SO YOU WANT TO LEARN PROC REPORT
Chris Moriak, Boehringer Ingelheim, Ridgefield, CT

ABSTRACT

Producing attractive reports is essential when working with either internal or external clients. By now, you have probably heard about PROC REPORT in the non-windowing environment, but you may have never used it. If you want to learn PROC REPORT but do not have either the time nor money to take a SAS course or read the extensive manual, then this paper is for you. I have condensed the basic components and major options of PROC REPORT into this short paper. This paper is for both beginners in its step-by-step instructions, as well as for advanced users with its reference section containing nearly every available option. The paper also contains two methods for easily calculating percentages within groups and for the entire report, and a useful medical utilization report.

INTRODUCTION

What is fantastic about PROC REPORT is that by learning the basics, one can create seemingly complicated reports. This paper introduces the basics of displaying data, using PROC REPORT to generate statistics, creating new columns, and adding summary lines. When finished reading this paper, one will see that by knowing the basic concepts that are outlined in this paper, one can produce Example 1 with no pre or post-processing of the data, using just one PROC REPORT.

BASIC REPORT SYNTAX

PROC REPORT is a powerful procedure that allows a programmer to do lists, subsets, statistics, and computations all within one procedure. It has 3 basic steps and 4 optional steps. Its syntax is as follows:

Option 1 - Subset Data
WHERE variable = ;

Option 2 – By Statement
BY variable1 variable2 ;

Step 2 - Assign Report Variables
COLUMN variable1 variable2 variableN ;

Step 3 - Define the Report Variables
DEFINE variable / type FORMAT= ‘label’ WIDTH= options ;

Option 3 - Define New Computed Variables
COMPUTE variable;
variable=statements ;
ENDCOMP;

Option 4 - Add Summary Lines
BREAK AFTER variable / options ;
RBREAK AFTER / options ;

RUN ;

Let’s now take a closer look at each step.

Step 1 - Call Procedure
PROC REPORT DATA=dataset options ;

As with nearly all PROC statements, the first part of PROC REPORT is to identify the dataset that SAS will be using. This is done with a DATA=dataset statement. Also like most PROCs, a list of options can follow (See Table A). I will touch upon the most common.

The first option is the windows/no windows option. This option tells SAS whether to send the output to the interactive Report window (similar to SAS/ASSIST), or to send the output to the Output window. The default is windows. Much of the time, one will want to avoid sending output to an interactive window, especially when the program is running in batch, or when the report is in...
production or a validated status. Therefore, this paper concentrates on the non-windowing environment. The option for this is NOWD, short for NOWINDOWS.

Next comes HEADLINE and HEADSKIP. These two options perform what they sound like. HEADLINE provides a header line below the column headers, while HEADSKIP skips one line between the column headers and the first row of data.

There are four other common options. One is SPLIT=‘character’. This option wraps the column header and contents at the character specified, so be sure to use an uncommon character such as ~ or /. Another option is SPACING=numeric. This specifies the number of spaces between columns. The other two options are for page formatting: PS=numeric for page-size and LS=numeric for line-size. If one has already set PS and LS in a global option statement, then one does not have to add them to the call procedure line.

Putting these options together, the procedure call statement reads:

```sql
proc report data=dataset nowd headline headskip split=’~’ spacing=2 ls=142 ps=53;
```

The statement above will be the same for most every PROC REPORT one creates.

**Option 1 – Subset Data**

Similar to other PROCs, PROC REPORT allows one to subset the data set by using any valid WHERE clause. For example, if we only wanted a subset of New York providers, the statement would be:

```sql
where state = ‘NY’ ;
```

This WHERE clause can be used on several variables by using AND or OR between the clauses:

```sql
where state = ‘NY’ and provider = ‘HOSP1’ ;
```

**Option 2 – BY Statement**

The BY statement can be used if one wants to separate the report into page sections by a variable or group of variables. The dataset, however, must be sorted by the BY variable(s) before the program invokes PROC REPORT. The statement is simply:

```sql
by variable1 variable2 ;
```

One can print the BY variable by using #byval$n in the TITLE statement where $n is the placement number of where the variable resides in the BY statement:

```sql
title1 “State = #byval1” ;
```

**Step 2 - Assign Report Variables**

The next step is to identify the columns of the report. First, write the term COLUMN, followed by an ordered list of variables that will comprise your columns. The order of the list must match the order in which the columns will appear in the report. There are three additional elements of note.

