, system options, titles, views and a collection of other read-only tables called dictionary tables. Dictionary tables serve a special purpose by providing system-related information about the current SAS session's SAS databases and applications. When a query is requested against a Dictionary table, SAS automatically launches a discovery process at runtime to collect information pertinent to that table. This information is made available any time after a SAS session is started.

The contents of Dictionary tables and SASHELP views permit a SAS session's activities to be accessed, monitored, and even controlled. This becomes particularly useful in the design and construction of "intelligent" code, programs, and software applications. Since the information can be queried and the results acted upon in a specific operation or task, the various actions may include the allocation of filerefs and/or librefs, the capture and retention of variable lists, the definition and labels associated with table and variable names, whether a data set is empty or the number of observations it contains, and an assortment of other useful information; the construction of dynamic and flexible SAS code; and the development of "custom" user-designed data dictionary repositories.

What Does Data-driven Programming Mean?

Programming languages are often classified by their basic features into one of the many programming paradigms. Three popular programming paradigms in use today by programming professionals are 1) **Procedural programming** – represented by blocks of code being organized logically by function, such as data input, data processing or manipulation, and data / results output; 2) **Object-oriented programming** – represented by a combination of functionality (behaviors) and data (attributes) hidden inside an object which can then be arranged into classes; and 3) **Data-driven programming** – represented by data controlling the flow of execution in a program.

Unlike procedural programming languages where a program's flow of execution is described using a detailed step-by-step logical approach to solving a problem or with object-oriented programming where an object is told how to behave without all the detailed steps that informs the object how to behave. Data-driven programming involves a program that has its decisions and processes (the flow of execution) controlled (or dictated) by the data (or data structures).

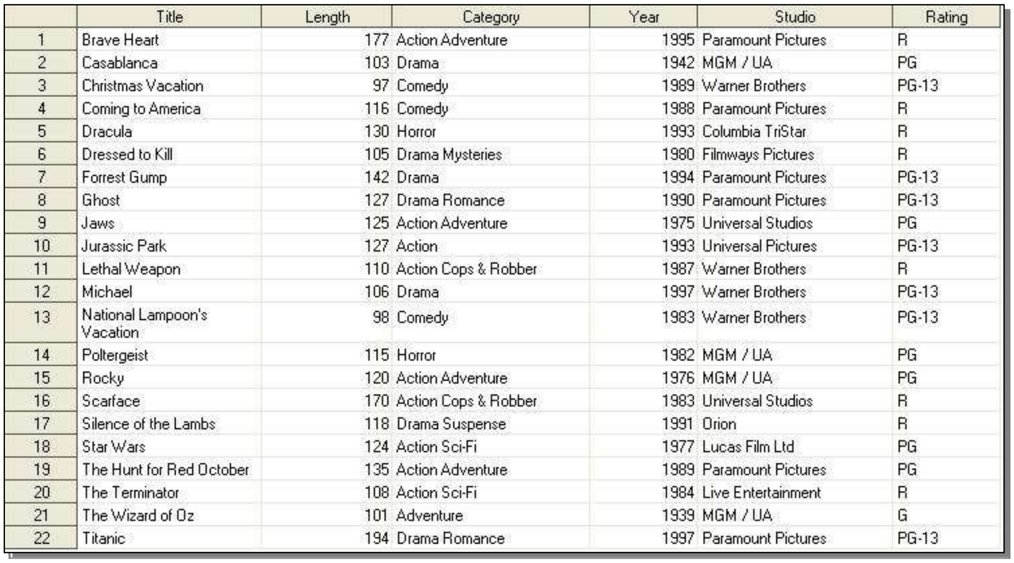
Why Use Data-driven Programming Methods?

Data-driven programming possesses many virtues over its rival programming paradigms including having a default action assigned to it, is generally shorter in length, is more flexible, and is easier to maintain due to a reduction, or elimination, of "hard-coded" values.

Tables Used in Examples

The data used in all the examples in this paper consists of a selection of movies that I've viewed over the years, along with actors. The Movies table consists of six columns: title, length, category, year, studio, and rating. Title, category, studio, and rating are defined as character columns with length and year being defined as numeric columns. The data stored in the Movies table is illustrated below.

