Webinar@Lunchtime

SQL und Makro – Passt das zusammen?

Herzlich Willkommen bei Webinar@Lunchtime

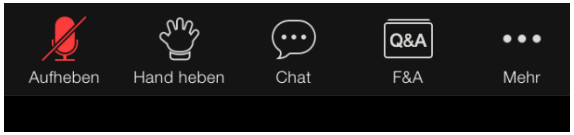
Moderation
Claudia Hohn

SAS Institute GmbH
Education Services Advisor

Training
Eva-Maria Kegelmann

SAS Institute GmbH
Sr Technical Training Consultant

Hinweise zum Ablauf des Webinars:



- Teilnehmer sind automatisch “stumm” geschaltet
- Sie können die Hand heben oder eine Frage stellen
- Die Veranstaltung wird aufgezeichnet und mit den Unterlagen auf www.sas.de/lunchtime zur Verfügung gestellt

SQL und Makro – Passt das zusammen?

SQL und Makrovariablen

SQL in Makroprogrammen

SQL und Makro – Passt das zusammen?

SQL und Makrovariablen

SQL in Makroprogrammen

SQL und Makrovariablen

- Makrovariablen mit PROC SQL erstellen
- Makrovariablen referenzieren mit der Makrofunktion %TSLIT
- Automatische Makrovariablen bei PROC SQL

Makrovariablen mit PROC SQL und der INTO-Klausel befüllen

- Allgemeine Form:

```
SELECT col1, col2, . . . INTO :mvar1, :mvar2,...
    FROM table-expression
    WHERE where-expression
    other clauses;
```

- Der Umgang mit führenden und nachfolgenden Leerzeichen ist unterschiedlich: Entfernen oder nicht?
- Formatierung der Werte möglich
  Ohne Formatangabe: Verwendung von BEST8. bei numerischen Werten.

Makrovariablen mit PROC SQL erstellen:

Einen Wert in eine Makrovariable schreiben

- Ein berechneter Wert wird in eine Makrovariable geschrieben
- Führende und nachfolgende Leerzeichen werden hierbei nicht entfernt.
- Formatierung des Wertes möglich

```
proc sql noprint;
    select avg(age) format=commax6.1
        into :durchschnittsalter
        from sashelp.class;
quit;
```

```
28      %put &=durchschnittsalter;
DURCHSCHNITTSALTER= 13,3
```

Option **TRIMMED**

Entfernen führender und nachfolgender Leerzeichen

- Ohne TRIMMED:

```
proc sql noprint;
    select avg(age) format=commax6.1
        into :durchschnittsalter
        from sashelp.class;
quit;
```

```
28      %put &=durchschnittsalter;
DURCHSCHNITTSALTER= 13,3
```

- Mit TRIMMED:

```
proc sql noprint;
    select avg(age) format=commax6.1
        into :durchschnittsalter trimmed
        from sashelp.class;
quit;
```

```
28      %put &=durchschnittsalter;
DURCHSCHNITTSALTER=13,3
```

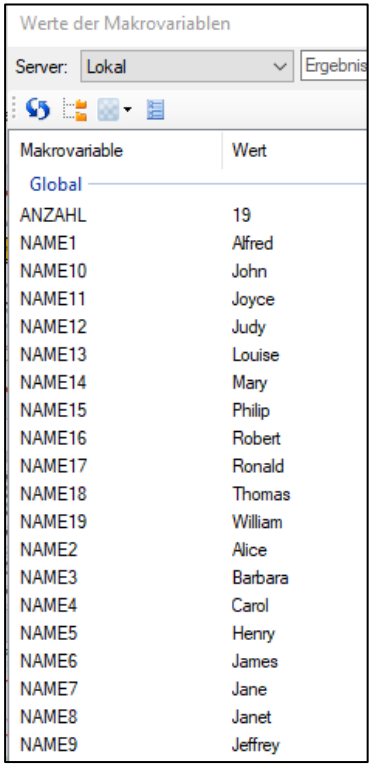
Makrovariablen mit PROC SQL erstellen:

Viele Makrovariablen mit **jeweils einem Wert** erstellen ohne die Anzahl zu wissen

- Führende und nachfolgende Leerzeichen werden hierbei automatisch entfernt
- Obere Grenze darf offen bleiben
- Der Name muss jedoch am Ende eine Zahl haben (hier: name1)

```
proc sql noprint;
  select name
    into :name1 -
      from sashelp.class;
quit;
```

- Die Datei hat 19 Zeilen
  ➔ Ergibt 19 Makrovariablen



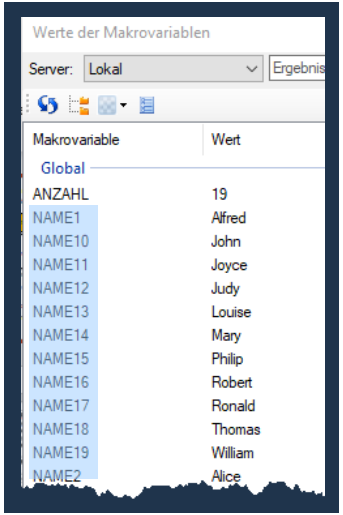
Makrovariablen mit PROC SQL erstellen:

Viele Makrovariablen mit jeweils einem Wert erstellen

–

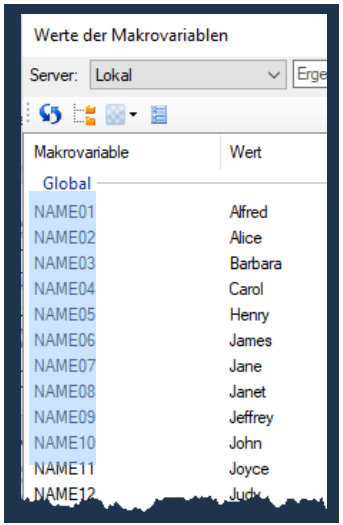
Reihenfolge optimieren

- Reihenfolge der Makrovariablen ist alphabetisch, nicht numerisch!



- Reihenfolge mit einem kleinen Trick optimieren:

```
proc sql noprint;
  select name
    into :name01 -
      from sashelp.class;
quit;
```



Makrovariablen mit PROC SQL erstellen:

Mehrere Werte in **eine Makrovariable** schreiben

- Führende und nachfolgende Leerzeichen werden hierbei automatisch entfernt
- Trennzeichen wählbar:

```
proc sql noprint;
    select distinct origin
       into :Kontinent separated by ', '
          from sashelp.cars;
quit;
```

- Die Datei hat 3 eindeutige Werte für **origin:**

```
28      %put &=kontinent;
KONTINENT=Asia, Europe, USA
```

- Anzahl merken:

```
%let anzahl = &sqlobs;
```

```
28      %put &=anzahl;
ANZAHL=3
```

Makrovariablen referenzieren

- Makrovariablenname mit vorangestelltem &
- Wenn innerhalb von Textstrings:
  Dann doppelte Anführungszeichen (“ “) verwenden, sonst werden Makrovariablen nicht aufgelöst

```
%let geschlecht = F;
proc sql;
   select * from sashelp.class
          where sex = "&geschlecht";
quit;
```

Makrovariablen referenzieren

Bei (einigen) Datenbanken

- Doppelte "" sind für Variablennamen vorbehalten
- Textstrings müssen in einfache Anführungszeichen
- Dann werden in SAS aber die Makrovariablen nicht aufgelöst!

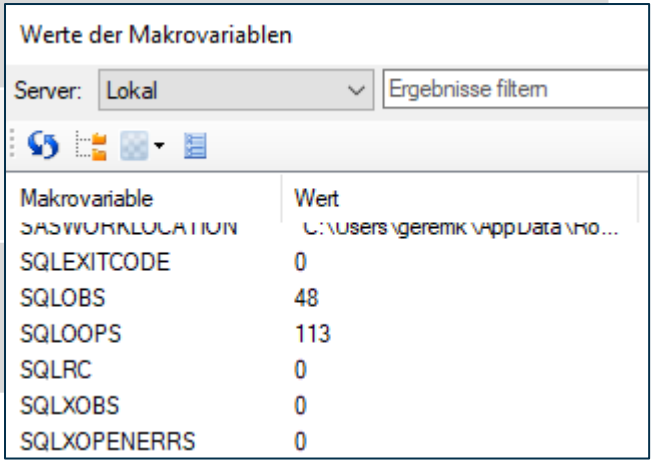
- Lösung: Makrofunktion %TSLIT(text)

```
%let geschlecht = F;
proc sql;
   select * from sashelp.class
      where sex = %tslit (&geschlecht);
quit;
```

