Ways to Summarize Data Using SUM Function in SAS

Anjan Matlapudi and J. Daniel Knapp Pharmacy Informatics, PerformRx, The Next Generation PBM, 200 Stevens Drive, Philadelphia, PA 19113

ABSTRACT

SUM is one of the most frequently used SAS [®] functions for aggregating numeric variables. Although summarizing data using the SUM function is a simple concept, it can become more complex when we deal with large data sets and many variables. This can sometimes lead to inaccurate results. Therefore it requires careful logic to choose the most appropriate function or procedure in each situation in order to output accurate results when we roll up or group data.

There are several ways to summarize data using the SUM function. This paper illustrates various methods ranging from using the SUM function in the simple data step to using the SUM function in SAS procedures such as PROC PRINT, PROC SUMMARY, PROC MEANS, PROC TABULATE and PROC SQL. This paper also covers how SAS handles missing values when you sum data.

INTRODUCTION

Let us first start with the most basic concepts of the SUM function and further explain the best possible way to summarize data including horizontal summation (across variables), vertical summation (across observations), and cumulative summation (running totals). Sample code is incorporated in this paper to generate test data and output results.

GENERATE SAMPLE DATA

The following code generates a sample data set to test various SAS functions and procedures. This sample pharmacy claims data contains the number of drug prescriptions and total drug spend for each drug type (brand drugs versus generic drugs) by pharmacy for the years 2010-2012.

```
*---Generate Sample Dataset---*;
data SampleData;
   input Pharmacy $ 1-10 DrugClass $ 11-17 Prescriptions 19-21
   Y2010 23-24 Y2011 26-28 Y2012 @30 ;
   Label Prescriptions = 'Prescription Volume in Million'
           Y2010 = 'Year 2010 (Drug Spend in Millions)'
           Y2011 = 'Year 2011 (Drug Spend in Millions)'
          Y2012 = 'Year 2012 (Drug Spend in Millions)';
   format Y2010 dollar8.2 Y2011 comma8.2 Y2012 dollar12.2;
   datalines;
         Generic 100 50 20 30
Rite Aid Generic 200 30 10 40
Walgreens Generic 300 60 20 20
Walmart Generic 400 . 30 30
        Brand 100 50 20 30
CVS
Rite Aid Brand 200 30 10 40
Walgreens Brand 300 60 20 20
Walmart Brand 400 70 30 30
      Unknown 500 70 10 10
run;
```

Notice in row #4 the Y2010 field has a missing numeric variable. Also note that in row #9 the Pharmacy field has one missing character variable. We purposefully placed these two examples in this data set to test how SAS handles missing values while we summarize this data. We also formatted the Y2010, Y2011, and Y2012 variables differently to examine how SAS summarizes variables with differing formats.

	Drugs Disp	ensed from Pharmac	cies During Yo	ear 2010-2012	
Pharmacy	Drug Class	Prescriptions	Y2010	Y2011	Y2012
cvs	Generic	100	\$50.00	20.00	\$30.00
Rite Aid	Generic	200	\$30.00	10.00	\$40.00
Walgreens	Generic	300	\$60.00	20.00	\$20.00
Walmart	Generic	400		30.00	\$30.00
CVS	Brand	100	\$50.00	20.00	\$30.00
Rite Aid	Brand	200	\$30.00	10.00	\$40.00
Walgreens	Brand	300	\$60.00	20.00	\$20.00
Walmart	Brand	400	\$70.00	30.00	\$30.00
	Unknown	500	\$70.00	10.00	\$10.00

BASIC SUM FUNCTIONS

Let us start with the simple SUM function to see how SAS computes horizontal summation. The DATA step below (Figure #1) is used to summarize the total amount of drug spend in all three years. First, we will use the arithmetic operator (+) to sum the drugs sold within three years and assign to the AddVar variable. Notice that SAS correctly computed all the variables horizontally, except the WalMart AddVar variable, which is showing a null value. This example demonstrates that SAS ignores missing values when we use the '+' operator.

Similarly the SAS system has another function called SUMABS, which only computes absolute values as we see the highlighted in SumVar4 variable. If you use list of variable such as example A-C, 1-3, Y2010-Y012 then you must place 'of' word in front of the list. If you use '0' in front of 'of' then SAS returns '0' instead of null if the observation is missing values.