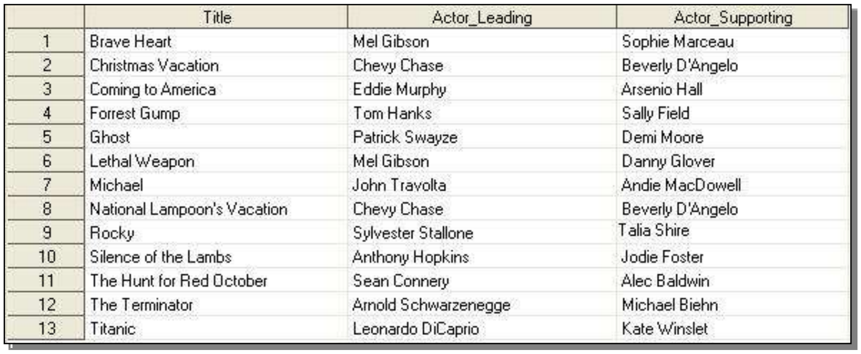
MOVIES Table



	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	Poltergeist	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

The data stored in the ACTORS table is illustrated below.

ACTORS Table



	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 Schwarzenegge	Michael Biehn
13	Titanic	Leonardo DiCaprio	Kate Winslet

Exploring SAS Metadata DICTIONARY Tables and SASHELP Views

SAS users can quickly and conveniently obtain useful information about their SAS session with a number of read-only SAS system tables called DICTIONARY tables. At any time during a SAS session, DICTIONARY tables can be accessed using the libref DICTIONARY in the FROM clause of a PROC SQL SELECT statement to capture information related to currently defined libnames, table names, column names and attributes, formats, and much more. SASHELP views can be accessed using any of your favorite procedures or in the DATA step.

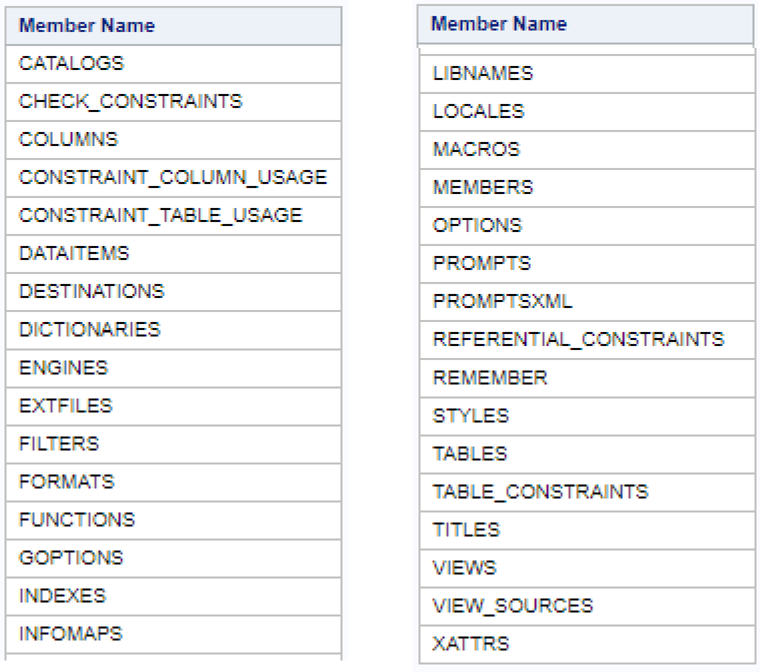
Identifying the Names of the DICTIONARIES Tables and SASHELP Views

SAS users can identify any new Dictionary table release by accessing the read-only DICTIONARIES Dictionary table or VSVIEW SASHELP view. The content of the DICTIONARIES Dictionary table reveals the names of supported Dictionary tables. The following PROC SQL query uses the UNIQUE (or DISTINCT) keyword to generate a listing of existing Dictionary tables.

PROC SQL Code

```
PROC SQL ;
  SELECT UNIQUE MEMNAME
    FROM DICTIONARY.DICTIONARIES ;
QUIT ;
```

Results from DICTIONARY.DICTIONARIES



Member Name
CATALOGS
CHECK_CONSTRAINTS
COLUMNS
CONSTRAINT_COLUMN_USAGE
CONSTRAINT_TABLE_USAGE
DATAITEMS
DESTINATIONS
DICTIONARIES
ENGINES
EXTFILES
FILTERS
FORMATS
FUNCTIONS
GOPTIONS
INDEXES
INFOMAPS

Member Name
LIBNAMES
LOCALES
MACROS
MEMBERS
OPTIONS
PROMPTS
PROMPTSXML
REFERENTIAL_CONSTRAINTS
REMEMBER
STYLES
TABLES
TABLE_CONSTRAINTS
TITLES
VIEWS
VIEW_SOURCES
XATTRS

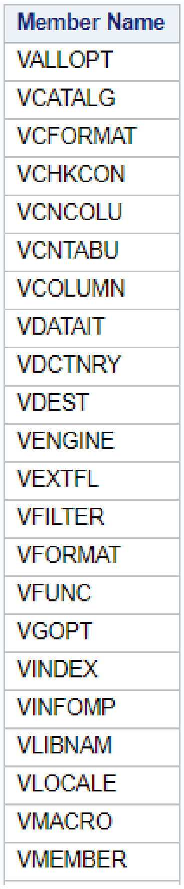
SAS 9.4 currently supports 32 DICTIONARY tables as is illustrated below. Earlier versions of SAS supported fewer Dictionary tables. SAS 9.3 supported 30 DICTIONARY tables; SAS 9.2 supported 29 Dictionary tables; and SAS 9.1 software supported 22 Dictionary tables.