First, if one wants to add the frequency (number of non-missing values) of a variable in the report, one accomplishes this by adding an N to the COLUMN statement. For example, if we wanted to include the frequency of CPTCODE for each PROVIDER, our COLUMN statement would read:

```sql
column provider cptcode n ;
```

Second, if one wants to perform multiple statistics on a specific variable, then one needs to create a new column variable and tell SAS that it is equal to the variable. For example, if we wanted the mean of PAYAMT as well the sum, then we need to create a new column variable such as AVGPAY and tell SAS that PAYAMT=AVGPAY. Our column statement would look like this:

```sql
column provider cptcode n payamt=avgpay ;
```

If we are adding a computed column (a variable not on the dataset and that has to be calculated), then we need to include the computed variable name to the list where it would be situated. For example, if we were to include a column called NEWRATE after PAYAMT that is 90% of PAYAMT, then our COLUMN statement would read:

```sql
column provider cptcode n payamt newrate payamt=avgpay;
```

**Step 3 - Define the Report Variables (type, format, ‘label’ width)**

The next step is what makes PROC REPORT seem long and complicated. On the contrary, it is performing the same actions as one would do in Excel, Access, Power Point, or any other tool for report writing. With all tools, one needs to define one’s report variables. In
PROC REPORT, one does this by using the term DEFINE for each column (variable) that will be in the report.

The syntax for defining the report variables:

```
DEFINE variable / type FORMAT= 'label'
           WIDTH=  options ;
```

There are six “types” for defining a variable: GROUP, ORDER, DISPLAY, ANALYSIS, COMPUTED, and ACROSS (See Table B). I will discuss the most commonly used types.

GROUP is used to specify the class variables. As with SQL, the report will be summarized by this variable. Each class, or summary, variable will have this define type.

ORDER sorts the data by this variable, much like the option in SQL. Since one can also control the sort order by a define option, I personally use the type GROUP instead of ORDER.

DISPLAY lists the values as they appear in the data. This is the default type.

ANALYSIS statistic tells SAS to perform the specified statistical function on the variable. A list of available statistics can be found in Table C.

COMPUTED informs SAS that the variable does not appear on the dataset and will need to be calculated. This calculation occurs later in the COMPUTE block.

After the define type, one states the format, if any, for SAS to use for the variable by writing FORMAT=format. Next comes the header, or label, of the column in quotes. The last term used is WIDTH=numeric, which specifies the width of the column. An example line would read:

```
define cptcode / group format=$code. “CPT~CODE”
               width=30 ;
```

Notice the tilda between “CPT” and “CODE”. This is our SPLIT character, and PROC REPORT will place “CPT” and “CODE” on separate lines.

An example of how one would perform a statistical function on a variable:

```
define payamt / analysis mean format=dollar8.
                “AVG~PAYAMT” width=10;
```

For additional options that one can add to the DEFINE statement, please see Table D.

**Option 3 – Defining Computed Variables**

The ability to compute new variables without having to add them to the dataset is one of PROC REPORT’s strongest attributes. With COMPUTE blocks, one can create a new column, percentages within groups, perform arithmetic on or between columns, or even print text for conditional outcomes.

If one is adding a computed variable, then one calculates the computed variable in a COMPUTE block after the last DEFINE statement:

```
COMPUTE variable ;
variable = statement ;
ENDCOMP ;
```

If the computed variable is based on a calculation of an ANALYSIS type variable, then one needs to refer to that variable in the COMPUTE block as variable.statistic. For example, if we want to create a new column called NEWRATE that is 90% of PAYAMT, then the define statements would be:

```
define payamt / analysis sum format=dollar8.
               “PAID~AMOUNT” width=10  ;
define newrate / computed format=dollar8. “NEW
                     RATE” width=10 ;
```

While the COMPUTE block would read:

```
compute newrate ;
newrate = payamt.sum * 0.9 ;
endcomp;
```

If the computed variable is based on N (the frequency), then use N without any statistic extension:

```
compute newvar ;
newvar = N/100 ;
endcomp;
```

One can even refer to another column by the keyword _cn_, where n is the column number.

