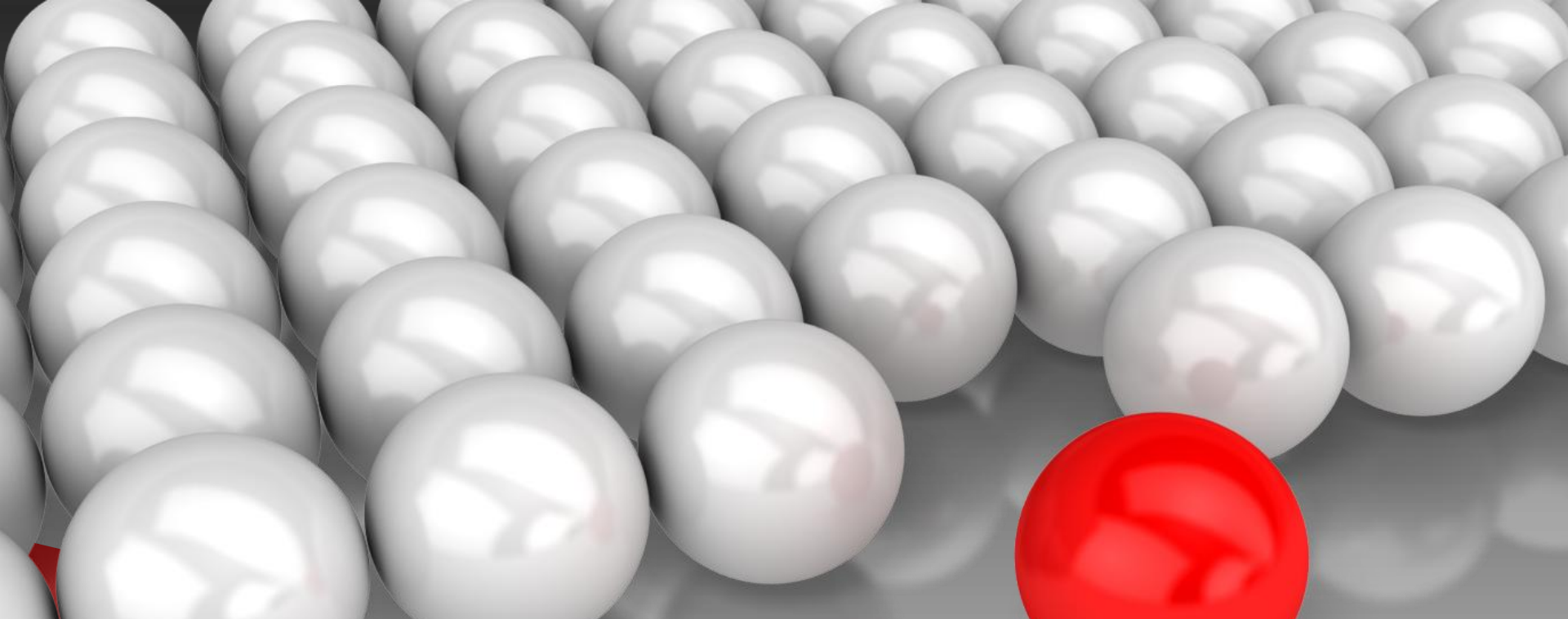
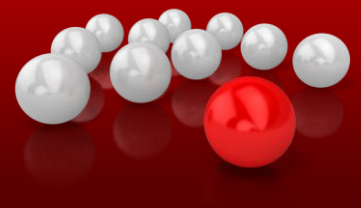