The contents of the VSVIEW SASHELP view reveals the names of supported SASHELP views in SAS 9.4. The following PROC SQL query uses the DISTINCT (or UNIQUE) keyword along with the SUBSTR function to identify a listing of SASHELP views starting with the character value, “V”.

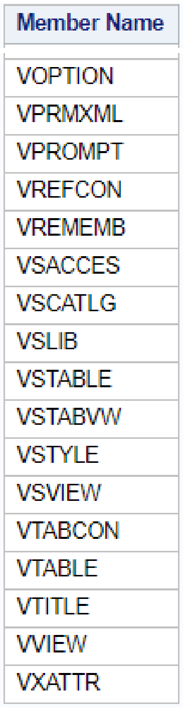
PROC SQL Code

```
PROC SQL ;
 SELECT DISTINCT MEMNAME
  FROM SASHELP.VSVIEW
   WHERE UPCASE(SUBSTR(MEMNAME,1,1)) = 'V' AND
         UPCASE(LIBNAME) = 'SASHELP'
    ORDER BY MEMNAME ;
QUIT ;
```

Results from SASHELP.VSVIEWS



Member Name
VALLOPT
VCATALG
VCFORMAT
VCHKCON
VCNCOLU
VCNTABU
VCOLUMN
VDATAIT
VDCTNRY
VDEST
VENGINE
VEXTFL
VFILTER
VFORMAT
VFUNC
VGOPT
VINDEX
VINFOMP
VLIBNAM
VLOCALE
VMACRO
VMEMBER



Member Name
VOPTION
VPRMXML
VPROMPT
VREFCON
VREMEMB
VSACCES
VSCATLG
VSLIB
VSTABLE
VSTABVW
VSTYLE
VSVIEW
VTABCON
VTABLE
VTITLE
VVIEW
VXATTR

Names and Purpose of Each DICTIONARY Table and SASHELP View

The names and purpose of the DICTIONARY tables and equivalent SASHELP views appear in the following table.

DICTIONARY Table	SASHELP View	Purpose
CATALOGS	**VCATALG**	SAS Catalogs and Catalog-specific Information.
CHECK_CONSTRAINTS	**VCHKCON**	Check Constraints information.
COLUMNS	**VCOLUMN**	Columns from All Tables.
CONSTRAINT_COLUMN_USAGE	**VCNCOLU**	Constraint Column Usage.
CONSTRAINT_TABLE_USAGE	**VCNTABU**	Constraint Table Usage.
DATAITEMS	**VDATAIT**	Information Map Data Items.
DESTINATIONS	**VDEST**	Open ODS Destinations.
DICTIONARIES	**VDCTNRY**	DICTIONARY Tables and their Columns.
ENGINES	**VENGINE**	Available Engines.
EXTFILES	**VEXTFL**	Implicitly-defined File Definitions and Files Defined in FILENAME statements.

FILTERS	**VFILTER**	Information Map Filters.
FORMATS	**VFORMAT**	Available SAS and User-defined Formats and Informats.
FUNCTIONS	**VFUNC**	Available Functions.
GOPTIONS	**VGOPT**	SAS/GRAPH Software Graphics Options.
INDEXES	**VINDEX**	Information related to Defined Indexes.
INFOMAPS	**VINFOMP**	Information Maps.
LIBNAMES	**VLIBNAM**	Information related to SAS Data Libraries.
LOCALES	**VLOCALE**	Available Locales, Regions, Languages and Currency Symbols.
MACROS	**VMACRO**	Information about Defined Macros.
MEMBERS	**VMEMBER**	Information about SAS Defined Tables, Catalogs and Views.
OPTIONS	**VOPTION**	Information about SAS Default System Options.
PROMPTS	**VPROMPT**	Information about Information Map Prompts.
PROMPTSXML	**VPRMXML**	Information Map Prompts XML.
REFERENTIAL_CONSTRAINTS	**VREFCON**	Information about Referential Constraints.
REMEMBER	**VREMEMB**	All Remembered Information.
STYLES	**VSTYLE**	Information about All Styles.
TABLES	**VTABLE**	SAS Tables and Table-specific Information.
TABLE_CONSTRAINTS	**VTABCON**	Information about Table Constraints.
TITLES	**VTITLE**	Information about Defined Titles.
VIEWS	**VVIEW**	Views and View-specific Information.
VIEW_SOURCES	**VSVIEW**	Sources Referenced by View.
XATTRS	**VXATTR**	Extended Attributes.