For examples on how to calculate percentages within groups and for entire reports, please see the section “Two Useful COMPUTE Statements”.

Option 4 - Adding summary lines

Another good tool of PROC REPORT is the ability to insert summary lines between groups, as well as insert a summary line for the entire report. Two commands perform these functions: BREAK AFTER and RBREAK. BREAK AFTER variable tells SAS to create a break after the last row of each unique value of the variable. To create a summary line, add a slash, then the term SUMMARIZE:

```
BREAK AFTER variable / SUMMARIZE options ;
```

There are other options that one may add after the term SUMMARIZE (See Table E). The most common are OL, UL, DOL, DUL, and SKIP. The option OL will place an OverLine above the summary numbers, while the UL will place an UnderLine below the summary numbers. Likewise, DOL will place a Double OverLine above the summary numbers, while the DUL will place a Double UnderLine below the summary numbers. The term SKIP will skip a line between the summary numbers and the next row.

One can summarize within all GROUP types by adding a separate BREAK AFTER command for each variable. For example, if we were summarizing by STATE and by PROVIDER, then our statements would read:

```
break after provider / summarize ol ul skip ;
break after state / summarize ol ul skip ;
```

To add a total summary line for the report, the term RBREAK AFTER is used with no variable specified. The term SUMMARIZE and the same options for BREAK AFTER can apply. For example:

```
rbreak after / summarize dol dul ;
```

Two Useful COMPUTE Statements

Percent within a Group

One can compute a new column variable that is the percentage of observations for a variable within a group. The following creates a new column variable PERCTN as the percentage of services for each PROVIDER.

```
proc report data=dataset nowd headline headskip
  split="~" spacing=2 ls=152 ps=53 ;
column provider n payamt perctn ;
define provider / group format=$hosp. ‘Hospital’
  width=15 ;
define n / format=comma6. ‘Admits’ width=8 ;
define payamt / analysis sum format=dollar8.
  ‘Paid~Amount’ width=10 ;
define perctn / computed format=percent8.1 ‘% 
of~Admits” ;
```

Second, we need to compute the total number of observations for each PROVIDER and retain the result as TOTN:

```
compute before provider ;
totn = n ;
endcomp;
```

Then, compute PERCTN as the number of observations (frequency) for each CPTCODE (N) and divide by the total number of observation (TOTN):

```
compute perctn ;
perctn = n/totn ;
endcomp ;
```

Percent for Entire Report

One can also create a new column variable that is the percentage of observations for a variable for the entire report. Using our example, we will create the new variable PERCTALL that is the percentage of services for each CPTCODE for all PROVIDERs.

First, we call the procedure, assign the report variables, and define the report variables:

```
proc report data=dataset nowd headline headskip
  split="~" spacing=2 ls=152 ps=53 ;
column provider n payamt perctall ;
define provider / group format=$hosp. ‘Hospital’
  width=15 ;
define n / format=comma6. ‘Admits’ width=8 ;
define payamt / analysis sum format=dollar8.
  ‘Paid~Amount’ width=10 ;
define perctall / computed format=percent8.1 ‘% 
of~Admits” ;
```

Second, we need to calculate the total number of observations and retain the result as TOTALL.

```
compute before ;
totall = n ;
endcomp;
```
Then, compute PERCTALL as the number of observations for each CPTCODE (N) and divides by the total number of observations (TOTN).

\[
\text{compute perctall;}
\]
\[
\text{perctall} = \frac{n}{\text{totall}};
\]
\[
\text{endcomp;}
\]

**Creating a Medical Utilization Report**

With using the information provided above, one can construct the medical utilization report found in Example 1. First, one needs to create a test dataset of claims.