SAS® Programming Tips, Tricks, and Techniques

A presentation by

Kirk Paul Lafler

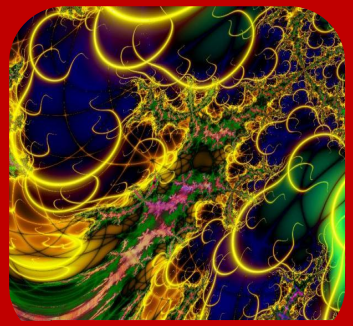


Presentation Objectives - Explore

Useful SAS System Options

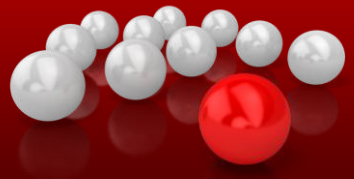
Interesting PROC SQL Options

A User-defined Function using PROC FCMP



Integrity Constraints for Tables

A User-defined Dictionary Table and SASHELP View Tool



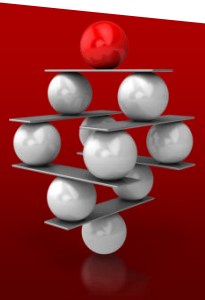
Example Datasets / Tables

	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

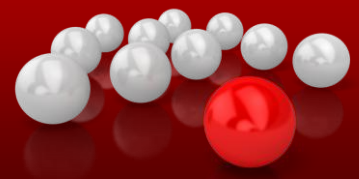
Movies

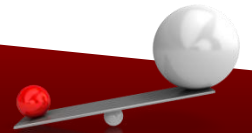
Actors

	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® System Options





SOURCE versus SOURCE2

Primary Source Statements:

```
OPTIONS  SOURCE2;
%INCLUDE ‘c:\Workshops\LogControlOptions.sas’;
PROC  PRINT  DATA=MOVIES  NOOBS;
  VAR  TITLE  RATING  CATEGORY;
RUN;
```

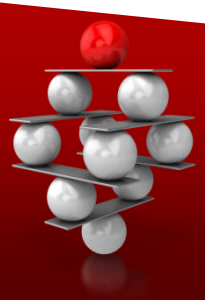
Secondary Source Statements (Included Code):

```
OPTIONS MSGLEVEL=I ;
/* DISPLAY SORT, MERGE PROCESSING, AND INDEX USAGE */;
```

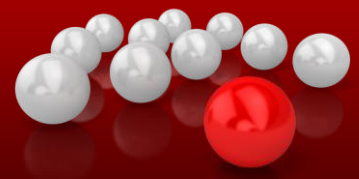
SOURCE versus SOURCE2 Log Results

SAS Log Results:

```
OPTIONS SOURCE2;
%INCLUDE 'c:\Workshops\LogControlOptions.sas';
NOTE: %INCLUDE (level 1) file
c:\Workshops\LogControlOptions.sas is file
c:\Workshops\LogControlOptions.sas.
+OPTIONS MSGLEVEL=I
+/* DISPLAY SORT, MERGE PROCESSING, AND INDEX USAGE */
+;
NOTE: %INCLUDE (level 1) ending.
PROC PRINT DATA=MOVIES NOOBS;
  VAR TITLE RATING CATEGORY;
RUN;
```

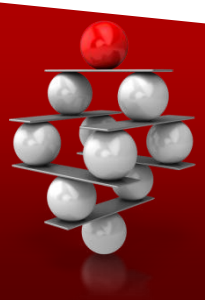


Exploring PROC SQL Options



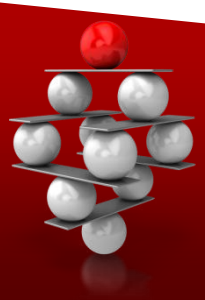
SQL Join Algorithms

- Nested Loop (aka “Brute” force) join algorithm
- Sort-merge join algorithm
- Index join algorithm
- Hash join algorithm



Influencing the SQL Optimizer

Option	Description
MAGIC=101	Influences the SQL optimizer to select the Nested Loop join algorithm.
MAGIC=102	Influences the SQL optimizer to select the Sort-Merge join algorithm.
MAGIC=103	Influences the SQL optimizer to select the Hash join algorithm.



Specifying MAGIC=101

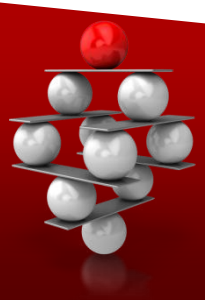
```
PROC SQL MAGIC=101;
  SELECT *
    FROM MOVIES, ACTORS
      WHERE MOVIES.TITLE  = ACTORS.TITLE;
QUIT;
```

SAS Log Results

```
PROC  SQL  MAGIC=101;
 SELECT *
   FROM  MOVIES,  ACTORS
     WHERE MOVIES.TITLE  = ACTORS.TITLE;
```

