Formatting Highly Detailed Reports: Eye-Friendly, Insight-Facilitating

Lisa Fine, United BioSource Corporation

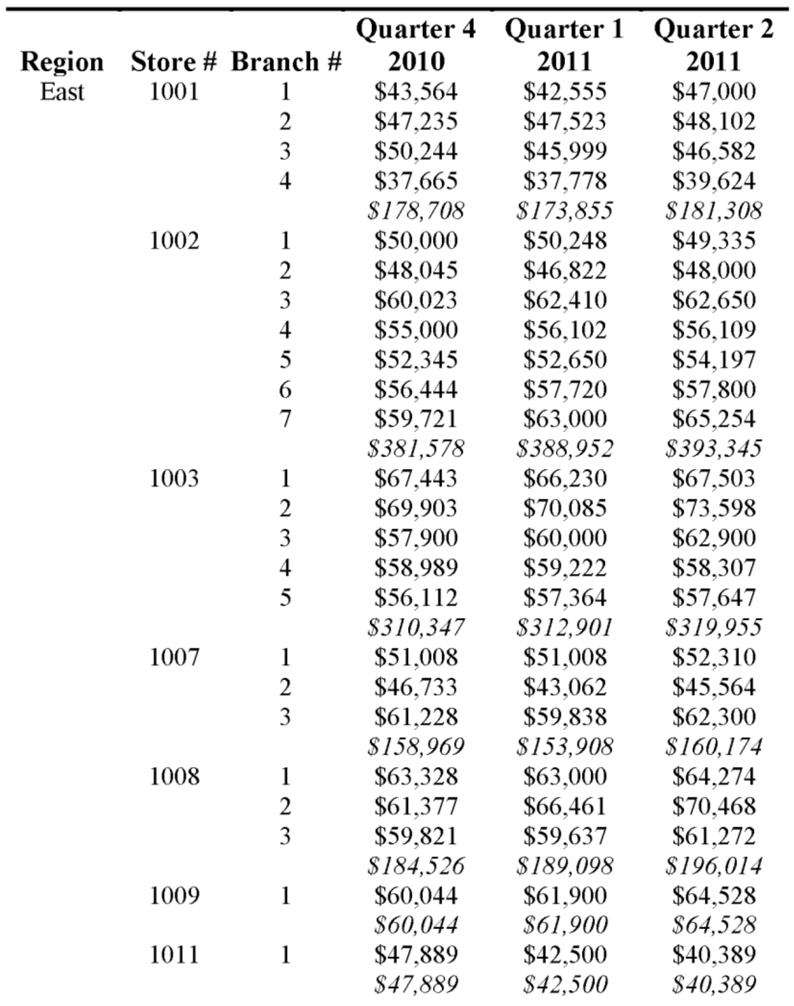
Introduction

Consider a highly detailed report that is overwhelming with numbers, page after page. The results in this type of report can be very difficult for the reader to decipher. However, often, just slight changes in style and formatting can transform a report crammed with details into one in which the reader can identify important items and make sense of the results.

Example: Format National Sales Report

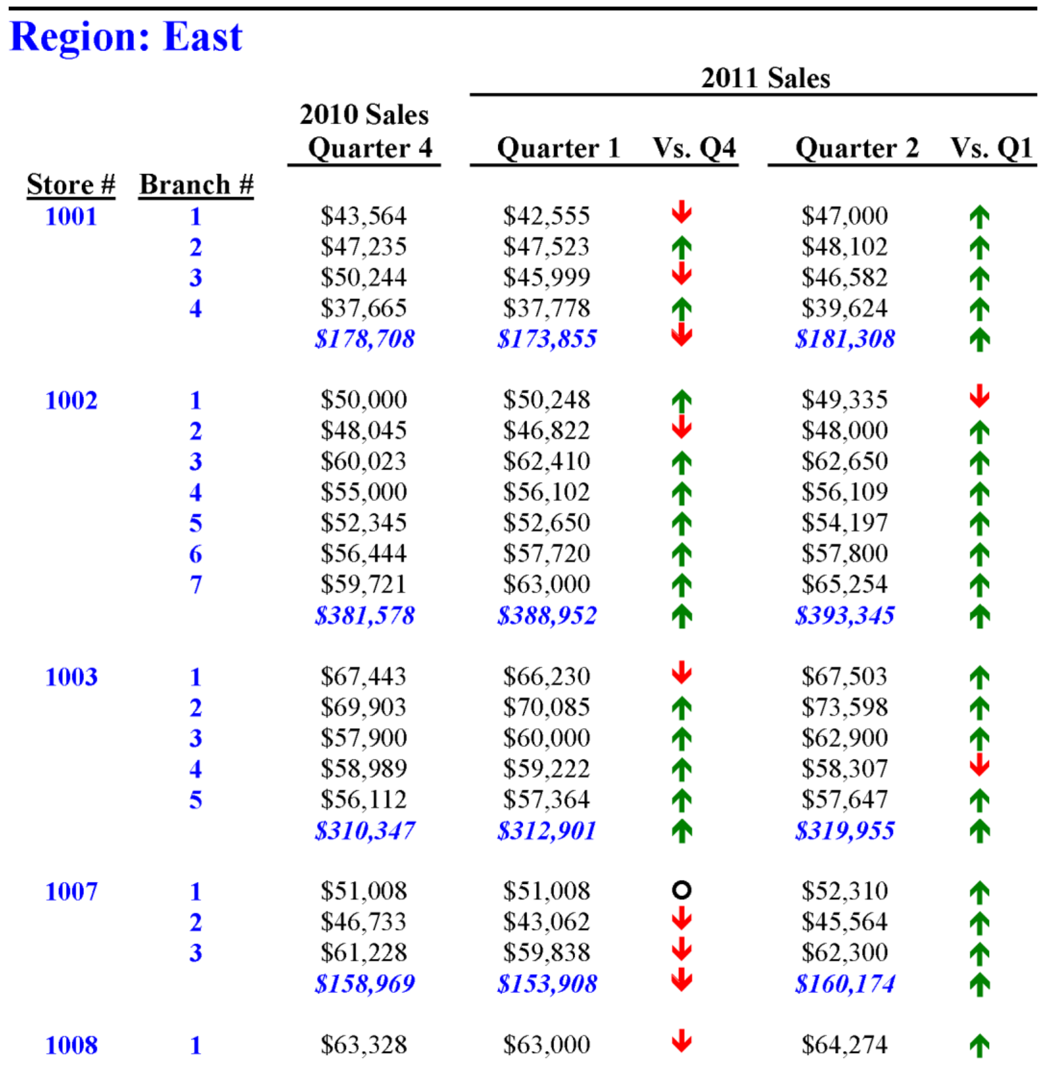
This example demonstrates how to modify a national sales report that displays quarterly sales by Region, Store Number, and Branch Number. Figures 2.1 and 2.2 show BEFORE and AFTER versions of the same report after implementing various ODS and PROC REPORT options. Almost all report enhancements are made using PROC REPORT along with ODS RTF enhancements. The only pre-processing of data in this example is to add a variable named BLANK that will be used to add empty columns in the REPORT procedure.