The TotalCost variable example shows that we can utilize additional mathematical operators to perform additional calculations in combination with the SUM function.

```
*---Basic sum functions---*;
data SUMOut;
   set SampleData;
   AddVar = Y2010+ Y2011+ Y2012;
   SumVar1=sum(Y2010,Y2011,Y2012);
   SumVar2=sum(of Y2010-Y2012);
   SumVar3=sum(0,of Y2010);
   SumVar4=sumabs(Y2010-Y2012);
   TotalCost=sum(y2010,y2011,y2012)*Prescriptions;
run;
```

		Basio	Sum Fund	tion Acros	s Variables	3					
	Drug					Add	Sum	Sum	Sum	Sum	total
Pharmacy	Class	Prescriptions	Y2010	Y2011	Y2012	Var	Var1	Var2	<mark>Var3</mark>	<mark>Var4</mark>	Cost
cvs	Generic	100	\$50.00	20.00	\$30.00	100	100	100	50	20	10000
Rite Aid	Generic	200	\$30.00	10.00	\$40.00	80	80	80	30	10	16000
Walgreens	Generic	300	\$60.00	20.00	\$20.00	100	100	100	60	40	30000
Walmart	Generic	400		30.00	\$30.00		60	60	0		24000
CVS	Brand	100	\$50.00	20.00	\$30.00	100	100	100	50	20	10000
Rite Aid	Brand	200	\$30.00	10.00	\$40.00	80	80	80	30	10	16000
Walgreens	Brand	300	\$60.00	20.00	\$20.00	100	100	100	60	40	30000
Walmart	Brand	400	\$70.00	30.00	\$30.00	130	130	130	70	40	52000
	Unknown	500	\$70.00	10.00	\$10.00	90	90	90	70	60	45000

SUM WITH PRINT PROCEDURE

The PROC PRINT procedure can output vertical summation results very quickly, but can only output results in the output window. Note that the PROC PRINT procedure does not have the capability to add a new variable.

```
proc print data = SampleData noobs;
sum Y2010 Y2011 Y2012;
run;
```

	Drugs Disp Drug	ensed from Pharma	cies During `	Year 2010-2012	2
Pharmacy	Class	Prescriptions	Y2010	Y2011	Y2012
cvs	Generic	100	\$50.00	20.00	\$30.00
Rite Aid	Generic	200	\$30.00	10.00	\$40.00
Walgreens	Generic	300	\$60.00	20.00	\$20.00
Walmart	Generic	400		30.00	\$30.00
cvs	Brand	100	\$50.00	20.00	\$30.00
Rite Aid	Brand	200	\$30.00	10.00	\$40.00
Walgreens	Brand	300	\$60.00	20.00	\$20.00
Walmart	Brand	400	\$70.00	30.00	\$30.00
	Unknown	500	\$70.00	10.00	\$10.00
			======	======	
			\$420.00	170.00	\$250.00

The By statement is used below in the PRINT procedure to group the DrugClass variable and display the results separately. The "WHERE" clause is used to eliminate the missing pharmacy.

```
*---Sort Data before Using By Clause---*;
proc sort data = SampleData;
  by drugclass;
run;

proc print data = SampleData noobs;
  Title'Summary Details for Each Drug Class';
  Title2'With Running Totals';
  by drugclass ;
  sum Y2010 Y2011 Y2012;
  where Pharmacy ne '';
run;
```

	Summary Detail With F DrugCl	Running Tota	ls	
Pharmacy	Prescriptions	Y2010	Y2011	Y201
CVS	100	\$50.00	20.00	\$30.0
Rite Aid	200	\$30.00	10.00	\$40.0
Walgreens	300	\$60.00	20.00	\$20.0
Walmart	400	\$70.00	30.00	\$30.0
		¢210 00	80.00	\$120.0
DrugClass	_	·		
	Dru	ugClass=Gene	<mark>ric</mark>	
		·	<mark>ric</mark>	
		ugClass=Gene Y2010	<mark>ric</mark>	Y201
Pharmacy	Prescriptions	ugClass=Gene Y2010 \$50.00	<mark>ric</mark>	Y201 \$30.0
Pharmacy CVS	Prescriptions	ugClass=Gene Y2010 \$50.00	ric Y2011 20.00 10.00	Y201 \$30.0 \$40.0
Pharmacy CVS Rite Aid	Prescriptions 100 200	Y2010 \$50.00 \$30.00	ric Y2011 20.00 10.00	Y201 \$30.0 \$40.0 \$20.0
Pharmacy CVS Rite Aid Walgreens	Prescriptions 100 200 300	Y2010 \$50.00 \$30.00	Y2011 20.00 10.00 20.00	Y201 \$30.0 \$40.0 \$20.0
Pharmacy CVS Rite Aid Walgreens	Prescriptions 100 200 300	Y2010 \$50.00 \$30.00 \$60.00	Y2011 20.00 10.00 20.00 30.00	Y201 \$30.0 \$40.0 \$20.0
Pharmacy CVS Rite Aid Walgreens Walmart	Prescriptions 100 200 300	Y2010 \$50.00 \$30.00 \$60.00	Y2011 20.00 10.00 20.00 30.00	\$30.0 \$40.0 \$20.0 \$30.0 \$30.0