- Setzt die Notwendigkeit von doppelten Anführungszeichen um Text außer Kraft und
- setzt einfache Anführungszeichen um den Eingabewert.

Automatische Makrovariablen bei PROC SQL

SQLEXITCODE	Returncode von SQL-Insert-Fehlern. Wird beim Ende von PROC SQL in die Makrovar. SYSERR geschrieben.
SQLOBS	Anzahl zurückgegebener Zeilen bei PROC SQL
SQLOOPS	Anzahl Iterationen, die die innere Schleife von PROC SQL durchläuft
SQLXOOPENERRS	Anzahl der DICTIONARY-Tabellen, die nicht geöffnet werden konnten
SQLRC	Returncode der SQL-Prozedur
SQLXMSG	Bei Pass-Through: DBMS-spezifische Fehlermeldung
SQLXRC	Bei Pass-Through: DBMS-spezifischer Returncode



SQL und Makro – Passt das zusammen?

SQL und Makrovariablen

SQL in Makroprogrammen

SQL in Makroprogrammen - Beispiele

- Zugreifen auf die Dictionary-Dateien ⇨
  Alle benutzerdefinierten Makrovariablen löschen
- Dynamische Views mit PROC SQL und Makrovariablen.
- Lokale und globale Symboltabellen:
  Bei Verwendung von PROC SQL in einem MAKRO
- BY-Gruppenverarbeitung für ein gesamtes Programm

Zugreifen auf die Dictionary-Dateien

- Enthalten Metadaten über Dateien, Bibliotheken, Variablen,… und Makrovariablen.
- Zugriff nur mit PROC SQL und der automatisch verfügbaren Bibliothek DICTIONARY.
- Aber: In SASHELP sind Views dazu verfügbar Gleicher Name mit einem vorangestellten V. Beispiel: SASHELP.VMACROS statt DICTIONARY.MACROS

- Herausfinden, welche Dictionary-Dateien es gibt:

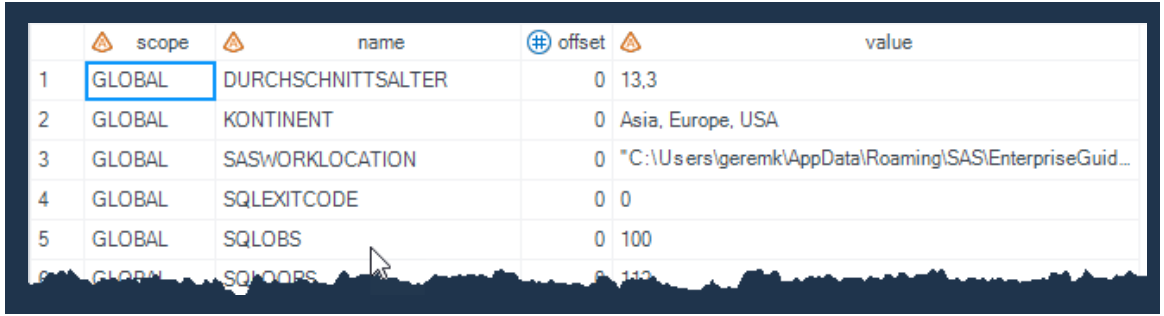
```
proc sql;
   select * from
                dictionary.dictionaries;
quit;
```

Zugreifen auf die Dictionary-Datei:

-

dictionary.macros

- Enthält alle Makrovariablen mit Eigenschaften:



	scope	name	offset	value
1	GLOBAL	DURCHSCHNITTSALTER	0	13,3
2	GLOBAL	KONTINENT	0	Asia, Europe, USA
3	GLOBAL	SASWORKLOCATION	0	"C:\Users\geremk\AppData\Roaming\SAS\EnterpriseGuid...
4	GLOBAL	SQLEXITCODE	0	0
5	GLOBAL	SQLOBS	0	100

- Namen der benutzerdefinierten Makrovariablen per Programm ermitteln:

```
proc sql noprint;
   select name into :alle_mv
                    separated by " "
      from dictionary.macros
         where scope="GLOBAL";
quit;
```

Alle benutzerdefinierten Makrovariablen löschen

- %SYMDEL löscht Makrovariablen.
- Aber: Es gibt kein **%SYMDEL _ALL_;**
- Lösung: Eigenes Makroprogramm

```
%macro deleteAllMacroVars();
   %local mv_to_delete;
   proc sql noprint;
      select name into :mv_to_delete
                       separated by " "
       from dictionary.macros
         where scope="GLOBAL" and
               not substr(name,1,3)="SYS";
   quit;
   %symdel &mv_to_delete.;
%mend deleteAllMacroVars;

%deleteAllMacroVars();
```

Dynamische Views mit PROC SQL und Makrovariablen

- So leider nicht:

```
%let geschlecht=F;
proc sql;
   create view schueler_m_or_f as
      select *
         from sashelp.class
            where sex="&geschlecht";
quit;

%let geschlecht=M;
proc print data=schueler_m_or_f;
run;
```

- View ist statisch auf den Wert der Makrovariable zur Definition des Views eingestellt.

Dynamische Views mit PROC SQL und Makrovariablen

- Jetzt klappt es:

```
proc sql;
   create view schueler_m_or_f as
      select * from sashelp.class
        where sex=symget('geschlecht');
quit;
```

- **Definition des Views:**
  Makrovariable wird noch nicht benötigt.

Dynamische Views mit PROC SQL und Makrovariablen - Ergebnis

- Und das kommt dabei heraus:

```
%let geschlecht=F;
proc print
    data=schueler_m_or_f;
run;
```

Obs	Name	Sex	Age	Height	Weight
1	Alice	F	13	56.5	84.0
2	Barbara	F	13	65.3	98.0
3	Carol	F	14	62.8	102.5
4	Jane	F	12	59.8	84.5
5	Janet	F	15	62.5	112.5
6	Joyce	F	11	51.3	50.5
7	Judy	F	14	64.3	90.0
8	Louise	F	12	56.3	77.0
9	Mary	F	15	66.5	112.0

```
%let geschlecht=M;
proc print
    data=schueler_m_or_f;
run;
```

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Henry	M	14	63.5	102.5
3	James	M	12	57.3	83.0
4	Jeffrey	M	13	62.5	84.0
5	John	M	12	59.0	99.5
6	Philip	M	16	72.0	150.0
7	Robert	M	12	64.8	128.0
8	Ronald	M	15	67.0	133.0
9	Thomas	M	11	57.5	85.0
10	William	M	15	66.5	112.0

- **Ausführung des Views:**
  Wert der Makrovariable zu diesem Zeitpunkt wird für die Selektion verwendet.

Lokale und globale Symboltabellen:

Bei Verwendung von PROC SQL in einem MAKRO

Ist die Makrovariable durchschnittsalter global oder lokal?

```
%macro test ;
   proc sql noprint;
      select avg(age) format=commax6.1
         into :durchschnittsalter
            from sashelp.class;
   quit;
%mend;
%test
```

Antwort: Lokal

Obwohl es keine Übergabeparameter gibt und auch sonst explizit keine lokale Makrovariable angelegt wird: PROC SQL legt selbst im Hintergrund lokale Makro-variablen an!!!!