NOTE: PROC SQL planner chooses sequential loop join.

```
QUIT;
```



Specifying MAGIC=102

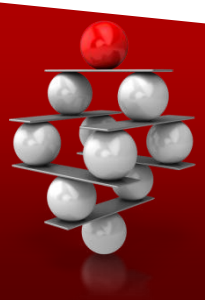
```
PROC SQL MAGIC=102;
  SELECT *
    FROM MOVIES, ACTORS
      WHERE MOVIES.TITLE  = ACTORS.TITLE;
QUIT;
```

SAS Log Results

```
PROC  SQL  MAGIC=102;
 SELECT *
   FROM  MOVIES,  ACTORS
     WHERE MOVIES.TITLE  = ACTORS.TITLE;
```

NOTE: PROC SQL planner chooses merge join.

```
QUIT;
```



Specifying MAGIC=103

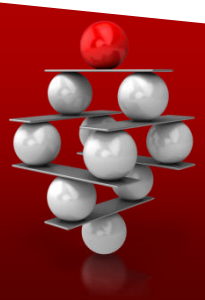
```
PROC SQL MAGIC=103;
  SELECT *
    FROM MOVIES, ACTORS
      WHERE MOVIES.TITLE  = ACTORS.TITLE;
QUIT;
```

SAS Log Results

```
PROC  SQL  MAGIC=103;
 SELECT *
   FROM  MOVIES,  ACTORS
     WHERE MOVIES.TITLE  = ACTORS.TITLE;
```

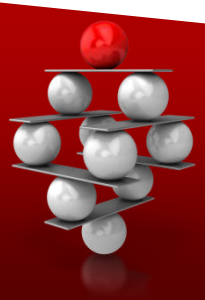
NOTE: PROC SQL planner chooses merge join.
NOTE: A merge join has been transformed to a hash join.

```
QUIT;
```



Specifying IDXWHERE=Yes

The IDXWHERE= dataset option can be specified to influence the SQL optimizer to use the most efficient index available (if one exists) to execute a query.



Specifying IDXWHERE=Yes

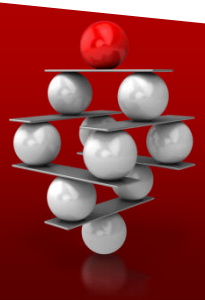
```
OPTIONS MSGLEVEL=I;
PROC SQL;
  SELECT *
    FROM MOVIES(IDXWHERE=Yes), ACTORS
      WHERE MOVIES.TITLE  =  ACTORS.TITLE;
QUIT;
```

SAS Log Results

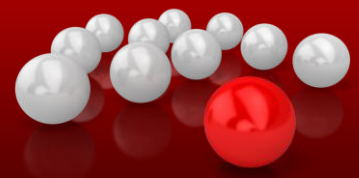
```
PROC  SQL;
 SELECT *
   FROM  MOVIES(IDXWHERE=Yes),  ACTORS
     WHERE MOVIES.TITLE  = ACTORS.TITLE;
```

INFO: Index Rating selected for WHERE clause optimization.

```
QUIT;
```



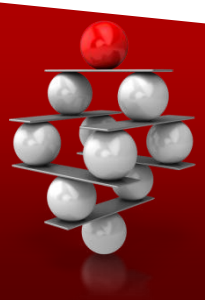
Exploring a User-defined Function using PROC FCMP



PROC FCMP Advantages

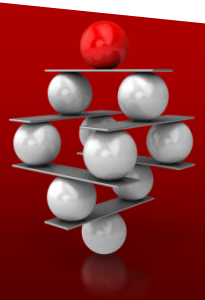
User-Defined functions created with the FCMP procedure provide specific advantages:

- ✓ **Code can be easier to read, write and modify**
- ✓ **Code is modular and callable**
- ✓ **Code is independent and not affected by its implementation**
- ✓ **Code is reusable by any program that has access to the dataset where the function is stored**



