MWSUG 2019 HW094

Not Even One Single Solitary Semicolon:

Powerful SAS Things You Can Do Without Writing Programs

Ben Cochran, The Bedford Group, Raleigh, NC

ABSTRACT:

This presentation starts by illustrating the conversion of different kinds of data into SAS data sets. Specifically, Excel spreadsheets and Microsoft Access tables are converted into SAS data. Then, these two different data sources are joined with an existing SAS data set. Finally, a series of graphical and tabular reports are generated from this combined data. All of these tasks are completed without writing any SAS programs

INTRODUCTION:

Could you be more efficient and get your SAS applications done so much faster if you could point and click you way along instead of writing programs? Of course you could. And, as an added bonus, you don't have to worry about spelling keywords correctly or worry about where to put the semicolons. A few years ago, I was working on a consulting project with an organization that had just installed SAS® Enterprise Guide (EG). This was a financial organization and their people were financial analysts, not SAS programmers. They knew how to make Microsoft Excel do cartwheels, but never heard of a DATA step. I arrived on site less than 2 weeks after EG was installed, and was amazed at how quickly they picked up where to point and click, and where to drag and drop. Most users were moving their mouse so fast, their hands turned into a blur. Needless to say, I was amazed at how quickly they learned how to operate within this SAS environment known as EG. This hands-on-workshop will show you how to do a number of very useful things very quickly. If you have not already done so, you will be amazed at how quickly you can accomplish

and master a multitude of tasks within Enterprise Guide. This paper is divided into x sections. The first one examines ways to access different kinds of data by simply using your mouse.

1. ACCESSING MICROSOFT ACCESS DATA:

This presentation follows the daily task of a fictitious Insurance company. They need to process massive amounts of claims data. Some of this data is found in an Excel spreadsheet, some is located in a Microsoft Access table, and the rest is located in a SAS data set. For the purposes of this workshop, all three of these data sources are located in a folder called 'Workshop_2'.



Figure 1. The workshop data

The first step is to access these three data sources and combine them into a single SAS data set. First, lets go after the **Southeast** region claims found in a Microsoft Access table in the DEMO.MDB database.

- 1. From Enterprise Guide, select: File → Open → Data. (see figure 2.). This action opens the **Open Data** window.
- 2. From the **Open Data** window, browse to find the **Workshop_2** folder. Select **demo.mdb**, then click on the **Open** button (see figure 3.).

🞯 ws	2 - SAS Er	nterprise	e Guide							
File	Edit	View	Tasks	Fa	vorit	es	Progr	am	Тоо	ls
1	New			×					×	P
	Open			×	Ø	Pro	ject	C	trl+0)
	Close Pr	oject				Dat	ta			
	Save ws2	2	Ctrl+S			Dat	ta Expl	oratio	n	

Figure 2. Accessing the data

🧀 Open Data	
Browse Search SAS Fo	olders
Look in: <u> </u> Works	hop_2 👻 🔶 👻
🚱 Desktop	Name
My Documents	claims_2019 demo.mdb
😼 My Computer	Dhio_Valley_Claims
ត My Network	

Figure 3. The data location

This initiates the **Import Data** task. This task has a series of four windows to navigate. On the 1st window, just accept the default values and if you don't want the default name of the SAS data set (DEMO_0002), then you can specify one. See Figure 4. Then select **Browse**

🛃 Import Data	from demo.mdb	×
1 of 4 S	Specify Data	S .sas
The Import Data data analysis ar	a wizard is used to convert non-SAS data into a SAS data file which is requ id reporting.	uired by other tasks for
Source data file		
Location:	Local File System	
File path:	C:\Ben\access\Workshop_2\demo.mdb	
Data type:	Access Database	•
Output SAS data	set	
SAS server:	SASApp	Browse
Library:	WORK	
Data set:	DEMO_0002	
	<back next="" v=""> Finish C</back>	Cancel Help

Figure 4. Screen 1 - Import Data task

Selecting **Browse** opens the **Save File** window. In the **Save File** window, select the **WORK** library, then specify the name of the SAS data set (**SE_CLAIMS**), then select **Save**. This takes you back to Screen 1. Selecting **Next** takes you to **screen 2**. See Figure 5.