**Generic Data Builder**

The following program will create a generic dataset of claims data consisting of PROVIDER, CPTCODE (procedure code), LOS (length of stay), and PAYAMT. This method can be used to create any kind of dataset, using PROC FORMAT to convert random numbers into data values.

```sas
/* Establishing maximum and minimum values */
%let minhosp=1; /*Minimum hospital range;*/
%let maxhosp=4; /*Maximum hospital range;*/
%let mincpt=1; /*Minimum CPT-code range;*/
%let maxcpt=4; /*Maximum CPT-code range;*/
%let minlos=1; /*Minimum length of stay;*/
%let maxlos=10; /*Maximum length of stay;*/
%let numclms=3000; /*Number of claims;*/

/* Creating formats for data values */
proc format;
  value hosp   1='HOSP1'
  2='HOSP2'
  3='HOSP3'
  4='HOSP4';
  value cpt    1='RMBRD'
  2='RMOBS'
  3='RMPED'
  4='RMPSY';
run;

/* Numbers are selected at random with set minimum and maximum ranges. These numbers are then turned into data values according to created formats. */
data claims (drop= i);
  do i=1 to &numclms ;
    hosp=put((ceil((&maxhosp-&minhosp+1)*ranuni(0)+ &minhosp-1)),hosp.);
    cptcode=put((ceil((&maxcpt-&mincpt+1)*rantri(0,.001) + &mincpt-1)),cpt.);
    los=(ceil((&maxlos-&minlos+1)*rantri(0,.0001) + &minlos-1));
    payamt=los*ranuni(0)*&payrate;
  output;
end;
format payamt 15.2;
run;
```

*Note: The RANTRI function gives a random number using triangular distribution. I used this function to create “real” test data by skewing the data distribution to provide lesser length of stays and more RMBRDs (regular room and board) and RMOBSs (obstetrics room and board).*

Now that we have our dataset, we can use what we have learned and create the following report:

```
**Call Procedure**;
proc report data=claims nowd headline headskip='~' spacing=2 ls=142 ps=53;
**Assign Report Variables**;
column cptcode hosp n payamt los
  los=alos vlos los=mlos apay
  payamt=mpay pcta;
**Define Report Variables**;
define cptcode / group "Cpt~Code" width=5;
define hosp / group "Hospital" width=10;
define n / format=comma6. "Admits" width=6;
define payamt / analysis sum format=dollar12. "Paid~Amount" width=11;
define los / analysis sum format=comma6.2 "ALOS" width=6;
```

```sas
run;
```
define vlos / computed format=comma6.2
  "Diff from~ALOS" width=9;
define mlos / analysis max
  format=comma6. "Max~LOS" width=6;
define apay / computed format=dollar8.
  "Avg Paid-per Day" width=8;
define mpay / analysis mean
  format=dollar8. "Avg Paid~Per Admit"
  width=10;
define pcta / computed
  format=percent8.1 "% of~Total Admits";

**Define Computed Variables**;
compute before;
admits=n;
endcomp;

compute before cptcode;
  proclos=los.sum/n;
endcomp;

compute vlos;
vlos=_c6_-proclos;
endcomp;

compute apay;
apay=payamt.sum/los.sum;
endcomp;

compute pcta;
pcta=n/admits;
endcomp;

**Add Summary Lines**;
break after cptcode /summarize dol dul suppress skip;
run;

Your output will be similar to Example 1. Your numbers will be different due to the creation of the random test data.