Displaying DICTIONARY Table Definitions

A dictionary table's definition can be displayed by specifying a DESCRIBE TABLE statement. The results of the statements and clauses used to create each dictionary table can be displayed on the SAS Log. For example, a DESCRIBE TABLE statement is illustrated below to display the CREATE TABLE statement used in building the OPTIONS dictionary table containing current SAS System option settings.

PROC SQL Code

```
PROC SQL ;
  DESCRIBE TABLE
    DICTIONARY.OPTIONS ;
QUIT ;
```

SAS Log Results

```
create table DICTIONARY.OPTIONS
  (
   optname char(32) label='Option Name',
   setting char(1024) label='Option Setting',
   optdesc char(160) label='Option Description',
   level char(8) label='Option Location'
  );
```

Note: The information contained in dictionary tables is also available to DATA and PROC steps outside the SQL procedure. Referred to as SASHELP views, each view is prefaced with the letter “V” and may be shortened with abbreviated names. SASHELP views can be accessed by referencing the view by its name in the SASHELP library. Please refer to the SAS Procedures Guide for further details on accessing and using dictionary views in the SASHELP library.

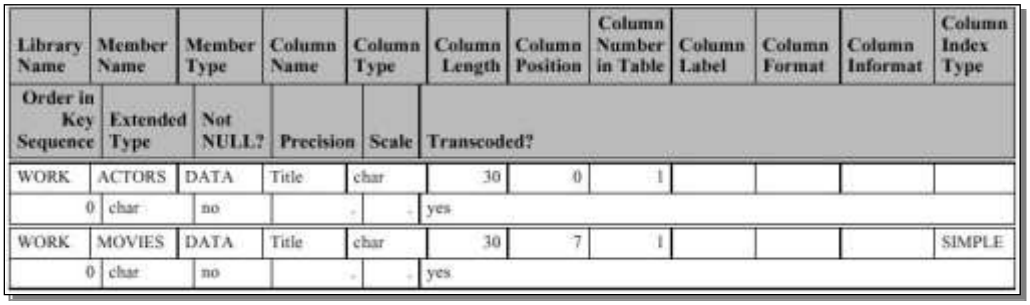
The COLUMNS DICTIONARY Table and VCOLUMN SASHELP View

Retrieving information about the columns in one or more data sets or tables is easy with the COLUMNS dictionary table. Similar to the results of the CONTENTS procedure, users are able to capture column-level information including column name, type, length, position, label, format, informat, and indexes, as well as produce cross-reference listings containing the location of columns in a SAS library. For example, the following code requests a cross-reference listing of the tables containing the TITLE column in the WORK library. **Note:** Care should be used when specifying multiple functions on the WHERE clause since the SQL Optimizer is unable to optimize the query resulting in all allocated SAS session librefs being searched. This can cause the query to run much longer than expected.

PROC SQL Code

```
PROC SQL ;
  SELECT *
    FROM DICTIONARY.COLUMNS
      WHERE UPCASE(LIBNAME)=”WORK” AND
            UPCASE(NAME)=”TITLE” ;
QUIT ;
```

Results



Library Name	Member Name	Member Type	Column Name	Column Type	Column Length	Column Position	Column Number in Table	Column Label	Column Format	Column Informat	Column Index Type
Order in Key Sequence	**Extended Type**	**Not NULL?**	**Precision**	**Scale**	**Transcoded?**						
WORK	ACTORS	DATA	Title	char	30	0	1				
0	char	no	.	.	yes						
WORK	MOVIES	DATA	Title	char	30	7	1				SIMPLE
0	char	no	.	.	yes						