Figure 2.1 BEFORE Formatting



Region	Store #	Branch #	Quarter 4 2010	Quarter 1 2011	Quarter 2 2011
East	1001	1	$43,564	$42,555	$47,000
		2	$47,235	$47,523	$48,102
		3	$50,244	$45,999	$46,582
		4	$37,665	$37,778	$39,624
			$178,708	*$173,855*	*$181,308*
	1002	1	$50,000	$50,248	$49,335
		2	$48,045	$46,822	$48,000
		3	$60,023	$62,410	$62,650
		4	$55,000	$56,102	$56,109
		5	$52,345	$52,650	$54,197
		6	$56,444	$57,720	$57,800
		7	$59,721	$63,000	$65,254
			$381,578	*$388,952*	*$393,345*
	1003	1	$67,443	$66,230	$67,503
		2	$69,903	$70,085	$73,598
		3	$57,900	$60,000	$62,900
		4	$58,989	$59,222	$58,307
		5	$56,112	$57,364	$57,647
			$310,347	*$312,901*	*$319,955*
	1007	1	$51,008	$51,008	$52,310
		2	$46,733	$43,062	$45,564
		3	$61,228	$59,838	$62,300
			$158,969	*$153,908*	*$160,174*
	1008	1	$63,328	$63,000	$64,274
		2	$61,377	$66,461	$70,468
		3	$59,821	$59,637	$61,272
			$184,526	*$189,098*	*$196,014*
	1009	1	$60,044	$61,900	$64,528
			$60,044	*$61,900*	*$64,528*
	1011	1	$47,889	$42,500	$40,389
			$47,889	*$42,500*	*$40,389*

Figure 2.2 AFTER Formatting



Region: East

		2010 Sales	2011 Sales			
Store #	**Branch #**	**Quarter 4**	**Quarter 1**	**Vs. Q4**	**Quarter 2**	**Vs. Q1**
1001	1	$43,564	$42,555	↓	$47,000	↑
	2	$47,235	$47,523	↑	$48,102	↑
	3	$50,244	$45,999	↓	$46,582	↑
	4	$37,665	$37,778	↑	$39,624	↑
		$178,708	*$173,855*	↓	*$181,308*	↑
1002	1	$50,000	$50,248	↑	$49,335	↓
	2	$48,045	$46,822	↓	$48,000	↑
	3	$60,023	$62,410	↑	$62,650	↑
	4	$55,000	$56,102	↑	$56,109	↑
	5	$52,345	$52,650	↑	$54,197	↑
	6	$56,444	$57,720	↑	$57,800	↑
	7	$59,721	$63,000	↑	$65,254	↑
		$381,578	*$388,952*	↑	*$393,345*	↑
1003	1	$67,443	$66,230	↓	$67,503	↑
	2	$69,903	$70,085	↑	$73,598	↑
	3	$57,900	$60,000	↑	$62,900	↑
	4	$58,989	$59,222	↑	$58,307	↓
	5	$56,112	$57,364	↑	$57,647	↑
		$310,347	*$312,901*	↑	*$319,955*	↑
1007	1	$51,008	$51,008	○	$52,310	↑
	2	$46,733	$43,062	↓	$45,564	↑
	3	$61,228	$59,838	↓	$62,300	↑
		$158,969	*$153,908*	↓	*$160,174*	↑
1008	1	$63,328	$63,000	↓	$64,274	↑

Overall Goals for Formatting the National Sales Report

The overall goals for formatting highly detailed reports are those that make the report easier on the eye.

The overall goals for this paper include:

- Adding white space between rows and columns.
- Making key items stand out with bold font, borders, underlines, and color.
- Inserting symbols, in this case arrows for quick reference.

Key Steps

Figure 2.1, the “BEFORE” version, is transformed into Figure 2.2, the “AFTER” version, by making the following modifications:

1 Region is displayed as a line above each report page, rather than as a column on each page.

2 Store #s and Branch #s are displayed in bold blue font (font color (or text color) is called “foreground” in the SAS® code).

3 Arrows are displayed after each 2011 Sales column to provide a quick reference to whether sales increased or decreased, and open circles are displayed to indicate that sales stayed the same from the previous quarter.

- Color is applied to the arrows to further highlight increases (green), decreases (red), and no change (black).

4 A spanning header, “2011 Sales” is added and bottom borders and underlines are added to header cells.

5 Summary lines, which show total Store sales, are brought out by applying blue font and adding a blank line after each summary line, before the start of each new store number.

6 Blank columns are added after the following columns: “Branch #”, “2010 Sales Quarter 4”, and after “Vs Q4”.

Each modification is explained in a later section of this paper.

Source Data

There is one source data set, Sales, which contains quarterly sales totals by Region, Branch Number, and Store Number. Table 2.1 displays the variable information for SALES. Table 2.2 contains a sample of the Sales data so the user can visualize the structure of the data set.

Table 2.1 Sales Variable Information

Variables in Creation Order

#	Variable	Type	Len	Label
1	REGION	Char	8	Region
2	STORE	Char	8	Store #
3	BRANCH	Num	8	Branch #
4	QTR4_2010	Num	8	Quarter 4 2010 Sales
5	QTR1	Num	8	Quarter 1 2011 Sales
6	QTR2	Num	8	Quarter 2 2011 Sales

Table 2.2 Sample Sales Data

REGION	STORE	BRANCH	QTR4_2010	QTR1	QTR2
East	1001	1	43564	42555	47000
East	1001	2	47235	47523	48102
East	1001	3	50244	45999	46582
East	1001	4	37665	37778	39624
East	1002	1	50000	50248	49335
East	1002	2	48045	46822	48000

ODS Style Template Used

The same ODS Style template, named “RSTYLERTF” was used to produce both Figure 2.1 and Figure 2.2 reports. This is a user-created template, meaning YOU need to create it with PROC TEMPLATE. Note that the template RSTYLERTF is later referenced in the ODS statement prior to running the REPORT procedure (see Program 2.1’s ODS statement starting with, “**ods rtf style=RSTYLERTF…”** in the Programs Used section).

To reproduce Figure 2.1 or Figure 2.2 you will need to to run the PROC TEMPLATE Program shown in PROGRAMS USED section.

Programs Used

The PROC TEMPLATE program is run first to create the ODS style template, which provides the base formatting for both Figure 2.1 and Figure 2.2.

The “Before Formatting” program (referred to as Program 2.1 for the remainder of this paper) was used to create Figure 2.1. Both the PROC TEMPLATE program and Progam 2.1 are displayed in this paper.

The final program, called Format.sas, was used to create Figure 2.2. Format.sas can be downloaded from the author’s web page at http://support.sas.com/publishing/authors/fine.html.

This paper was written based on SAS 9.3. The author’s web page provides some SAS9.1.3 code workarounds for the examples that use features from later SAS releases.

PROC TEMPLATE Program to Create New Style Template

We create a new ODS style template because we want to slightly alter the default RTF ODS style template prior to producing the reports. We name the new template RSTYLERTF with the DEFINE STYLE statement. The PARENT= statement specifies that RSTYLERTF should inherit the style elements from STYLES.RTF.