### PROC REPORT Option Tables

<table>
<thead>
<tr>
<th>Usage</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADLINE</td>
<td>Places a line under the column names</td>
</tr>
<tr>
<td>HEADSKIP</td>
<td>Skips a line between the column names and the first row</td>
</tr>
<tr>
<td>SPLIT=''</td>
<td>Wraps the column name after the specified character</td>
</tr>
<tr>
<td>NOWD</td>
<td>No windows option</td>
</tr>
<tr>
<td>LS=</td>
<td>Spacing for line-size.</td>
</tr>
<tr>
<td>PS=</td>
<td>Spacing for page-size.</td>
</tr>
<tr>
<td>SPACING=</td>
<td>Specifies spacing between columns</td>
</tr>
<tr>
<td>NOHEADER</td>
<td>Does not print column names</td>
</tr>
<tr>
<td>MISSING</td>
<td>Do not drop missing values in calculations and groupings.</td>
</tr>
<tr>
<td>OUTREPT=</td>
<td>Stores report definition in a catalog entry</td>
</tr>
<tr>
<td>REPORT=</td>
<td>Calls specific stored report definition.</td>
</tr>
<tr>
<td>OUT=</td>
<td>Places output in specified dataset.</td>
</tr>
<tr>
<td>COLWIDTH=</td>
<td>Specifies default column width for numeric and computed variables.</td>
</tr>
<tr>
<td>LIST</td>
<td>Writes report definition to the log.</td>
</tr>
<tr>
<td>NAMED</td>
<td>Writes “name=” before value.</td>
</tr>
<tr>
<td>WRAP</td>
<td>Displays one value of each column, going to a consecutive lines before the printing the next row value.</td>
</tr>
<tr>
<td>PANELS=</td>
<td>Specifies number of panels on a page.</td>
</tr>
<tr>
<td>PSPACE=</td>
<td>Space between panels.</td>
</tr>
<tr>
<td>VARDEF=</td>
<td>Specifies the divisor to use when calculating variances (n, df, wdf, weight)</td>
</tr>
<tr>
<td>SHOWALL</td>
<td>Overrides NOPRINT and NOZERO define options.</td>
</tr>
<tr>
<td>BOX</td>
<td>Puts boxes around all columns and rows</td>
</tr>
</tbody>
</table>
Table B – Types for Defining Variables

Usage | Purpose
--- | ---
GROUP | Summarizes rows based on group variable values (class variable).
ORDER | Orders rows by variable.
ANALYSIS | Perform a statistical function.
ACROSS | Tabular reports with variable values as column headers.
DISPLAY | List values as they appear in the dataset.
COMPUTED | Create calculated variables not on data set.

Table C – Types of Statistical Functions

Statistic | Definition
--- | ---
N | Count non-missing observations
NMISS | Count of missing observations
MEAN | Mean
STD | Standard deviation
MIN | Minimum
MAX | Maximum
RANGE | Range (Maximum-Minimum)
SUM | Sum
USS | Uncorrected sum of squares
CSS | Sum of squares corrected for the mean
STDERR | Standard error of mean
CV | Percent coefficient of variation
T | Student’s t-value when population mean=0
PRT | Two tailed p-value for Student’s t-value
VAR | Variance
SUMWGT | Sum of weights

Table D – Additional Options When Defining a Column

Option | Purpose
--- | ---
CENTER | Centers values within column.
DESCENDING | Orders rows in descending order.
FLOW | Wraps the value of a character variable within the column.
RIGHT | Right justifies values within a column.
LEFT | Left justifies values within a column.
NOPRINT | Suppresses printing of the variable.
NOZERO | Suppresses printing of a column if all values are 0 or missing.
ORDER= | Controls ordering of rows (Data, Formatted, Freq, Internal).
SPACING= | Spacing between column.
PAGE | Places column and all columns to the left on a separate page.

Table E – Options to Use with BREAK or RBREAK

Option | Purpose
--- | ---
OL | Overline
UL | Underline
DOL | Double Overline
DUL | Double Underline
SUMMARIZE | Summarizes the column.
SKIP | Skip a line before the next row
PAGE | Skip a page before the next row
SUPPRESS | Does not write the name of the summarizing variable
CONCLUSION

PROC REPORT has a lot to offer, and I consider it to be one of the most powerful procedures in SAS. Before writing that next report using PROC TABULATE or DATA _NULL_, try PROC REPORT. There are books with excellent examples of how to produce most any kind of report. Please see my references for more information on this procedure.

REFERENCES


ACKNOWLEDGEMENTS

I would like to thank John Adams for his research into solving this problem.

SAS is a registered trademark of SAS Institute Inc., Cary, NC.

CONTACT INFORMATION

The author welcomes comments, suggestions, and questions by phone 203-798-4239, or by e-mail cmoriak@rdg.boehringer-ingelheim.com.
### Example 1