Lokale und globale Symboltabellen:

Bei Verwendung von PROC SQL in einem MAKRO

Wenn durchschnittsalter global sein soll:

```
%macro test ;
   %global durchschnittsalter;
   proc sql noprint;
      select avg(age) format=commax6.1
         into :durchschnittsalter
            from sashelp.class;
   quit;
%mend;
%test
```

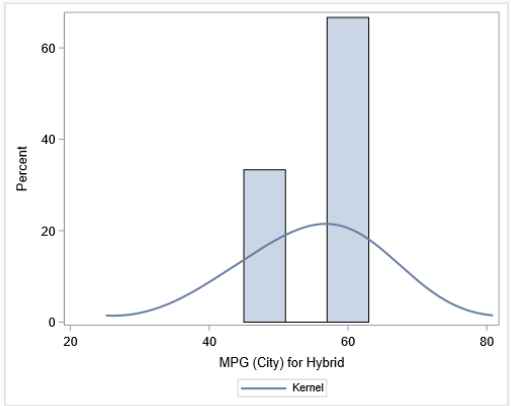
BY-Gruppen-Verarbeitung für ein gesamtes Programm:

Was soll am Ende herauskommen?

Ziel:

Für jede Ausprägung von TYPE sollen Häufigkeiten + Histogramm zusammen angezeigt werden.

Origin	Type	Frequency
Asia	Hybrid	3



Origin	Type	Frequency
Asia	SUV	25
Europe	SUV	10
USA	SUV	25

BY-Gruppen-Verarbeitung für ein gesamtes Programm:

1. Verwendete Prozeduren

```
title "Type by Origin in SASHELP.CARS";

proc freq data=sashelp.cars;
   table origin*type /nocum nopercent
                      list;
run;

proc sgplot data=sashelp.cars;
   histogram mpg_city;
   density mpg_city / type=kernel;
run;
```

BY-Gruppen-Verarbeitung für ein gesamtes Programm:

2. Nur für einen TYPE

```
title "HYBRID by Origin in SASHELP.CARS";

proc freq data=sashelp.cars
                  (where=(type="Hybrid"));
  table origin*type/nocum nopercent list;
run;

proc sgplot data=sashelp.cars
                    (where=(type="Hybrid"));
   histogram mpg_city;
   density mpg_city / type=kernel;
run;
```

BY-Gruppen-Verarbeitung für ein gesamtes Programm:

3. Ermitteln aller Typen und der Anzahl

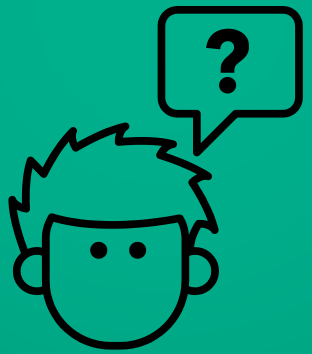
```
proc sql noprint;
  select distinct TYPE
     into :varVal1-
        from SASHELP.CARS;
  %let varCount = &sqlobs;
quit;
```

BY-Gruppen-Verarbeitung für ein gesamtes Programm:
4. Makroprogramm für eine Auswertung pro Wert von TYPE

```
%macro ReportOnEachType;
  %do i = 1 %to &varCount;
    title "&&varVal&i. by Origin in SASHELP.CARS";
    proc freq data=sashelp.cars (where=(type="&&varVal&i."));
       table origin*type /nocum nopercent list;
    run;

    proc sgplot data=sashelp.cars (where=(type="&&varVal&i."));
       histogram mpg_city;
       density mpg_city / type=kernel;
    run;
  %end;
%mend;
%ReportOnEachType;
```

Fragen?



SAS Kursempfehlung

SAS® Makrosprache 1: Grundlagen

- 12. – 14. Oktober, Englisch
  03. – 05. November, Deutsch

SAS® SQL 1: Essentials

08. – 10. November, Deutsch
22. – 24. November, Englisch

SAS®
LEARNING
CONFERENCE

Coming soon:
SAS® Learning Conference
The Power of Connectivity

Save the date: **18. November 2021** (virtueller Event)

Nächstes Webinar@Lunchtime:

21. Oktober 2021

www.sas.de/lunchtime

Folien zum Download unter www.sas.de/lunchtime

Vielen Dank für Ihr Feedback ☺

sas.com