Import Data from demo.mdb	×
2 of 4 Select Data Source	S.Sas.
Select the Microsoft Access table or view: Customers Employees Invoice New_Results NewClaims NuClaims Orders OV_Claims results SE_Claims test	 Examine all text field values in order to determine length. Rename columns to comply with SAS naming conventions.
Back V	ext> Finish Cancel Help

Figure 5. Screen 2 - Import Data task

Select **Next>** to go to Screen **3** of **4**. On the **third** screen (see figure 6.), select the columns to include in the SAS data set. In this case, we want all of then, so just select **Next >**.

🛃 In	🛃 Import Data from demo.mdb 🧾 💌							
3	3 of 4 Define Field Attributes							
Selec	Select columns and define attributes:							
Inc	Source Name	Name	Label	Туре	Source Informat	Len.	Output Format	Output Informat
	renw_sfx	renw_sfx	renw_sfx_nbr	String	\$CHAR3.		\$CHAR3.	\$CHAR3.
V	clm_nbr	clm_nbr	clm_nbr	String	\$CHAR10.	10	\$CHAR10.	\$CHAR10.
V	clm_rpt_dt	clm_rpt_dt	clm_rpt_dt	Date/	DATE9.	8	DATE9.	DATE9.
V	claim_paid	claim_paid	claim_paid	Number	BEST12.	8	BEST12.	BEST12.
V	adj_expns	adj_expns	adj_expns	Number	BEST12.	8	BEST12.	BEST12.
V	state	state	state	Number	BEST12.	8	BEST12.	BEST12.
V	statecode	statecode	statecode	String	\$CHAR22.	22	\$CHAR22.	\$CHAR22.
V	region	region	region	String	\$CHAR17.	17	\$CHAR17.	\$CHAR17.
V	total_paid	total_paid	total_paid	Currency	DOLLAR15.	8	DOLLAR1	DOLLAR15.
V	paid_yr	paid_yr	paid_yr	Number	BEST12.	8	BEST12.	BEST12.
S	elect All	Clear All						Modify
			<back td="" v<=""><td>ext></td><td>Finish</td><td></td><td>Cancel</td><td>Help</td></back>	ext>	Finish		Cancel	Help

Figure 6. Screen 3 - Import Data task

On the fourth screen, just accept the default values, then select **Finish.** This completes and runs the Import Data task. The process flow window now looks like this (see Figure 7.)

Process Flow -							
🕨 Run 👻 🔲 Sta	🕨 Run 👻 🔳 Stop Export 👻 Schedule 👻 📸 Project Log 🧱 Properties 👻						
2							
demo.mdb	Import Data (demo.mdb [SE_Claims])	Data Imported from demo.mdb					

Figure 7. Process Flow window

What looks like the name of the data set in the process flow window is actually the data set label. If you right click on the data set and select Properties, you will see that the actual name of the data set is WORK.SE_CLAIMS.

The next task is to get the data for the 'Ohio Valley' region which is in an Excel spreadsheet. We will do this after we complete the first exercise.

EXERCISE 1.

If you have not already done so, import the SE_CLAIMS access table and create a data set named WORK.SE_CLAIMS. When finished, the process flow window should look like Figure 7.

2. ACCESSING MICROSOFT EXCEL DATA:

You will notice a lot of similarity between section 1. and section 2. To access the data for the Ohio Claims, which is in an Excel spreadsheet, select **File --> Open --> Data** from the Process Flow window. See Figures 2 and 3. This time, from the **Open Data** window, select the Excel spreadsheet (Ohio_Valley Claims), then select **Open.** Again, this spawns the **Import Data** task which takes you through a series of four screens.



Figure 8a. Top portion of Screen 1

Source data file				
Location:	Local File System]		
File path:	C:\Ben\access\Workshop_2\Ohio_Valley_Claims.xlsx			
Data type:	Excel Workbook 🔹			
Output SAS data set				
SAS server:	SASApp	Browse		
Library:	WORK]		
Data set:	Ohio_Valley_Claims			
	<back vext=""> Finish Cancel</back>	Help		

Figure 8b. Lower portion of Screen 1

There are at least two things to notice: the default name of the SAS data set to be created (work. Ohio_Valley_Claims), and the Finish button is un-grayed. If you want all the default behavior of the Import Data task, (and we do), then select **Finish**. This completes and runs the **Import Data** task and creates the **WORK.Ohio_Valley_Claims** data set. Now the **process flow** window looks like this... see Figure 9.