The SAS System

<table>
<thead>
<tr>
<th>CPT CODE</th>
<th>HOSPITAL</th>
<th>ADMITS</th>
<th>AMOUNT</th>
<th>LOS</th>
<th>ALOS</th>
<th>DIFF FROM MAX LOS</th>
<th>AVG PAID PER DAY</th>
<th>AVG PAID PER ADMIT</th>
<th>TOTAL ADMITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMBRD</td>
<td>HOSP1</td>
<td>287</td>
<td>$596,842</td>
<td>1,010</td>
<td>3.52</td>
<td>0.25</td>
<td>$591</td>
<td>$2,080</td>
<td>10.0%</td>
</tr>
<tr>
<td></td>
<td>HOSP2</td>
<td>248</td>
<td>$590,667</td>
<td>951</td>
<td>3.83</td>
<td>0.07</td>
<td>$621</td>
<td>$2,382</td>
<td>8.7%</td>
</tr>
<tr>
<td></td>
<td>HOSP3</td>
<td>233</td>
<td>$510,718</td>
<td>932</td>
<td>4.00</td>
<td>0.23</td>
<td>$548</td>
<td>$2,192</td>
<td>8.1%</td>
</tr>
<tr>
<td></td>
<td>HOSP4</td>
<td>252</td>
<td>$567,657</td>
<td>952</td>
<td>3.78</td>
<td>0.01</td>
<td>$596</td>
<td>$2,253</td>
<td>8.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$2,265,882</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMOBS</td>
<td>224</td>
<td>$531,765</td>
<td>921</td>
<td>4.11</td>
<td>0.28</td>
<td>$577</td>
<td>$2,374</td>
<td>7.8%</td>
</tr>
<tr>
<td></td>
<td>HOSP2</td>
<td>174</td>
<td>$403,207</td>
<td>641</td>
<td>3.68</td>
<td>-0.15</td>
<td>$629</td>
<td>$2,317</td>
<td>6.1%</td>
</tr>
<tr>
<td></td>
<td>HOSP3</td>
<td>216</td>
<td>$552,363</td>
<td>818</td>
<td>3.79</td>
<td>-0.05</td>
<td>$675</td>
<td>$2,557</td>
<td>7.5%</td>
</tr>
<tr>
<td></td>
<td>HOSP4</td>
<td>194</td>
<td>$438,121</td>
<td>718</td>
<td>3.70</td>
<td>-0.13</td>
<td>$610</td>
<td>$2,258</td>
<td>6.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1,925,456</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMPED</td>
<td>152</td>
<td>$310,656</td>
<td>532</td>
<td>3.50</td>
<td>-0.26</td>
<td>$584</td>
<td>$2,044</td>
<td>5.3%</td>
</tr>
<tr>
<td></td>
<td>HOSP2</td>
<td>150</td>
<td>$339,310</td>
<td>552</td>
<td>3.68</td>
<td>-0.08</td>
<td>$615</td>
<td>$2,262</td>
<td>5.2%</td>
</tr>
<tr>
<td></td>
<td>HOSP3</td>
<td>174</td>
<td>$415,001</td>
<td>669</td>
<td>3.84</td>
<td>0.08</td>
<td>$620</td>
<td>$2,385</td>
<td>6.1%</td>
</tr>
<tr>
<td></td>
<td>HOSP4</td>
<td>150</td>
<td>$367,301</td>
<td>601</td>
<td>4.01</td>
<td>0.25</td>
<td>$611</td>
<td>$2,449</td>
<td>5.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1,432,269</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMPSY</td>
<td>123</td>
<td>$285,258</td>
<td>469</td>
<td>3.81</td>
<td>-0.09</td>
<td>$608</td>
<td>$2,288</td>
<td>21.9%</td>
</tr>
<tr>
<td></td>
<td>HOSP2</td>
<td>86</td>
<td>$213,784</td>
<td>362</td>
<td>4.21</td>
<td>0.31</td>
<td>$591</td>
<td>$2,486</td>
<td>3.0%</td>
</tr>
<tr>
<td></td>
<td>HOSP3</td>
<td>108</td>
<td>$265,003</td>
<td>406</td>
<td>3.76</td>
<td>-0.14</td>
<td>$653</td>
<td>$2,454</td>
<td>3.8%</td>
</tr>
<tr>
<td></td>
<td>HOSP4</td>
<td>94</td>
<td>$237,527</td>
<td>366</td>
<td>3.89</td>
<td>-0.01</td>
<td>$649</td>
<td>$2,527</td>
<td>3.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1,001,573</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$2,437</td>
<td>14.3%</td>
</tr>
</tbody>
</table>