We make the desired style element modifications for our new template via CLASS statements. Specifically, we remove some of the table and cell borders, thicken the table's top border, remove the background color from the header cell, and modify the page margins.

```
** PROC Template Program;
proc template;
  define style rstyleRTF;
    parent=styles.rtf;
  class table/
    rules=none
    frame=above
    borderwidth=1.5 pt
    cellpadding=0;
  class Header/
    background=_undef_;
  class body/
    bottommargin = .75 in
    topmargin    = 1.25 in
    rightmargin  = .75 in
    leftmargin   = .75 in;
end;
run;
```

The RSTYLERTF template will inherit the style elements from the PARENT template (STYLES.RTF) except the table, header and body elements specified by the CLASS statements.

Table 2.3 demonstrates some of the PROC TEMPLATE changes.

Table 2.3 PROC TEMPLATE Table and Header Changes

RTF DEFAULT ❶ →

Region
East
West
North
South

TABLE Change:
Rules=None ❷ →

Region
East
West
North
South

TABLE Change:
Frame=Above and Borderwidth=1.5 pt ❸ →

Region
East
West
North
South

TABLE Change:
Cellpadding=0 ❹ →

Region
East
West
North
South

HEADER Change:
Background=_undef_ ❺

Region
East
West
North
South

❶ This shows the default RTF ODS Style Template (the PARENT of the new ODS Style Template, RSTYLERTF).

❷ Rules=None requests that there are no border lines between table cells.

❸ Only the above portion of the table frame is requested. This above border is widened (made thicker) to 1.5 pt.

❹ The cell padding is set to 0 to remove the padding around the text within the cells. This has the effect of reducing the row heights.

❺ The background of the "Region" header is specified as undefined to remove the default background color.

Recommended books for more information on PROC TEMPLATE and ODS Style Templates include:

Haworth, Lauren E., Zender Cynthia L., and Burlew, Michelle M. 2009. Output Delivery System: The Basics and Beyond. Cary, NC: SAS Institute Inc.

Smith, Kevin D. 2013. PROC TEMPLATE Made Easy: A Guide for SAS® Users. Cary, NC: SAS Institute Inc.

The "Before Formatting" Program (Program 2.1)

Figure 2.1, the "BEFORE" figure was generated from the following program, after running the PROC TEMPLATE program.

```
** The "Before Formatting" Program (Program 2.1);
** Get Data;
data sales;
```

```
   set sasuser.sales;
run;

** Produce the Report;
ods _all_ close;
ods escapechar = "^";
options nodate nonumber orientation=portrait;
ods rtf style=RSTYLERTF file="C:\client\projectABC\UnFmt.rtf";

proc report data=sales nowd split="|" center missing
  style(column)=[just=c cellwidth=.8 in] out=prefmt;
column REGION STORE BRANCH QTR4_2010 QTR1 QTR2;
  define REGION / order "Region"   style(column)=[cellwidth=.65 in];
  define STORE  / order "Store #"  style(column)=[cellwidth=.65 in];
  define BRANCH / order "Branch #" style(column)=[cellwidth=.7 in];
  define QTR4_2010 / "Quarter 4 2010" format=dollar10.;
  define QTR1 / "Quarter 1 2011" format=dollar10.;
  define QTR2 / "Quarter 2 2011" format=dollar10.;

  ** Sum Sales per store;
  break after STORE   / summarize suppress;
run;

ods rtf close;
ods html;
title;
footnote;
```

For reference, Table 2.4 shows a partial print of the PROC REPORT output data set created in the PROC REPORT statement via “OUT=PREFMT. You can see that an extra column named _BREAK_ was produced. This automatic temporary variable would be created regardless of whether we had the BREAK statement, however the _BREAK_ column would contain only blank values.

The “BREAK AFTER STORE /” statement by itself creates extra rows for which the value of _BREAK_ is “STORE” and the numeric columns, QTR4_2010, QTR1, QTR2 are summed.

The addition of the SUMMARIZE option allows the summary rows to be displayed in the printed report. The SUPPRESS option prevents the REGION and STORE values from being displayed on the summary lines of the printed report. You can see that the REGION and STORE values do still remain in the summary lines of the PROC REPORT data set.

Table 2.4 PROC REPORT Data Set "PREFMT"

REGION	STORE	BRANCH	QTR4_2010	QTR1	QTR2	_BREAK_
East	1001	1	43564	42555	47000	
East	1001	2	47235	47523	48102	
East	1001	3	50244	45999	46582	
East	1001	4	37665	37778	39624	
East	1001	.	178708	173855	181308	STORE
East	1002	1	50000	50248	49335	
East	1002	2	48045	46822	48000	
East	1002	3	60023	62410	62650	
East	1002	4	55000	56102	56109	
East	1002	5	52345	52650	54197	
East	1002	6	56444	57720	57800	
East	1002	7	59721	63000	65254	
East	1002	.	381578	388952	393345	STORE
East	1003	1	67443	66230	67503	
East	1003	2	69903	70085	73598	
East	1003	3	57900	60000	62900	
East	1003	4	58989	59222	58307	
East	1003	5	56112	57364	57647	
East	1003	.	310347	312901	319955	STORE
East	1007	1	51008	51008	52310	
East	1007	2	46733	43062	45564	
East	1007	3	61228	59838	62300	
East	1007	.	158969	153908	160174	STORE
East	1008	1	63328	63000	64274	
East	1008	2	61377	66461	70468	
East	1008	3	59821	59637	61272	
East	1008	.	184526	189098	196014	STORE

Implementation: Transforming Figure 2.1 Into Figure 2.2

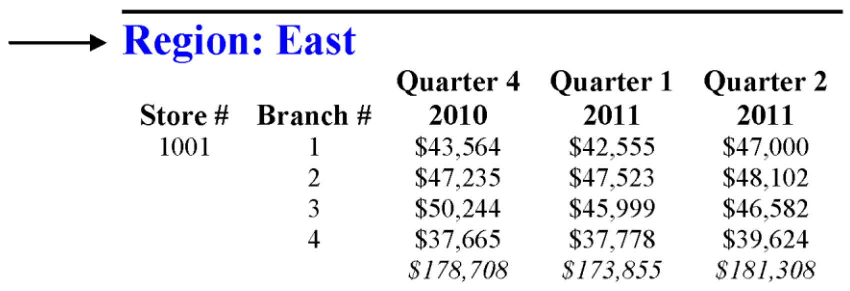
The remainder of the paper will add the code that transforms Figure 2.1. to Figure 2.2, one section at a time.

.

Displaying Region as a Line Above Each Report Page

As shown in Figure 2.3, the report displays Region as a line rather than as a report column.

Figure 2.3 - Display Region Before Report



Region: East

Store #	Branch #	Quarter 4 2010	Quarter 1 2011	Quarter 2 2011
1001	1	$43,564	$42,555	$47,000
	2	$47,235	$47,523	$48,102
	3	$50,244	$45,999	$46,582
	4	$37,665	$37,778	$39,624
		$178,708	*$173,855*	*$181,308*

Overview of Region Display

Displaying REGION as a line above each report page is accomplished with a BREAK statement and a COMPUTE block. A change in REGION is the designated BREAK before which a new page should be started. “Before” each page is the location for the COMPUTED region LINE.