You can also use the PAGEBY option to print each drug class on a separate page as shown in the output results below (page break):

```
proc print data = SampleData(where=(Pharmacy NE '')) noobs;
sum y2010 y2011 y2012;
Title'Summary Details for Each Drug Class';
Title2'Page By Each Drug Class';
id pharmacy;
by drugclass ;
pageby drugclass;
run;
```

	•	ls for Each [-	
	Page By	Each Drug C	Lass	
	Dru	gClass=Brand		
Pharmacy	Prescriptions	Y2010	Y2011	Y2012
CVS	100	\$50.00	20.00	\$30.00
Rite Aid	200	\$30.00	10.00	\$40.0
Walgreens	300	\$60.00	20.00	\$20.00
Walmart	400	\$70.00	30.00	\$30.00
DrugClass		\$210.00	80.00	\$120.0
		Page Break		
	Summary Detai			
	Summary Detai		Orug Class	
	Summary Detai	ls for Each I Each Drug C	Orug Class Lass	
	Summary Detai Page By	ls for Each I Each Drug C	Orug Class Lass	
	Summary Detai Page By	ls for Each I Each Drug C lass=Generic	Orug Class Lass	Y201:
Pharmacy	Summary Detai Page By DrugC Prescriptions	ls for Each I Each Drug C lass=Generic Y2010	Orug Class Lass Y2011	
Pharmacy CVS	Summary Detai Page By DrugC Prescriptions 100	ls for Each I Each Drug C lass=Generic Y2010 \$50.00	Orug Class Lass Y2011 20.00 10.00	Y201: \$30.00 \$40.00
Pharmacy CVS Rite Aid	Summary Detai Page By DrugC Prescriptions 100 200	ls for Each I Each Drug C: lass=Generic Y2010 \$50.00 \$30.00	Y2011 20.00 10.00 20.00 30.00	Y201: \$30.00
Pharmacy CVS Rite Aid Walgreens	Summary Detai Page By DrugC Prescriptions 100 200 300	ls for Each I Each Drug C: lass=Generic Y2010 \$50.00 \$30.00	Y2011 20.00 10.00 20.00 30.00	Y201: \$30.01 \$40.01 \$20.01

Similarly, all of the above PRINT SUM examples can be performed using PROC SQL statements. SQL scripts automatically output the results to the output window unless you use the create table statement within PROC SQL, in which case results are output to a dataset.

```
proc sql;
    *create table as;
    select Y2010, Y2011, Y2012,
    sum(Y2010,Y2011,Y2012) as YearTotals
    from SampleData;
quit;
```

PRINT LAST SUM OBSERVATION

The statement below outputs the last observation from the sample dataset.

```
data SumFinal;
   set SampleData end=Lastobs;
   YearSum = sum(Y2010,Y2011,Y2012);
   if lastobs;
run;
```

			Basic Sum	Function Acro	oss Variables		
,	Pharmacy	Drug Class	Prescriptions	Y2010	Y2011	Y2012	Year Total
		Unknown	500	\$70.00	10.00	\$10.00	<mark>90</mark>

PROC SUMMARY

PROC SUMMARY is one of the most powerful procedures to summarize numeric variables and place aggregated results into a new SAS data set. The syntax below shows how to sum drug spend per year (Y2010, Y2011, Y2012) and output in a new data set.

```
proc summary data = SampleData;
  var Y2010 Y2011 Y2012;
  output out=ProcSumOut sum=;
run;
proc print data =ProcSumOut noobs;
  Title 'Total Drugs Spend in the Year 2010-2012';
run;
```

	Total Amoun	t Sold in the	e Year 2010-20	12
TYPE	_FREQ_	Y2010	Y2011	Y2012
0	9	\$420.00	170.00	\$250.00

You can use either the BY or CLASS statement to group selected variables in the dataset. The CLASS or BY statement is used to group the drug spend per pharmacy for each year. Notice the VAR statement is used to aggregate numeric values for Y2010,Y2011 and Y2012. The OUTPUT statement is used to place the summarized results into the output ProcSumClassOut dataset.