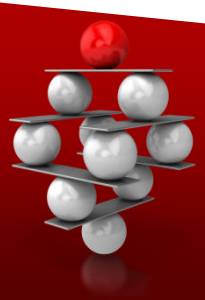
User-defined Function (Part 1)

```
/* Constructing a User Defined Function in FCMP */
proc fcmp outlib=sasuser.myfunctions.examples ;
  function Age_of_Movie_function(year) ;
    if year NE . then do ;
      Age_of_Movie = year(today()) – year ;
    end ;
    return(Age_of_Movie);
  endsub ;
quit ;
```



User-defined Function (Part 2)

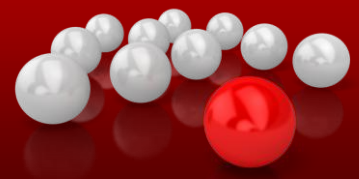
```
options cmplib=sasuser.myfunctions ;

/* Call the User Defined Function created in FCMP */
data Age_of_Movie ;
  set mydata.movies ;
  Age_of_Movie = Age_of_Movie_function(year) ;
  put Year= Age_of_Movie= ;
run ;
```

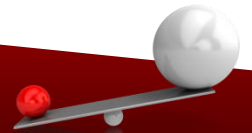


User-defined Function (Log Results)

```
928   /* Constructing a User Defines Function (UDF) in FCMP */
929   proc fcmp outlib=sasuser.myfunctions.examples;
930     function Age_of_Movie_function(year);
931       if year NE . then do;
932          Age_of_Movie = year(today()) - year;
933       end;
934       return(Age_of_Movie);
935     endsub;
936   quit;

NOTE: Function Age_of_Movie_function saved to sasuser.myfunctions.examples.
NOTE: PROCEDURE FCMP used (Total process time):
      real time               0.15 seconds
      cpu time                0.04 seconds


937   options cmplib=sasuser.myfunctions;
938   /* Call the User Defined Function (UDF) created in FCMP */
939   data Age_of_Movie;
940     set mydata.movies;
941     Age_of_Movie = Age_of_Movie_function(year);
942     put Year= Age_of_Movie=;
943   run;

Year=1995 Age_of_Movie=17
Year=1942 Age_of_Movie=70
Year=1989 Age_of_Movie=23
Year=1988 Age_of_Movie=24
Year=1993 Age_of_Movie=19
Year=1980 Age_of_Movie=32
Year=1994 Age_of_Movie=18
Year=1990 Age_of_Movie=22
Year=1975 Age_of_Movie=37
Year=1993 Age_of_Movie=19
Year=1987 Age_of_Movie=25
Year=1997 Age_of_Movie=15
Year=1983 Age_of_Movie=29
Year=1982 Age_of_Movie=30
Year=1976 Age_of_Movie=36
Year=1983 Age_of_Movie=29
Year=1991 Age_of_Movie=21
Year=1977 Age_of_Movie=35
Year=1989 Age_of_Movie=23
Year=1984 Age_of_Movie=28
Year=1939 Age_of_Movie=73
Year=1997 Age_of_Movie=15
NOTE: There were 22 observations read from the data set MYDATA.MOVIES.
NOTE: The data set WORK.AGE_OF_MOVIE has 22 observations and 7 variables.
NOTE: DATA statement used (Total process time):
      real time               0.07 seconds
      cpu time                0.06 seconds
```

Exploring Integrity Constraints for Tables

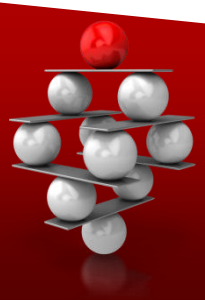




Methods of Building Integrity

Data integrity problems such as missing information, duplicate values, and invalid data values can affect user confidence in a database environment. The objective is to establish rules in the database table(s) to safeguard and protect data.

- **Application programs – Older / less reliable**
- **Database table environment – Newer / more reliable**
  - ✓ **PROC DATASETS**
  - ✓ **PROC SQL**

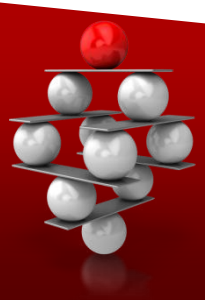


Column and Table Constraints

Data integrity is maintained by assigning column and table constraints. Modifications made through update and delete operations can have referential integrity constraints built into the database environment.