Process Flow +			
🕨 Run 👻 🔳 St	op Export + Sch	nedule 👻 📴 Project Lo	g 🛛 🔚 Properties 🕞
2			
demo.mdb	Import Data (demo.mdb [SE_Claims])	Data Imported from demo.mdb	
<u>×</u>			
Ohio_Valley_Cl aims.xlsx	Import Data (Ohio_Valley_Cl aims.xlsx[Shee	Data Imported from Ohio_Valley_C	

Figure 9. Process Flow window

Again, what looks like the name of the SAS data set is a data set label. If you right click on the new data set and select Properties, you will see the actual name of the data set is WORK.Ohio_Valley_Claims.

EXERCISE 2.

If you have not already done so, import the Ohio_Valley_Claims spreadsheet and create a data set named WORK.Ohio_Valley_CLAIMS. When finished, the process flow window should look like Figure 9.

3. ACCESSING THE CLAIMS_2019 SAS DATA SET:

If we know the library where **CLAIMS_2019** is located, we can expand the library list until we find the data set, then just drag and drop it into the Process Flow window. If there is NOT a SAS library established that contains the **CLAIMS_2019** data set, then select **File --> Open --> Data** and browse to the location (folder) where this data set is located. Since this data is already in the form of a SAS data set, the data set icon appears in the Process Flow window. See Figure 10.

Process Flow -		
🕨 Run 👻 🔳 St	op Export + Sch	nedule 👻 🔭 Project Log 📰 Properties
2		
demo.mdb	Import Data (demo.mdb [SE_Claims])	Data Imported from demo.mdb
×		
Ohio_Valley_Cl aims.xlsx	Import Data (Ohio_Valley_Cl aims.xlsx[Shee	Data Imported from Ohio_Valley_C
CLAIMS_2019		

Figure 10. The Process Flow window

Note: The default behavior for Enterprise Guide is to Open the SAS data set when you bring it into the Process Flow window. If the data set does not open right away, then double click on its icon in the process flow window and the data set will open.

4. MANIPULATING THE DATA

Now that all the data appears in the Process Flow window, the next task is to combine all the data. To do this, we are going to start by opening the Claims_2019 data set by double clicking on it. Then select **Data** --> Append Table...

CLA	IMS_2019 -								
	Filter and Sort 🛱	Query Builder	Where	Data	▼ Describe ▼ Graph ▼ Analyze ▼	Ехро	rt 👻 Send To	-	
	🔌 pol_id_nbr	💩 renw_sfx_nbr	🔌 pol_exp		Append Table	paid	🔞 adj_expns	🔞 state	💩 statecode
1	089Y22189	4	19		Sort Data	4.34	116	42	PA
2	916Q54846	0	16	SW,	Create Format	75.9	93	25	MA
3	631R31755	9	19	w.d Sw.		6.22	149	6	CA
4	193S25625	0	17	#. d	Create Format from Data Set	6.22	160	20	KS
5	816R17804	7	19		Transpose	0000	194	2	AK

Figure 11. Initiating the Append Table task

Tables Results	Tables	Tables					
	Tables to append						
	Table Name	File Location					
	CLAIMS_2019	WS_2.CLAIMS_2019					
	Add Table	Delete Table	catenated. Use this area				

When the Append Data task opens, it looks like Figure 12.

Figure 12. The Append Table task

The CLAIMS_2019 is considered the BASE table, and we want to ADD to the base. The first thing to do is to click on the **Add Table** button. We want to ADD the SE_CLAIMS and OHIO_VALLEY_CLAIMS table to the base. The Add Table button opens the **Open Data** window.