Key steps for displaying a Region line above each report page include

- Specify ORDER and NOPRINT options in the DEFINE statement for REGION.
- Insert a page break before each new value of REGION using a BREAK statement.
- Via a COMPUTE block, display the region specification before each page’s report in the form “Region: “ *REGION value*, and apply styles including left justification, bolded font, increased font size and blue foreground to the line.

Code to Make the Region Display in Figure 2.3

The following additions are made to Program 2.1 to display the region as a line.

```
** Specify ORDER and NOPRINT Options for REGION; ❶
define REGION / order noprint;

** Insert Page Break Before each Region;
break before REGION / page; ❷

** Display the Region Before each Page and Apply Styles;
compute before _page_; ❸
  line "^S={foreground=blue font_weight=bold font_size=16 pt just=left}
        Region: " REGION $20.; ❹
endcomp;
```

❶ While REGION is a report variable specified in the column statement, the goal is to print REGION in a line prior to the report, rather than as a column. Therefore the NOPRINT option is applied in the DEFINE statement to prevent the column printing. The ORDER option is applied to allow for page BREAKs and COMPUTEd lines BEFORE REGION.

❷ To ensure that only one region’s data appears on each page, a BREAK statement is used to start a new page for each new value of REGION.

❸ After ensuring each page contains only one region, the location for lines displaying region text is specified as “before _page_”, which translates into “Display the region above the table on each page.”

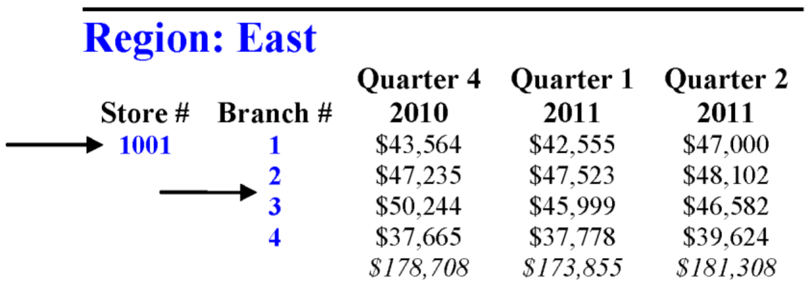
❹ Specifically, the region text should appear in the form "Region: " *Actual REGION value*.

The font weight is set to bold, the font size is increased to 16 pt, and the font color is changed from the default black to blue, to further highlight the region being reported. The line is also left justified.

Displaying Store and Branch Column Data in Bold Blue Font

Figure 2.4 shows the application of color and bold weight to STORE and BRANCH data. The style modifications are applied in these variables' DEFINE statements.

Figure 2.4 – Blue Font: Store and Branch Values



Region: East

Store #	Branch #	Quarter 4 2010	Quarter 1 2011	Quarter 2 2011
1001	1	$43,564	$42,555	$47,000
	2	$47,235	$47,523	$48,102
	3	$50,244	$45,999	$46,582
	4	$37,665	$37,778	$39,624
		$178,708	*$173,855*	*$181,308*

Code for Store and Branch Display

The highlighted code indicates the additions to the Program 2.1 STORE and BRANCH DEFINE statements.

```
** Modify Foreground and Font_weight for STORE and BRANCH Data; ❶
define STORE  / order "Store #" style(column)=[cellwidth=.65 in
                                foreground=blue font_weight=bold];
define BRANCH / order "Branch #" style(column)=[cellwidth=.7 in
                                foreground=blue font_weight=bold];
```

❶ The columns are styled using the STYLE(COLUMN)= option, along with the CELLWIDTH=, FOREGROUND=, and FONT_WEIGHT= attribute specifications on the respective DEFINE statements. The CELLWIDTH= style attribute is used to define the STORE and BRANCH column sizes as .65 and .7 inches wide. The blue colored font ("foreground") and bold font weight are applied to STORE and BRANCH values with the FOREGROUND= and FONT_WEIGHT= specifications.

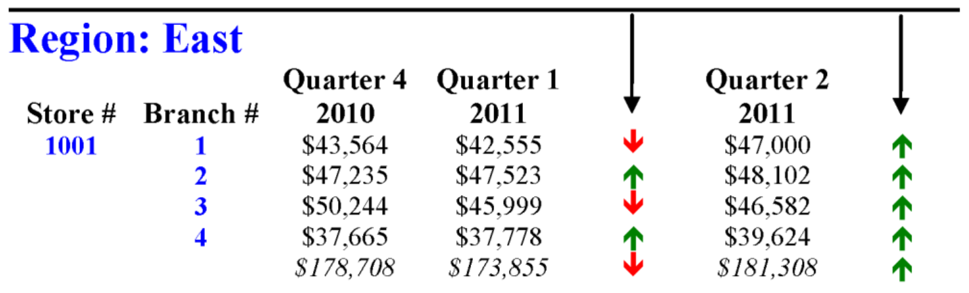
Note that the ORDER option is applied to both STORE and BRANCH (as was done for the unformatted report, Figure 2.1).

- Applying the ORDER usage option orders the stores and branches, prevents repeated values from being displayed and, importantly, prevents the numeric branch numbers from being assigned their default ANALYSIS. With an ANALYSIS usage, the branch numbers would be summed by the BREAK after the store / summarize statement.

How to Insert Arrows for Quick Reference to Sales Increases/Decreases

Increases/Decreases Arrows have been inserted so the reader can quickly identify if sales increased, decreased, or stayed the same from the previous quarter, as shown in Figure 2.5.

Figure 2.5 – Insert Arrows for Quick Reference



Region: East

Store #	Branch #	Quarter 4 2010	Quarter 1 2011	↓	Quarter 2 2011	↓
1001	1	$43,564	$42,555	↓	$47,000	↑
	2	$47,235	$47,523	↑	$48,102	↑
	3	$50,244	$45,999	↓	$46,582	↑
	4	$37,665	$37,778	↑	$39,624	↑
		$178,708	*$173,855*	↓	*$181,308*	↑

Overview on Arrow Insertion

In this section COMPUTE blocks are used to create new variables that represent the change in sales from quarter to quarter. The numeric differences are converted to arrows via a format. Color and font style is then applied to the arrows, including the summary line, via a color format used in CALL DEFINE statements. The key steps for inserting arrows to show changes from the previous quarter's sales include

- Create the arrow and color formats to be applied to the sales differences.
- Add the COMPUTEd variable names DIR1 and DIR2 to the COLUMN Statement.
- Apply the Wingdings font_face and arrow format in the DIR1 and DIR2 DEFINE Statements.
- Using a COMPUTE Block:
  - Create the two difference variables DIR1 and DIR2.
    - DIR1 represents Quarter 1 2011 sales (QTR1) minus Quarter 4 2010 sales (QTR4_2010).
    - DIR2 represents Quarter 2 2011sales (QTR2) minus Quarter 1 2011 sales (QTR1).
  - Conditionally apply color to increases, decreases, and no change in CALL DEFINE statements so the summary line arrows receive the proper color application.