Column and Table Constraints

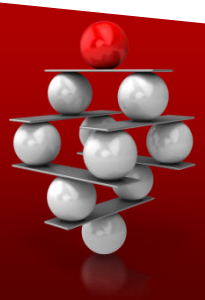
- ✓ NOT NULL
- ✓ UNIQUE
- ✓ CHECK



NOT NULL Constraint

To prevent null values from appearing in any row of a table for a specified column, a NOT NULL constraint can be coded.

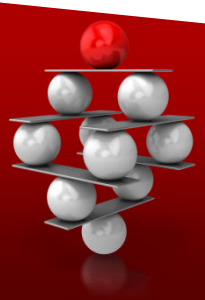
```
PROC SQL;
  CREATE TABLE work.RENTAL_INFO
    (TITLE CHAR(30) NOT NULL,
      RENTAL_AMT NUM FORMAT=DOLLAR6.2);
QUIT;
```



UNIQUE Constraint

The UNIQUE constraint prevents rows containing duplicate values for a specified column from being added to a table.

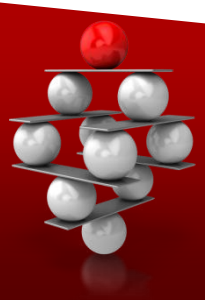
```
PROC SQL;
  CREATE TABLE work.RENTAL_INFO
    (TITLE CHAR(30) UNIQUE,
      RENTAL_AMT NUM FORMAT=DOLLAR6.2);
QUIT;
```



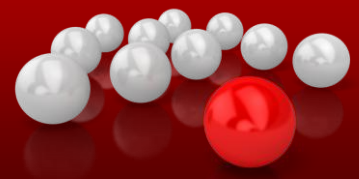
CHECK Constraint

A CHECK constraint can be specified to assign specific rules that a column must adhere to.

```
PROC SQL;
  ALTER TABLE  MOVIES
    ADD  CONSTRAINT  CHECK_RATING
      CHECK (RATING IN ('G', 'PG', 'PG-13', 'R'));
QUIT;
```

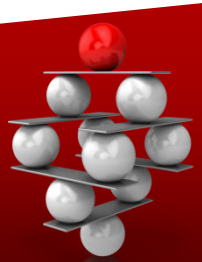


Exploring a User-defined Tool using Dictionary Tables



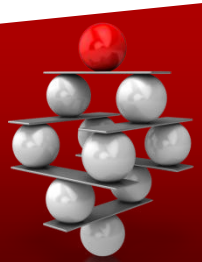
Dictionary Tables / SASHELP Views

- SAS collects information about a session
- Session information is captured as read-only content
- Tables are accessible using PROC SQL
  - ✓ Specify table in FROM clause of a SELECT
  - ✓ DICTIONARY libref is automatically assigned
- SASHELP Views are accessed in a DATA step or with any of your favorite PROCs



Viewing Dictionary Tables / Views

- # of DICTIONARY Tables/Views:
  - ✓ 22 in SAS 9.1
  - ✓ 29 in SAS 9.2
  - ✓ 30 in SAS 9.3

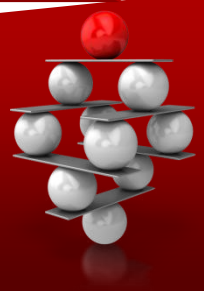


Cross-reference Listing – PROC PRINT

```
PROC PRINT DATA=SASHELP.VCOLUMN NOOBS;
   VAR LIBNAME MEMNAME NAME TYPE LENGTH;
    WHERE UPCASE(LIBNAME)=UPCASE("SASUSER") AND
          UPCASE(NAME)=UPCASE("TITLE");
RUN;
```

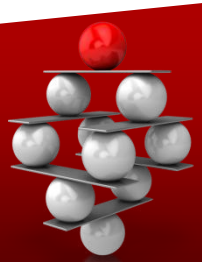
Cross-reference listing for the column TITLE

Library Name	Member Name	Column Name	Column Type	Column Length
SASUSER	ACTORS	Title	char	30
SASUSER	MOVIES	Title	char	30