It is important to note that SUMMARY procedure creates two automatic variables (_TYPE_ and _FREQ_). The _TYPE_ value yields '0' for the grand total row and '1' through 'N' for the remaining depending on the number of class variables. In the following example the _TYPE_ value for the 4 pharmacies is '1' because there is only one CLASS variable.

The _FREQ_ variable denotes the numbers of observations per pharmacy (we input two records for each pharmacy, one for 'generic' drugs and one for 'brand' drugs).

```
proc summary data = SampleData(where=(drugclass <> 'Unknown'));
  class Pharmacy;
  var Y2010 Y2011 Y2012;
  output out=ProcSumClassOut sum=;
run;
```

		Proc Sumr	nary with Cla	ss Statement	
Pharmacy	_TYPE_	_FREQ_	Y2010	Y2011	Y2012
	0	8	\$350.00	160.00	\$240.00
CVS	1	2	\$100.00	40.00	\$60.00
Rite Aid	1	2	\$60.00	20.00	\$80.00
Walgreens	1	2	\$120.00	40.00	\$40.00
Walmart	1	2	\$70.00	60.00	\$60.00

The BY statement is used to group the data by pharmacy and yields similar results as the CLASS statement, the difference being the grand total (_TYPE_ = 0) applies to each pharmacy, rather than applying to overall

total of all pharmacies as in the previous example. Use of the BY statement requires first sorting of the data set.

```
*---Sort data when we use BY statement---*;
Proc sort data = Sampledata;by Pharmacy;
run;

proc summary data = SampleData(where=(drugclass <> 'Unknown'));
    By Pharmacy;
    var Y2010 Y2011 Y2012;
    output out=ProcSumByOut sum=;
run;
```

	Statement	mary with By S	Proc Summ							
Y2012	Y2011	Y2010	_FREQ_	_TYPE_	Pharmacy					
\$60.00	40.00	\$100.00	2	0	CVS					
\$80.00	20.00	\$60.00	2	0	Rite Aid					
\$40.00	40.00	\$120.00	2	0	Walgreens					
\$60.00	60.00	\$70.00	2	0	Walmart					

The following example displays the use of two CLASS variables. This breaks down each pharmacy by Brand versus Generic drug spend.

```
Proc summary data = SampleData (where =(drugclass <> 'Unknown'));
   Class Pharmacy DrugClass;
   var Y2010 Y2011 Y2012;
   output out=ProcSumClass2Out sum=;
run;
```

		Proc	Summary O	utput Data		
	Drug					
Pharmacy	Class	_TYPE_	_FREQ_	Y2010	Y2011	Y201
		0	8	\$350.00	160.00	\$240.0
	Brand	1	4	\$210.00	80.00	\$120.0
	Generic	1	4	\$140.00	80.00	\$120.0
CVS		2	2	\$100.00	40.00	\$60.0
Rite Aid		2	2	\$60.00	20.00	\$80.0
Walgreens		2	2	\$120.00	40.00	\$40.0
Walmart		2	2	\$70.00	60.00	\$60.0
CVS	Brand	3	1	\$50.00	20.00	\$30.0
CVS	Generic	3	1	\$50.00	20.00	\$30.0
Rite Aid	Brand	3	1	\$30.00	10.00	\$40.0
Rite Aid	Generic	3	1	\$30.00	10.00	\$40.0
Walgreens	Brand	3	1	\$60.00	20.00	\$20.0
Walgreens	Generic	3	1	\$60.00	20.00	\$20.0
Walmart	Brand	3	1	\$70.00	30.00	\$30.0
Walmart	Generic	3	1		30.00	\$30.0

The following code uses a 'WHERE' clause to specify the _TYPE_ to output, in this case the grand total:

```
proc summary data = SampleData(where=(drugclass <> 'Unknown'));
  class Pharmacy;*--(or By);
  var Y2010 Y2011 Y2012;
  output out=ProcSumDropVarOut(where=(_TYPE_ = 0)) sum=;
run;
```

		Proc Sumi	nary Output D	ata	
Pharmacy	_TYPE_	_FREQ_	Y2010	Y2011	Y2012
	0	8	\$350.00	160.00	\$240.00