The TABLES DICTIONARY Table and VTABLE SASHELP View

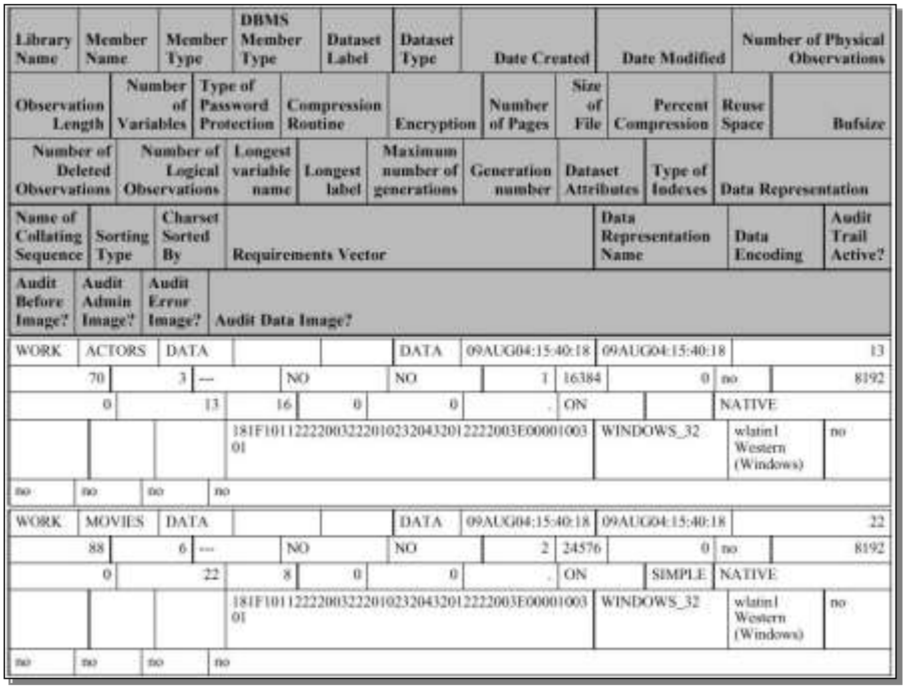
When users need more information about SAS files consider using the TABLES Dictionary table or the VTABLE SASHELP view. The TABLES dictionary table provides detailed information about the library name, member name and type, date created and last modified, number of observations, observation length, number of variables, password protection, compression, encryption, number of pages, reuse space, buffer size, number of deleted observations, type of indexes, and requirements vector. For example, to obtain a detailed list of files in the WORK library, a PROC SQL SELECT query can be constructed as follows.

Note: Because the TABLE Dictionary table produces a considerable amount of information, users should consider specifying a WHERE clause when accessing this table.

<u>**PROC SQL Code**</u>

```
PROC SQL ;
  SELECT *
    FROM DICTIONARY.TABLES
      WHERE UPCASE(LIBNAME)=”WORK” ;
QUIT ;
```

<u>**Results**</u>



Library Name	Member Name	Member Type	DBMS Member Type	Dataset Label	Dataset Type	Date Created	Date Modified	Number of Physical Observations
WORK	ACTORS	DATA			DATA	09AUG04:15:40:18	09AUG04:15:40:18	13
WORK	MOVIES	DATA			DATA	09AUG04:15:40:18	09AUG04:15:40:18	22

Observation Length	Number of Variables	Type of Password Protection	Compression Routine	Encryption	Number of Pages	Size of File	Percent Compression	Reuse Space	Bufsize
70	3	---	NO	NO	1	16384	0	no	8192
88	6	---	NO	NO	2	24576	0	no	8192

Number of Deleted Observations	Number of Logical Observations	Longest variable name	Longest label	Maximum number of generations	Generation number	Dataset Attributes	Type of Indexes	Data Representation
0	13	16	0	0	.	ON		NATIVE
0	22	8	0	0	.	ON	SIMPLE	NATIVE

Name of Collating Sequence	Sorting Type	Charset Sorted By	Requirements Vector	Data Representation Name	Data Encoding	Audit Trail Active?
			181F1011222200322201023204320122222003E00001003 01	WINDOWS_32	wlatin1 Western (Windows)	no
			181F1011222200322201023204320122222003E00001003 01	WINDOWS_32	wlatin1 Western (Windows)	no

Audit Before Image?	Audit Admin Image?	Audit Error Image?	Audit Data Image?
no	no	no	no
no	no	no	no

Accessing Information from SAS DICTIONARY Tables to Do Cool Things

SAS users can quickly and conveniently obtain useful information about their SAS session with a number of read-only SAS system tables called DICTIONARY tables. At any time during a SAS session, DICTIONARY tables can be accessed using the libref DICTIONARY in the FROM clause of a PROC SQL SELECT statement to capture information related to currently defined libnames, table names, column names and attributes, formats, and much more. SASHELP views can be accessed using any of your favorite procedures or in the DATA step. SAS 9.1 software supported 22 Dictionary tables and SASHELP views, SAS 9.2 supported 29 Dictionary tables and SASHELP views, SAS 9.3 supported 30 DICTIONARY tables and SASHELP views, and SAS 9.4 supports 32 DICTIONARY tables and SASHELP views.