Cross-reference Listing – PROC PRINT

```
%MACRO CROSSREF(LIB, COLNAME);
  PROC  PRINT  DATA=SASHELP.VCOLUMN  NOOBS;
    VAR LIBNAME  MEMNAME  NAME  TYPE  LENGTH;
     WHERE UPCASE(LIBNAME)=UPCASE("&LIB") AND
                UPCASE(NAME)=UPCASE("&COLNAME");
  RUN;
%MEND CROSSREF;
%CROSSREF(SASUSER,TITLE);
```

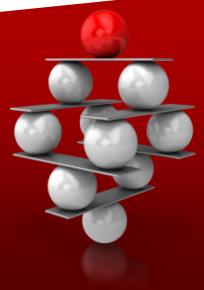


Cross-reference Listing – PROC PRINT

```
PROC PRINT DATA=SASHELP.VCOLUMN NOOBS;
  VAR LIBNAME MEMNAME NAME TYPE LENGTH;
    WHERE UPCASE(LIBNAME)="SASUSER"  AND
          UPCASE(NAME)="TITLE";
RUN;
```

Cross-reference listing for the column TITLE

Library Name	Member Name	Column Name	Column Type	Column Length
SASUSER	ACTORS	Title	char	30
SASUSER	MOVIES	Title	char	30



Conclusion

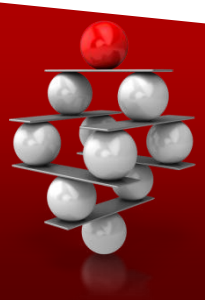

Useful SAS System Options

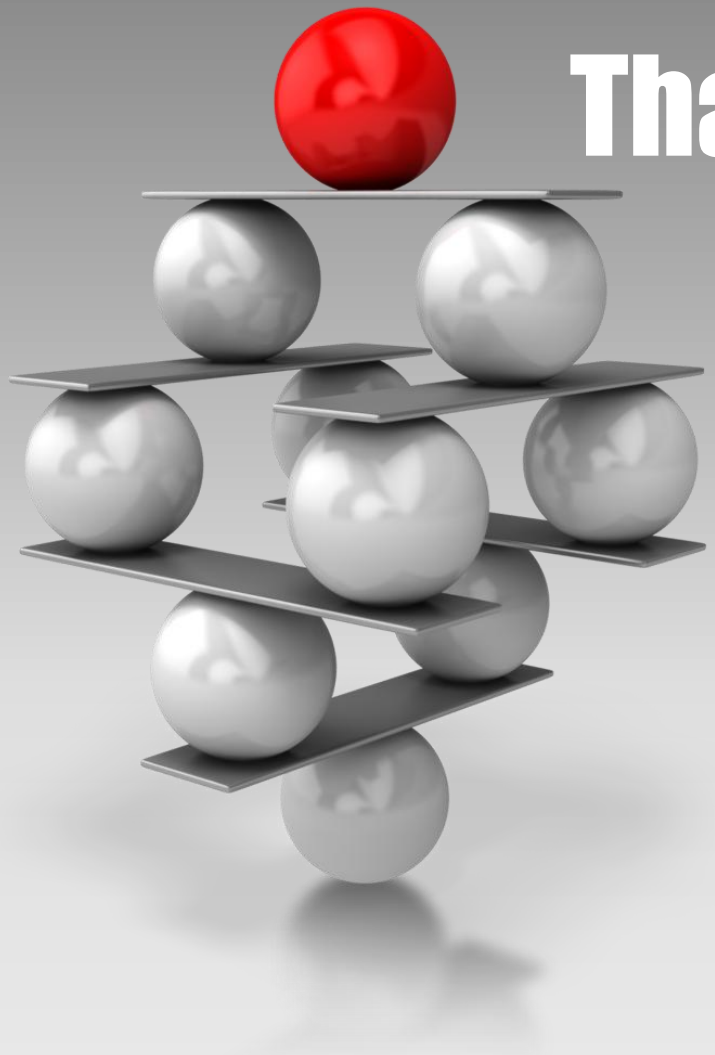

User-defined Tool using Dictionary Tables


PROC SQL Options


Integrity Constraints for Tables


User-defined Function using PROC FCMP





Thank You for Attending!

Questions?

A presentation by

Kirk Paul Lafler
KirkLafler@cs.com
@sasNerd