Code for Arrow Insertion

The following additions are made to Program 2.1 to create DIR1 and DIR2 and style these as symbols.

```
** Create Color and Arrow Formats; ❶
proc format;
  value arrow
    low - < 0 = “EA”x
    0 = “A2”x
    0 <- high = “E9”x;

  value color
    low - < 0 = red
    0 = black
    0 <- high = green;
run;
```

```
** Add Computed Variable Names DIR1 and DIR2 to the COLUMN Statement; ❷
proc report data=sales nowd split=‶|″ center missing
                       style(column)=[just=c cellwidth=.8 in];
  columns REGION STORE BRANCH QTR4_2010 QTR1 DIR1 QTR2 DIR2;

** Add DIR1 and DIR2 DEFINE Statements. Apply Arrow Format and Wingdings
   Font; ❸
  define DIR1 / " " computed format=arrow.
                    style(column)=[cellwidth=.6 in font_face=wingdings];
  define DIR2 / " " computed format=arrow.
                    style(column)=[cellwidth=.6 in font_face=wingdings];

** Add Compute Blocks to Derive Difference Variables DIR1 And DIR2 and Style
   with Color. Format, including Summary Lines;
compute DIR1;
   DIR1=QTR1.sum-QTR4_2010.sum; ❹
   call define(‶DIR1″,″style″,″style={foreground=color.}″); ❺
endcomp;
compute DIR2;
   dir2=QTR2.sum-QTR1.sum;
   call define(‶DIR2″,″style″,″style={foreground=color.}″);
endcomp;
```

❶ Two formats are created. The arrow format contains ASCII hexadecimal (hex) codes (“EA”,”E9”,”A2”) that will be rendered as arrows and circles once Wingdings font is applied in the DEFINE statements for the sales difference variables DIR1 and DIR2. Table 2.5 displays the Hex codes, and how they will be rendered in the RTF output after Wingdings font is applied.

Table 2.5 Arrow Format Specifying Hex Code and Corresponding Wingdings Representation

Hex Code	Wingdings Representation
A2	○
E9	↑
EA	↓

For other example conversions of Hex Code to Wingdings go to http://dmcritchie.mvps.org/rexx/htm/fonts.htm

The COLOR format will be used to apply either Red, Black, or Green to DIR1 and DIR2 in the CALL DEFINE statements.

❷ The computed variables are added to the COLUMN statement since they must be report variables to show as columns.

❸ While the arrow format could have been applied in the conditional CALL DEFINE blocks, they are applied in the DEFINE statements since all rows of the difference variables use the arrow format.

Based on the arrow format, negative sales differences are applied the “EA”x value (down arrow), 0s are applied the “A2”x value (open circle), and positive difference are applied the “E9”x value (up arrow). If the format is not applied windings symbols other than the desired ones will be displayed in the report.

❹ Per COMPUTE block rules, the analysis variables used to derive the computed variables are referenced by their compound names (*VARIABLE.statistic*) such as QTR1.sum and QTR4_2010.sum (sum is the default statistic for analysis variables).

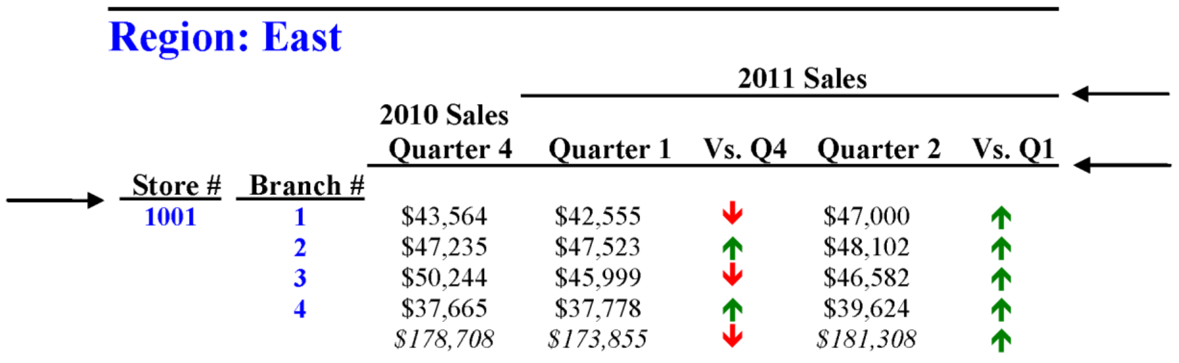
❺ The CALL DEFINE statement is used to conditionally apply font color to the arrows and open circle using the COLOR format. The color format was applied within the COMPUTE block CALL DEFINE statement (versus in DIR1 and DIR2 DEFINE statements), so it would be applied to the summary line arrows as well (otherwise the summary line arrows display in blue colored font).

Increases and decreases use Wingdings font (to obtain the arrow symbol) and green versus red foregrounds respectively. For no change (e.g. DIR1 or DIR2 = 0), a black open circle is shown.

How to Add Spanning Headers, Bottom Cell Borders and Underlines

Figure 2.6 shows the addition of the “2011 Sales” header that spans over the 2011 quarterly sales columns and their corresponding arrows columns. Bottom cell borders and underlines are also added to help clarify sections of the report.

Figure 2.6 – Bottom Cell Borders and Text Underlines



Region: East

		2010 Sales	2011 Sales			
Store #	Branch #	Quarter 4	Quarter 1	Vs. Q4	Quarter 2	Vs. Q1
1001	1	$43,564	$42,555	↓	$47,000	↑
	2	$47,235	$47,523	↑	$48,102	↑
	3	$50,244	$45,999	↓	$46,582	↑
	4	$37,665	$37,778	↑	$39,624	↑
		$178,708	*$173,855*	↓	*$181,308*	↑

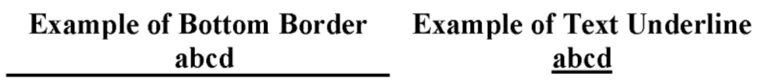
Highlights on Adding Spanning Headers, Borders and Underlines

The Sales column headers are created in the COLUMN statement (versus DEFINE statements) to ensure

1. to ability to create spanning headers, and
2. these headers are displayed a level above the “Store #” and “Branch #” headers.

This section makes use of border control to add header bottom borders, and text decoration functionality to add header underlines. The example below shows the difference between a bottom border and a text underline.

Comparison of Bottom Border and Text Underline



Example of Bottom Border
abcd

Example of Text Underline
abcd

Bottom borders are added by inserting the inline formatting function borderbottomwidth as shown below.

`"^{style [borderbottomwidth=1pt]` *header text*`}"`

Text Underlines are added by inserting the inline formatting function textdecoration as shown below

`"^S={textdecoration=underline}` *header text*`"`

Specific steps in creating spanning headers include

- In the COLUMN statement, add bottom cell borders and create a spanning header, “2011 Sales” to span over relevant columns.
- Add “Store #” and “Branch #” headers via the DEFINE statements so they are displayed a level below the sales headers. For these headers, text underlining is used rather than applying bottom borders to the table cells.
- Set the Sales variables’ header text to null in the DEFINE statements since these headers are being labeled in the COLUMN statements.