Accessing and Displaying the Number of Rows in a Table

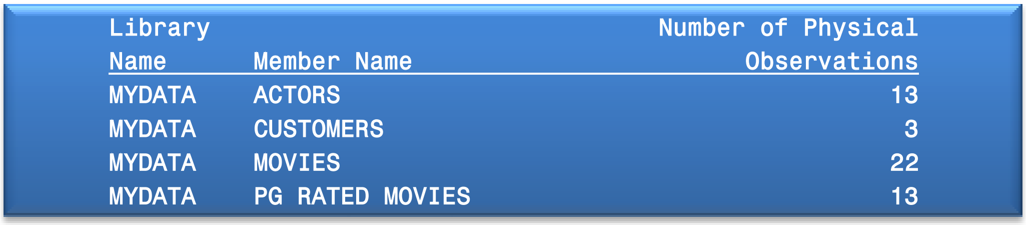
The DICTIONARY table, TABLES, can be accessed to capture and display each table name and the number of observations in the user-assigned MYDATA libref. The following PROC SQL code provides a handy way to quickly determine the number of rows in one or all tables in a libref without having to execute multiple PROC CONTENTS by using the stored information in the Dictionary table TABLES.

<u>**PROC SQL Code**</u>

```
PROC SQL ;
  SELECT LIBNAME, MEMNAME, NOBS
    FROM DICTIONARY.TABLES
```

```
        WHERE UPCASE(LIBNAME)="MYDATA" AND
              UPCASE(MEMTYPE)="DATA" ;
  QUIT ;
```

Results



```
 Library                                  Number of Physical
 Name      Member Name                          Observations
 MYDATA    ACTORS                                         13
 MYDATA    CUSTOMERS                                       3
 MYDATA    MOVIES                                         22
 MYDATA    PG RATED MOVIES                                13
```

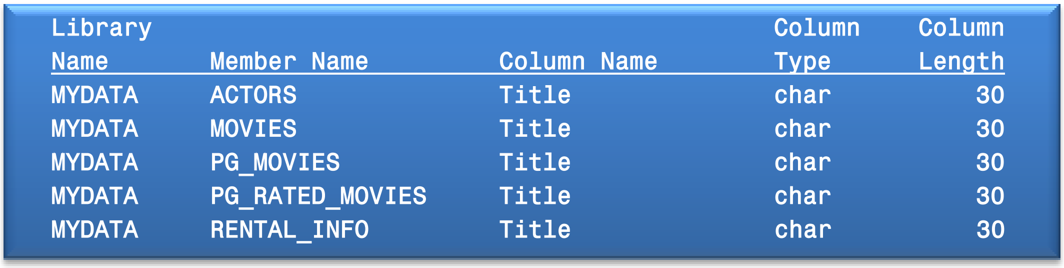
Accessing and Displaying the Column Definitions for a "Key" Variable (or Variables) in All Tables

The DICTIONARY table, COLUMNS, is accessed to display all table names (data sets) that contain the variable TITLE in the user-assigned MYDATA libref as a cross-reference listing. To retrieve the needed type of information, you could execute multiple PROC CONTENTS against selected tables. Or in a more efficient method, you could retrieve the information directly from the read-only Dictionary table COLUMNS with the selected columns LIBNAME, MEMNAME, NAME, TYPE and LENGTH, as shown. For more information about Dictionary tables, readers may want to view the "free" SAS Press Webinar by Kirk Paul Lafler at http://support.sas.com/publishing/bbu/webinar.html#lafler2 or the published paper by Kirk Paul Lafler, Exploring Dictionary Tables and SASHELP Views.

PROC SQL Code

```
  PROC SQL ;
    SELECT LIBNAME, MEMNAME, NAME, TYPE, LENGTH
      FROM DICTIONARY.COLUMNS
        WHERE UPCASE(LIBNAME)="MYDATA" AND
           UPCASE(NAME)="TITLE" AND
           UPCASE(MEMTYPE)="DATA" ;
  QUIT ;
```

Results



```
 Library                                            Column      Column
 Name        Member Name          Column Name       Type        Length
 MYDATA      ACTORS               Title             char            30
 MYDATA      MOVIES               Title             char            30
 MYDATA      PG_MOVIES            Title             char            30
 MYDATA      PG_RATED_MOVIES      Title             char            30
 MYDATA      RENTAL_INFO          Title             char            30
```

Capturing a List of Variables from the COLUMNS Dictionary Table

The DICTIONARY table, COLUMNS, can be accessed to capture and display each column name contained in one or more tables in the WORK libref. The following PROC SQL code provides a handy way to quickly capture the names of any, and all, columns contained in the MOVIES table without having to execute PROC CONTENTS.