PROC MEANS

The MEANS and SUMMARY procedures are the same functionality, except the MEANS procedure displays results in the output window. The following is the syntax for the MEANS procedure. By default, the MEANS procedure displays the 'N', 'MEAN', 'STD', 'MAX', and 'MIN' statistical measures. Many additional descriptive statistical measures are available if specified. You can add these options as shown below; notice the 'N' and 'NMISS' options output the number of missing and non missing observations.

```
proc means N NMISS NONOBS SUM MEAN MAXDEC =0 data =SampleData;
  var Y2010 Y2011 Y2012;
  Title 'Means Procedure with Output Options';
  run;
```

	Means Procedure with Ou	utput Options			
Variable	Label	Sum	Mean	N	Mis
Y2010	Year 2010 (Amount in Millions)	420	53	8	
Y2011	Year 2011 (Amount in Millions)	170	19	9	
Y2012	Year 2012 (Amount in Millions)	250	28	9	

Another example displays the use of the CLASS statement.

```
proc means data =SampleData(where=(Pharmacy NE ''));
  class Pharmacy;
  var Y2010 Y2011 Y2012;
  output out=ProcMeanByOut SUM=;
run;
```

Proc Means with Class Output Data						
Pharmacy	_TYPE_	_FREQ_	Y2010	Y2011	Y2012	
	0	8	\$350.00	160.00	\$240.00	
cvs	1	2	\$100.00	40.00	\$60.00	
Rite Aid	1	2	\$60.00	20.00	\$80.00	
Walgreens	1	2	\$120.00	40.00	\$40.00	
Walmart	1	2	\$70.00	60.00	\$60.00	

PROC TABULATE

The TABULATE procedure utilizes the SUM function to generate tabular reports. Below is the basic syntax to sum drugs spent in the year 2010.

```
*---Basic Syntax for Proc tabulate---*;
proc tabulate data=SampleData;
   Title ' yeardrug sale for each Pharmacy';
   var Y2010;
   table Y2010;
run;
```

Drugs Spend in Year 2010



If you want output for all the years drugs spent, you can use the following code to generate three tables.

```
proc tabulate data=SampleData;
  Title ' yeardrug sale for each Pharmacy';
  class DrugClass;
  var Y2010 Y2011 Y2012;
  table DrugClass Y2010;
  table DrugClass Y2011;
  table DrugClass Y2012;
run;
```

Yearly Drugs Dispensed from Each Pharmacy

	Year 2010 (Amount in			
Brand Generic		Unknown	Millions)	
N	N	N	Sum	
4.00	4.00	1.00	420.00	

Yearly Drugs Dispensed from Each Pharmacy

	DrugClass			
Brand	Brand Generic		(Amount in Millions)	
N	N	N	Sum	
4.00	4.00	1.00	170.00	

Yearly Drugs Dispensed from Each Pharmacy

	DrugClass			
Brand	Generic	Unknown	(Amount in Millions)	
N	N	N	Sum	
4.00	4.00	1.00	250.00	

The TABULATE procedure is an excellent format to present your data in single or multiple dimensional tables. The code below displays a two dimensional table that represents total drug spend for each pharmacy and by each drug class for year 2010.

```
proc tabulate data=SampleData;
   Title ' yeardrug sale for each Pharmacy';
   var Y2010;
   class Pharmacy DrugClass;
   Table Pharmacy, DrugClass*Y2010*SUM ;
run;
```

Dimensional Table for Year 2012

	DrugClass			
	Brand Generic			
	Year 2010 (Drug Spend in Millions)	Year 2010 (Drug Spend in Millions)		
	Sum	Sum		
Pharmacy				
cvs	50.00	50.00		
Rite Aid	30.00	30.00		
Walgreens	60.00	60.00		
Walmart	70.00			

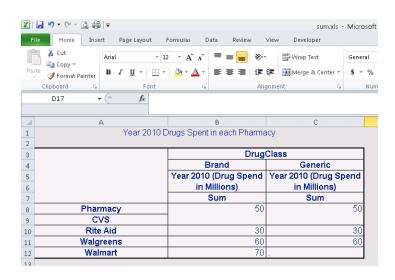
Another advantage is we can easily output these tables in HTML using ODS (output delivery system).

```
ods html Body = 'Pharmacy Details.HTML';
proc tabulate data=SampleData;
   Title 'Year 2010 Drugs Spent in each Pharmacy';
   var Y2010;
   class Pharmacy DrugClass;
   Table Pharmacy, DrugClass*Y2010*SUM;
run;
ods html close;
```

	DrugClass				
	Brand	Generic Year 2010 (Drug Spend in Millions)			
	Year 2010 (Drug Spend in Millions)				
	Sum	Sum			
Pharmacy					
cvs	50.00	50.00			
Rite Aid	30.00	30.00			
Walgreens	60.00	60.00			
Walmart	70.00				