Code for Adding Spanning Headers, Borders and Underlines

The following additions are made to the Program 2.1 COLUMN and DEFINE statements to modify the column headers.

```
COLUMN region store branch
** Add Bottom Cell Borders And Create A Spanning Header;
   ("2010 Sales|^{style [borderbottomwidth=1 pt]Quarter 4}" QTR4_2010) ❶
   ("^{style [borderbottomwidth=1 pt] 2011 Sales}" ❷
       ("^{style [borderbottomwidth=1 pt] Quarter 1}" QTR1) ❸
       ("^{style [borderbottomwidth=1 pt] Vs. Q4}"    dir1)
       ("^{style [borderbottomwidth=1 pt] Quarter 2}" QTR2)
       ("^{style [borderbottomwidth=1 pt] Vs. Q1}"    dir2)
    );
** Underline Store # and Branch # Headers; ❹
define store  / order "^S={textdecoration=underline}Store #"
                style(column)=[cellwidth=.65 in
                foreground=blue font_weight=bold];
define branch / order "^S={textdecoration=underline}Branch #"
                style(column)=[cellwidth=.7 in
                foreground=blue font_weight=bold];
** Sales Column Headers are created in the COLUMN statement and set to null
in DEFINE statements so variable names are not displayed; ❺
define QTR4_2010 / "" format=dollar10.;
define QTR1      / "" format=dollar10.;
define DIR1      / computed  "" style(column)=[font_face=wingdings
                                     cellwidth=.6 in] format=arrow.;
define QTR2      / "" format=dollar10.;
define DIR2      / computed  "" style(column)=[font_face=wingdings
                                     cellwidth=.6 in] format=arrow.;
```

❶ 2010 Sales Quarter 4 is a standalone item that is given a cell bottom border.

❷ A spanning header “2011 Sales” with a bottom border is created to encompass each 2011 Sales and each Increase/Decrease column, “Quarter 1”, “Vs. Q4”, “Quarter 2”, and “Vs. Q1”.

The "^{style [borderbottomwidth=1 pt] 2011 Sales}" code covers all columns within the outermost begin and end parentheses (i.e. encompasses all of the “❸” boxed code section).

❸ Within this spanning header, each individual 2011 Sales header is given a bottom border as well.

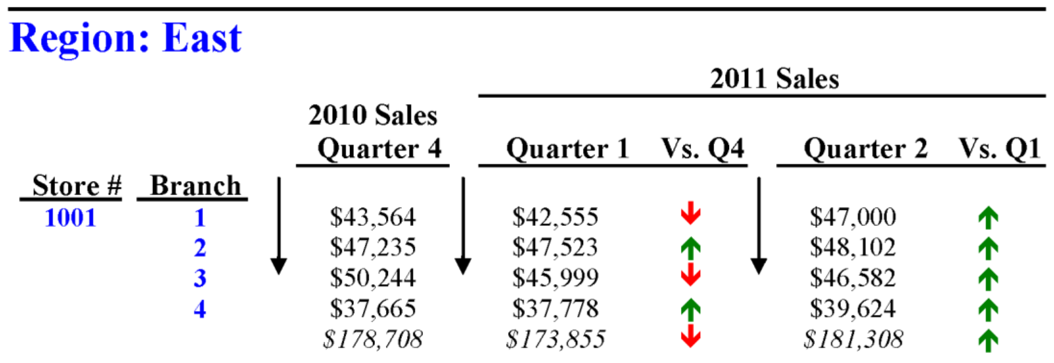
❹ The underlines for Store # and Branch # headers are applied by inserting the textdecoration inline formatting within the headers in the DEFINE statements. These headers could have been created in the COLUMN statement but the preference for this report is to display these headers below the Sales header levels.

❺ Sales Column Headers are specified in the COLUMN statement and are therefore set to null (see gray highlighted code) in the DEFINE statements so the variable names are not displayed.

Adding Blank Columns to Make the Report Easier to Read

Figure 2.7 presents the report with added blank columns where white space is desired.

Figure 2.7 – Showing Added Blank Columns



Region: East

			2010 Sales		2011 Sales			2011 Sales	
Store #	Branch		Quarter 4		Quarter 1	Vs. Q4		Quarter 2	Vs. Q1
1001	1	↓	$43,564	↓	$42,555	↓	↓	$47,000	↑
	2		$47,235		$47,523	↑		$48,102	↑
	3		$50,244		$45,999	↓		$46,582	↑
	4		$37,665		$37,778	↑		$39,624	↑
			$178,708		*$173,855*	↓		*$181,308*	↑

Overview of Adding Blank Columns

A DATA step is used to derive the new variable BLANK, which is set to null and specified as a PROC REPORT COLUMN variable in places blank columns are desired.

- Create "BLANK" variable in data set.
- Add the new variable to the COLUMN statement in places for which a blank column is desired.
- Specify null label and the desired cell width for each blank column in the DEFINE statement.

Code for Adding Blank Columns

```
** Add "BLANK" variable to the SALES data set; ❶
data sales;
  set sales;
  blank = "";
run;
** Add BLANK to COLUMN Statement in Desired Locations; ❷
COLUMN region store branch BLANK
** Add Bottom Cell Borders And Create A Spanning Header;
   ("2010 Sales|^{style [borderbottomwidth=1 pt]Quarter 4}" QTR4_2010) BLANK
   ("^{style [borderbottomwidth=1pt] 2011 Sales}"
       ("^{style [borderbottomwidth=1pt] Quarter 1}" QTR1)
       ("^{style [borderbottomwidth=1pt] Vs. Q4}"    dir1) BLANK
       ("^{style [borderbottomwidth=1pt] Quarter 2}" QTR2)
       ("^{style [borderbottomwidth=1pt] Vs. Q1}"    dir2)
    );
** Add BLANK DEFINE statement with null header and desired column width; ❸
define BLANK / "" style(column)=[cellwidth=.15 in];
```

❶ BLANK variable is added to the dataset to allow us to insert white space in the report where extra blank columns are desired.

❷ The COLUMN statement determines the column order in the report. Therefore the BLANK variable is listed each time a blank column is desired within the COLUMN variable list. We use the BLANK variable three times in this report.

❸ If the labels are not set to null the heading "BLANK" will appear as a column header in the report. If the cell width is not set the default column width may produce a blank column wider/narrower than desired.

Note that the BLANK columns could have been created via COMPUTE blocks. For example, we could have used the following code to create three new report variables.

```
compute BLANK1 / char length=2;
  BLANK1 = " ";
endcomp;

compute BLANK2 / char length=2;
  BLANK2= " ";
endcomp;

compute BLANK3 / char length=2;
  BLANK3 = " ";
endcomp;
```

We would then reference the variable names BLANK1, BLANK2, and BLANK3 in the COLUMN statement, and use three DEFINE statements to apply null headers and desired cell widths.