PROC SQL Code

```
  PROC SQL NOPRINT ;
    SELECT NAME,
           COUNT(NAME)
           INTO :MVARIABLES SEPARATED BY ' ',
                :MVARIABLESNUM
     FROM DICTIONARY.COLUMNS
```

```
        WHERE UPCASE(LIBNAME)="WORK"
          AND UPCASE(MEMNAME)="MOVIES" ;
    QUIT ;
    %PUT &MVARIABLES &MVARIABLESNUM ;
```

SAS Log Results

```
    %PUT &MVARIABLES &MVARIABLESNUM ;
    Title Length Category Year Studio Rating        6
```

The previous example can be expanded so only the character-defined variables are saved in the macro variable. The following PROC SQL code captures the names of the character-defined columns contained in the MOVIES table and the contents of the macro variable is then specified in a SELECT statement to produce a report.

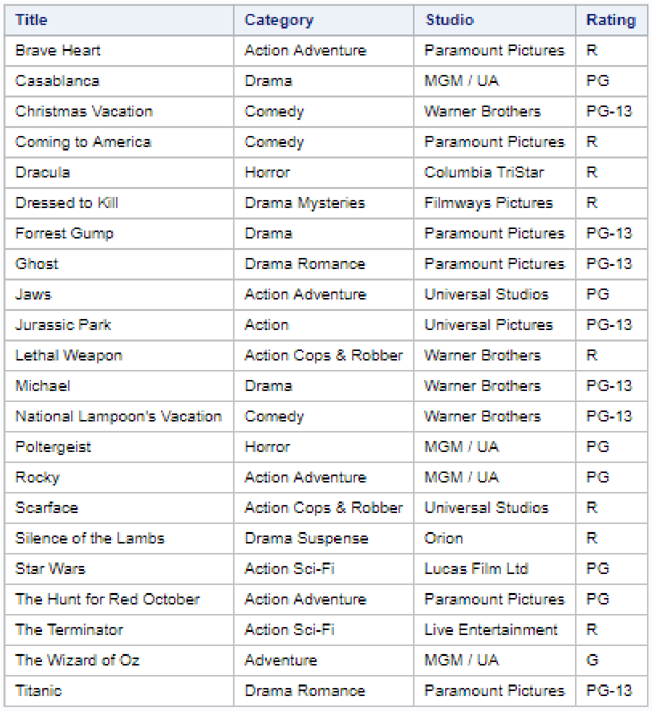
PROC SQL Code

```
    PROC SQL NOPRINT ;
      SELECT NAME
             INTO :MVARIABLES SEPARATED BY ', '
       FROM DICTIONARY.COLUMNS
        WHERE UPCASE(LIBNAME)="WORK"
          AND UPCASE(MEMNAME)="MOVIES"
          AND UPCASE(TYPE)="CHAR" ;
      %PUT &MVARIABLES ;
      RESET PRINT ;
      SELECT &MVARIABLES FROM MOVIES ;
    QUIT ;
```

SAS Log Results

```
    %PUT &MVARIABLES ;
    Title Category Studio Rating
```

PROC PRINT Results



Title	Category	Studio	Rating
Brave Heart	Action Adventure	Paramount Pictures	R
Casablanca	Drama	MGM / UA	PG
Christmas Vacation	Comedy	Warner Brothers	PG-13
Coming to America	Comedy	Paramount Pictures	R
Dracula	Horror	Columbia TriStar	R
Dressed to Kill	Drama Mysteries	Filmways Pictures	R
Forrest Gump	Drama	Paramount Pictures	PG-13
Ghost	Drama Romance	Paramount Pictures	PG-13
Jaws	Action Adventure	Universal Studios	PG
Jurassic Park	Action	Universal Pictures	PG-13
Lethal Weapon	Action Cops & Robber	Warner Brothers	R
Michael	Drama	Warner Brothers	PG-13
National Lampoon's Vacation	Comedy	Warner Brothers	PG-13
Poltergeist	Horror	MGM / UA	PG
Rocky	Action Adventure	MGM / UA	PG
Scarface	Action Cops & Robber	Universal Studios	R
Silence of the Lambs	Drama Suspense	Orion	R
Star Wars	Action Sci-Fi	Lucas Film Ltd	PG
The Hunt for Red October	Action Adventure	Paramount Pictures	PG
The Terminator	Action Sci-Fi	Live Entertainment	R
The Wizard of Oz	Adventure	MGM / UA	G
Titanic	Drama Romance	Paramount Pictures	PG-13

Conclusion

Unlike procedural programming languages where a program's flow of execution is described using a detailed step-by-step logical approach to solving a problem or with object-oriented programming where an object is told how to behave without all the detailed steps that informs the object how to behave. Data-driven programming involves a program that has its decisions and processes (the flow of execution) controlled (or dictated) by the data (or data structures).