We can also output the end results into an excel spread sheet as shown below.

```
ods listing close;
ods html file = 'C:\Anjan Personel\NESUG 2012\Sum Functions\sum.xls';
proc tabulate data=SampleData;
   Title 'Year 2010 Drugs Spent in each Pharmacy';
   var Y2010;
   class Pharmacy DrugClass;
   Table Pharmacy, DrugClass*Y2010*SUM;
run;
ods html close;
ods listing;
```



SQL PROCEDURE

SAS capability also includes Structured Query Language, which we can utilize to apply inner queries, sub queries, select case, joint multiple datasets, macros and even stored procedures in PROC SQL.

The query below is a simple example to sum yearly totals by drug class as well compute grand totals.

```
proc sql;
  title ' SQL Procedure to Output Results using SUM Function';
  select drugclass,
  Y2010,Y2011,Y2012,
  sum(Y2010) as Y2010_total,
  sum(Y2011) as Y2011_total,
  sum(Y2012) as Y2012_total,
  sum(Y2010, Y2011, Y2012) as YearTotal,
  sum(sum(Y2010, Y2011, Y2012)) as FinaleTotal
  from SampleData
  group by drugclass
  order by drugclass;
quit;
```

		SQL Pr	ocedure to Output	Results	using SUM Fu	inction		
	Year 2010	Year 2011						
	(Drug	(Drug	Year 2012					
Drug	Spend in	Spend in	(Drug Spend	Y2010_	Y2011_	Y2012_	Year	<mark>Finale</mark>
Class	Millions)	Millions)	in Millions)	total	total	total	Total	Total
Brand	\$30.00	10.00	\$40.00	210	80	120	80	410
Brand	\$50.00	20.00	\$30.00	210	80	120	100	410
Brand	\$60.00	20.00	\$20.00	210	80	120	100	410
Brand	\$70.00	30.00	\$30.00	210	80	120	130	410
Generic	\$60.00	20.00	\$20.00	140	80	120	100	340
Generic	\$50.00	20.00	\$30.00	140	80	120	100	340
Generic	\$30.00	10.00	\$40.00	140	80	120	80	340
Generic		30.00	\$30.00	140	80	120	60	340
Unknown	\$70.00	10.00	\$10.00	70	10	10	90	90

CONCLUSION

We covered various ways to use the SUM function in DATA step and PROC step programming. The SUM function is very versatile. You have alternatives such as SUMMARY, MEANS, or SQL to summarize your data. You can use choose TABULATE procedure to produce tabular formats in MSExcel or HTML

In this paper, we only covered the SUM functionality, but all these procedures have additional statistical functions and options that you can use depending on your data analysis goals. We also provided you sample code to generate test data. You can copy and paste this logic into the SAS program window and use the example code and output results to better understand this capability. We hope this paper has provided you with new and different ideas that you can implement in your data analysis, while having fun summing your data.

REFERENCES

Curtis A. Smith, New Ways and Means to Summarize Files. Data Warehousing and Enterprise Solutions SUGI28 Paper 165-22

Frank Ferriola, 'What's Your _TYPE_? How to find the Class You Want in PROC SUMMARY'. <u>SUGI 27</u> Coder's Corner Paper 77-27.

AKNOWLDEGMENTS

We would like to acknowledge Mr. Shimels Afework, Senior Director, PerformRx.

PerformRx provides pharmacy benefit management (PBM) services through proactively managing escalating pharmacy costs while focusing on clinical improvement and financial results.

CONTACT INFORMATION:

Your comments and questions are valued and encouraged. Contact the authors at:

Name Anjan Matlapudi

Senior Pharmacy Analyst, Pharmacy Informatics Department

Address PerformRx, The Next Generation PBM

200 Stevens Drive

Philadelphia, PA 19113

Work Phone: (215)937-7252 Fax: (215)863-5100

E-mail: anjan.matlapudi@performrx.com

anjanmat@gmail.com

Name Knapp, J. Daniel, MBA

Senior Manager, Pharmacy Informatics Department

Address PerformRx, The Next Generation PBM

200 Stevens Drive

Philadelphia, PA 19113

Work Phone: (215)937-7251 Fax: (215)863-5100

E-mail: <u>Daniel.Knapp@performrx.com</u>

Jdjeep57@yahoo.com

SAS® is a registered trademark or trademark of SAS® Institute, Inc. in the USA and other countries.