Rather, for this example we took advantage of the automatic creation of aliases for the input data set variable BLANK that occurred because we referenced it more than once in the COLUMN statement. We can see that this automatic creation occurred by adding the LIST option to the PROC REPORT statement. (LIST writes PROC REPORT code to the log, including some of the defaults we did not specify).

Using the LIST option in the PROC REPORT statement, the SAS log shows us that aliases _A1 and _A2 were created for BLANK.

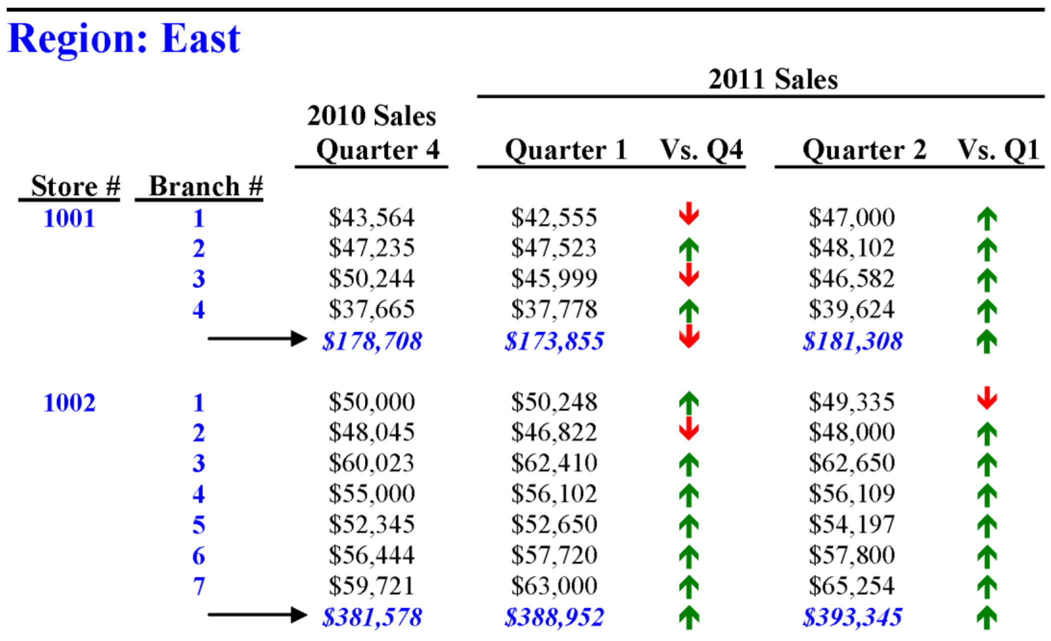
```
PROC REPORT DATA=WORK.SALES LS=98  PS=55  SPLIT="|" CENTER MISSING ;
COLUMN  ( REGION STORE BRANCH
 blank
 ("2010 Sales|^{style [borderbottomwidth=1 pt]Quarter 4}" qtr4_2010 )
 blank=_A1
 ("^{style [borderbottomwidth=1pt] 2011 Sales}"
  ("^{style [borderbottomwidth=1pt] Quarter 1}"  qtr1 )
  ("^{style [borderbottomwidth=1pt] Vs. Q4}"dir1 )
 blank=_A2
  ("^{style [borderbottomwidth=1pt] Quarter 2}"  qtr2 )
  ("^{style [borderbottomwidth=1pt] Vs. Q1}"  dir2 )
 ) );
```

Also note, using this method for obtaining blank columns only required one DEFINE statement for the variable BLANK.

Style / Add Blank Line After each Summary Line

As a final step, summary lines are distinguished by adding color and white space as shown in Figure 2.8.

Figure 2.8 Style Summary Lines



Region: East

			2011 Sales			
Store #	Branch #	2010 Sales Quarter 4	Quarter 1	Vs. Q4	Quarter 2	Vs. Q1
1001	1	$43,564	$42,555	↓	$47,000	↑
	2	$47,235	$47,523	↑	$48,102	↑
	3	$50,244	$45,999	↓	$46,582	↑
	4	$37,665	$37,778	↑	$39,624	↑
	→	*$178,708*	*$173,855*	↓	*$181,308*	↑
1002	1	$50,000	$50,248	↑	$49,335	↓
	2	$48,045	$46,822	↓	$48,000	↑
	3	$60,023	$62,410	↑	$62,650	↑
	4	$55,000	$56,102	↑	$56,109	↑
	5	$52,345	$52,650	↑	$54,197	↑
	6	$56,444	$57,720	↑	$57,800	↑
	7	$59,721	$63,000	↑	$65,254	↑
	→	*$381,578*	*$388,952*	↑	*$393,345*	↑

Highlights on Styling Summary Line and Adding Blank Line

Key steps to highlighting sales totals include

- Summarize with a BREAK statement (this was done for the unformatted report as well). In this example we apply bold blue font to the summary lines.
- Further highlight the totals by adding a blank line after the last value of each Store #.

Code for Styling Summary Line and Adding Blank Line

```
** Apply Bold Blue Font to Summary Lines;
break after STORE   / summarize suppress style(summary)=[font_weight=bold
foreground=blue]; ❶

** Add Blank Line after each store # section ❷;
compute after STORE;
  line " ";
endcomp;
```

❶ Specify that PROC REPORT summarize the analysis variables after STORE (i.e. each store number). BRANCH, a numeric variable would be an analysis variable by default, and therefore summarized, but the ORDER option overrides this.

We specify desired styles for the summary line with the style(summary)= option.

❷ Via a COMPUTE block add a blank line after each STORE.

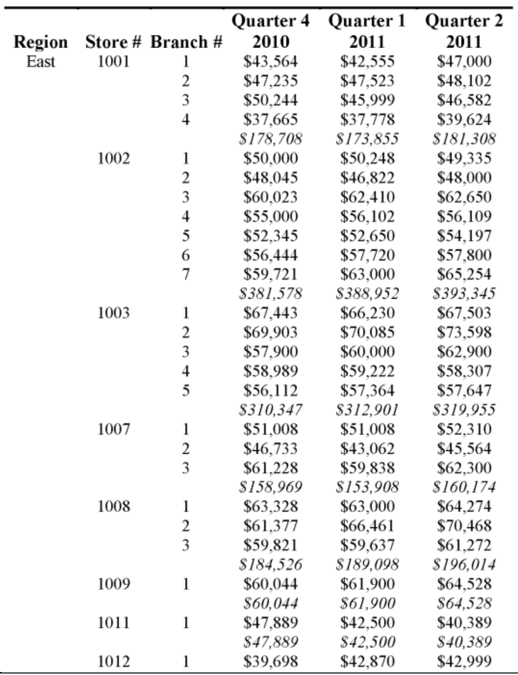
Recap

The paper demonstrated ways to turn a highly detailed report (Figure 2.1) into a manageable readable format (Figure 2.2) simply by applying various ODS and PROC REPORT features. Table 2.6 recaps the enhancements made to Figure 2.2. and the main programming modifications that were required.