The SAS System's read-only Dictionary tables and corresponding SASHELP views provide valuable information about SAS libraries, data sets, columns and attributes, catalogs, indexes, macros, system options, titles, views, and much more. Users are encouraged to research these powerful resources of information to better understand information about data, for the creation of system documentation and performance tuning, as well as other important application areas.

References

Abolafia, Jeff and Frank DiIorio (2008), "Building Intelligent Macros: Using Metadata Functions with the SAS® Macro Language," Proceedings of the 2008 SAS Global Forum (SGF) Conference.

Batkhan, Leonid, 2016, "Modifying variable attributes in all datasets of a SAS library", a SAS Blog Post, http://blogs.sas.com/content/sgf/2016/11/25/modifying-variable-attributes-in-all-datasets-of-a-sas-library/.

Carpenter, Arthur L. (2017), "Building Intelligent Macros: Using Metadata Functions with the SAS® Macro Language," 2017 SAS Global Forum (SGF) Conference, California Occidental Consultants, Anchorage, AK, USA.

Davis, Michael (2001), "*You Could Look It Up: An Introduction to SASHELP Dictionary Views*," Proceedings of the 2001 SAS Users Group International (SUGI) Conference, Bassett Consulting Services, North Haven, CT, USA.

Hamilton, Jack (1998), "*Some Utility Applications of the Dictionary Tables in PROC SQL*," Proceedings of the 1998 Western Users of SAS Software (WUSS) Conference, 85-90.

Lafler, Kirk Paul (2016), "*Valuable Things You Can Do with SAS DICTIONARY Tables and SASHELP Views*," Wisconsin Illinois SAS Users (WIILSU) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2013). *PROC SQL: Beyond the Basics Using SAS, Second Edition*, SAS Institute Inc., Cary, NC, USA.

Lafler, Kirk Paul (2012), "*Exploring DICTIONARY Tables and SASHELP Views*," South Central SAS Users Group (SCSUG) Conference and Kansas City SAS Users Group (KCSUG) Meeting, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), "*DATA Step versus PROC SQL Programming Techniques*," 2009 South East SAS Users Group (SESUG) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2009), "*Exploring DICTIONARY Tables and SASHELP Views*," 2009 Western Users of SAS Software (WUSS) Conference and 2009 Pharmaceutical SAS Users Group (PharmaSUG) Conference, Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2008), "*Undocumented and Hard-to-find PROC SQL Features*," Greater Atlanta SAS Users Group (GASUG) Meeting (June 11th, 2008); Pharmaceutical SAS Users Group (PharmaSUG) Conference (June 1st - 4th, 2008); 2008 Michigan SAS Users Group (MSUG) Meeting (May 29th, 2008); 2008 Vancouver SAS Users Group Meeting (April 23rd, 2008); and 2008 PhilaSUG User Group Meeting (March 13th, 2008); Software Intelligence Corporation, Spring Valley, CA, USA.

Lafler, Kirk Paul (2006), "*Exploring Dictionary Tables with PROC SQL*," SAS Press Webinar Series – June 27, 2006.

Lafler, Kirk Paul (2005), "*Exploring Dictionary Tables and SASHELP Views*," Proceedings of the Thirteenth Annual Western Users of SAS Software Conference.

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

Varney, Brian (2000). "How to Think Through the SAS® DATA Step," Proceedings of the 2000 SAS Users Group International (SUGI) Conference.

Wang, Hui (2015). "*Creating Data-Driven SAS® Code with CALL EXECUTE*," Proceedings of the 2015 PharmaSUG Conference.

Whitlock, Ian (2006). "How to Think Through the SAS® DATA Step," Proceedings of the 2006 SAS Users Group International (SUGI) Conference.

Whitlock, Ian (1998). "*CALL EXECUTE: How and Why*," Proceedings of the 1998 SAS Users Group International (SUGI) Conference.

Trademark Citations

About The Author

Kirk Paul Lafler is an entrepreneur, consultant and founder at Software Intelligence Corporation, and has been using SAS since 1979. Kirk is a SAS application developer, programmer, certified professional, provider of SAS consulting and application development services, mentor, advisor and adjunct professor at University of California San Diego Extension, emeritus sasCommunity.org Advisory Board member, and educator to SAS users around the world. As the author of six books including Google® Search Complete (Odyssey Press. 2014) and PROC SQL: Beyond the Basics Using SAS, Second Edition (SAS Press. 2013); 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:

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