Table 2.6 Recap

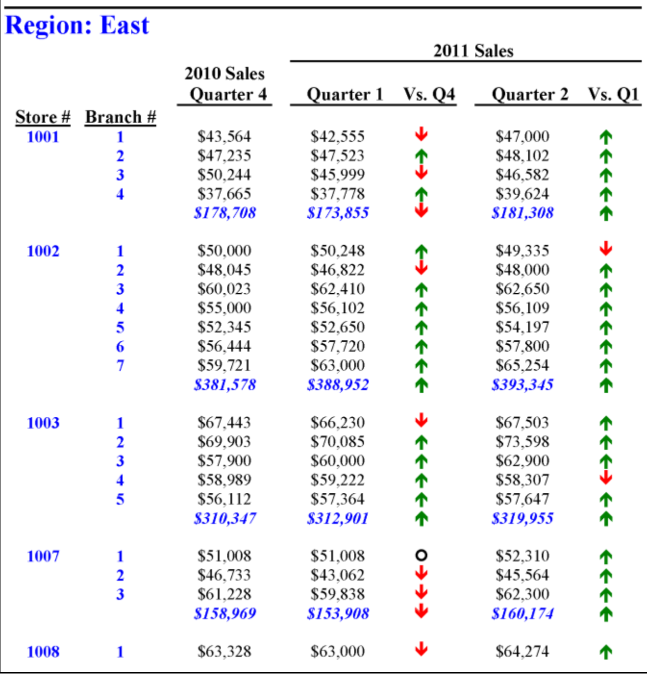
Enhancements	Main Programming Modifications
◆ Region was displayed as a line above each report page, rather than as a column on each page.	Specifying NOPRINT in the REGION DEFINE statement; BREAK Statement to create page break (before REGION / page), COMPUTE Block to create a Region line (before _page_).
◆ Store #s and Branch #s were displayed in bold blue font (“foreground”).	Styled with Style(column) = options in DEFINE statements.
◆ Arrows were displayed after each 2011 Sales column to provide a quick reference to whether sales increased, decreased, or stayed the same from the previous quarter.	COMPUTED variables to derive quarterly sales differences; Formats were used to apply symbol type and color to the difference variables. Color was applied in CALL DEFINE statements to override the blue colored font of Summary Line symbols.
◆ A spanning header, “2011 Sales” was created and bottom borders and underlines were added to header cells.	COLUMN statement used to create spanning and nested headers with the use of parentheses. Inline formatting style functions were inserted to add borders and underlines.
◆ Summary lines, which show total Store sales, were brought out by applying blue font color and adding a blank line after each summary line, before the start of each new store number.	Summary lines were styled with the style(summary)= option; A COMPUTE block was used to add a blank line after each STORE.
◆ Blank columns were added after the following columns: Branch #, 2010 Sales Quarter 4, and Vs Q4.	A DATA step was used to create a new variable, BLANK. This was reported in desired locations by adding BLANK to the PROC REPORT COLUMN statement where blanks were desired. The desired cellwidth and a null header were applied in a DEFINE statement.

Figure 2.1 BEFORE Formatting



Region	Store #	Branch #	Quarter 4 2010	Quarter 1 2011	Quarter 2 2011
East	1001	1	$43,564	$42,555	$47,000
		2	$47,235	$47,523	$48,102
		3	$50,244	$45,999	$46,582
		4	$37,665	$37,778	$39,624
			$178,708	*$173,855*	*$181,308*
	1002	1	$50,000	$50,248	$49,335
		2	$48,045	$46,822	$48,000
		3	$60,023	$62,410	$62,650
		4	$55,000	$56,102	$56,109
		5	$52,345	$52,650	$54,197
		6	$56,444	$57,720	$57,800
		7	$59,721	$63,000	$65,254
			$381,578	*$388,952*	*$393,345*
	1003	1	$67,443	$66,230	$67,503
		2	$69,903	$70,085	$73,598
		3	$57,900	$60,000	$62,900
		4	$58,989	$59,222	$58,307
		5	$56,112	$57,364	$57,647
			$310,347	*$312,901*	*$319,955*
	1007	1	$51,008	$51,008	$52,310
		2	$46,733	$43,062	$45,564
		3	$61,228	$59,838	$62,300
			$158,969	*$153,908*	*$160,174*
	1008	1	$63,328	$63,000	$64,274
		2	$61,377	$66,461	$70,468
		3	$59,821	$59,637	$61,272
			$184,526	*$189,098*	*$196,014*
	1009	1	$60,044	$61,900	$64,528
			$60,044	*$61,900*	*$64,528*
	1011	1	$47,889	$42,500	$40,389
			$47,889	*$42,500*	*$40,389*
	1012	1	$39,698	$42,870	$42,999

Figure 2.2 AFTER Formatting



Region: East

		2010 Sales	2011 Sales		2011 Sales	
Store #	Branch #	Quarter 4	Quarter 1	Vs. Q4	Quarter 2	Vs. Q1
1001	1	$43,564	$42,555	↓	$47,000	↑
	2	$47,235	$47,523	↑	$48,102	↑
	3	$50,244	$45,999	↓	$46,582	↑
	4	$37,665	$37,778	↑	$39,624	↑
		$178,708	***$173,855***	↓	***$181,308***	↑
1002	1	$50,000	$50,248	↑	$49,335	↓
	2	$48,045	$46,822	↓	$48,000	↑
	3	$60,023	$62,410	↑	$62,650	↑
	4	$55,000	$56,102	↑	$56,109	↑
	5	$52,345	$52,650	↑	$54,197	↑
	6	$56,444	$57,720	↑	$57,800	↑
	7	$59,721	$63,000	↑	$65,254	↑
		$381,578	***$388,952***	↑	***$393,345***	↑
1003	1	$67,443	$66,230	↓	$67,503	↑
	2	$69,903	$70,085	↑	$73,598	↑
	3	$57,900	$60,000	↑	$62,900	↑
	4	$58,989	$59,222	↑	$58,307	↓
	5	$56,112	$57,364	↑	$57,647	↑
		$310,347	***$312,901***	↑	***$319,955***	↑
1007	1	$51,008	$51,008	○	$52,310	↑
	2	$46,733	$43,062	↓	$45,564	↑
	3	$61,228	$59,838	↓	$62,300	↑
		$158,969	***$153,908***	↓	***$160,174***	↑
1008	1	$63,328	$63,000	↓	$64,274	↑

References

Fine, Lisa. 2013. PROC REPORT by Example: Techniques for Building Professional Reports Using SAS. Cary, NC: SAS Institute Inc.

(Many excellent sources went into writing PROC REPORT by Example. These sources are cited in the book's "References" section)

Acknowledgments

What started out as a paper ended up becoming a book thanks to the encouragement of Michael Raithel and Cynthia Zender. The content and writing was then greatly improved by insightful feedback from my book reviewers, Allison Booth, Robin Crumpton, Mei Du, Jane Eslinger, Tim Hunter, Bari Lawhorn, Russell Lavery, and Matt Nizol.

Contact Information

Your comments and questions are valued and encouraged. Contact the author:

Lisa Fine

United BioSource Corporation

E-mail: Lisa.Fine@UBC.com

Trademark Citations