Look in: 🔯 Projec	t 🗸] + • € ;	× 🖆 🛙	- \$5		
🚱 Desktop	Name /	Label	Туре	Source	Location	Last Modified
 My Documents My Computer My Network Places Servers SAS Folders 	MM_ACC.NewClaims (SASApp) WORK.DEMO_0000 (SASApp) WORK.DEMO_0001 (SASApp) WORK.NEW_CLAIMS (SAS WORK.OHIO_VALLEY_CLAI WORK.OV_CLAIMS (SASApp) WORK.SE_CLAIMS (SASApp) WS_2.CLAIMS_2019 (SASA	NewClaims Data Imported f Data Imported f NEW_CLAIMS Data Imported f OV_CLAIMS Data Imported f CLAIMS_2019	Input Data Input Data Input Data Input Data Input Data Input Data Input Data Input Data	Import D Import D Import D Program Import D	Process Process Process Process Process Process Process	3/26/2019 9:1 3/25/2019 10: 3/25/2019 10: 3/26/2019 9:1 3/26/2019 1:0 3/26/2019 1:0 3/26/2019 12: 3/26/2019 1:5
	File name: Data Imported fr Files of type: All File Types	om Ohio_Valley_Cla	ims.xlsx			

Figure 13. The Open Data window

When the Open Data window opens, the current project is the initial view. From it, select the WORK.OHIO_VALLEY_CLAIMS data set, then select **Open.** The **Append Table** window now looks like this (see Figure 14).

Tables Results	Tables		
	Tables to append		2
	Table Name	File Location	
	CLAIMS_2019 OHIO_VALLEY_CLAIMS	WS_2.CLAIMS_2019 WORK.OHIO_VALLEY_CLAIMS	
	Add Table	Delete Table	

Figure 14. The Append Table task

Select the **Add Table** button again to repeat the process to add the SE_CLAIMS data set. When all three data sets appear in the **Append Table** window, select **Run.** The Process Flow window now looks like this...

2			
demo.mdb	Import Data (demo.mdb [SE_Claims])	Data Imported from demo:mdb	Append_Table
<u>×</u>		-	
Ohio_Valley_Cl aims.xlsx	Import Data (Ohio_Valley_Cl aims.xlsx[Shee	Data Imported from Ohio_Valley_C	
CLAIMS_2019			

Figure 15. The Process Flow window

The default name of the appended data set is WORK.APPEND_TABLE. The presenter will show you how to modify the Append Table task render a 'customized' name for the data set.

EXERCISE 3.

If you have not already done so, append the three tables so that the **Process Flow** looks like Figure 15. Let the name of the resulting data set default to WORK.APPEND_TABLE. Or give it another name of your choice.

Now that we have the three tables appended, we can proceed to analyze the data and generate some reports.

4. REPORT GENERATION

The first report we want to generate will have more to do with analyzing the data to detect possible 'dirty' data. Let's first look at the Regions. There should be only six regions represented in the data. Are there six regions? How can we find out quickly? Open the Append_Table data set and select **Describe --> One-Way Frequencies** as shown in Figure 16.

Ар	Append Table -									
	📰 Input Data (3) 🗒 Code 📋 Log 🖾 Output Data									
9	Ϛ 🔍 Modify Task 🐺 Filter and Sort 🏨 Query Builder 🍸 Where Data 🕶 Describe 🕶 Graph 🕶 Analyze 📼 Export 👻 Send To 👻 🔢									
	💩 pol_id_nbr	💩 renw_sfx_nbr	🔌 pol_expr_yr	🔌 clm_nbr			List Data	ons	🔞 state	💧 s
1	089Y22189	4	19	3800F14A		~	Summany Statistics Wizard	116	42	PA
2	916Q54846	0	16	2637D19B		2	Summary Statistics Wizard	93	25	MA
3	631R31755	9	19	5466A57D		Σ	Summary Statistics	149	6	CA
4	193S25625	0	17	5995B76F			Summary Tables Wizard	160	20	KS
5	816R17804	7	19	4710A78D			Summer Tables	194	2	AK
6	429L68921	9	19	0090B48B			Summary Tables	42	30	MT
7	915O64546	3	19	8409A10B			List Report Wizard	208	49	UT
8	508X62711	6	15	5389C44A		673	Characterize Data	208	49	UT
9	430J52746	1	17	4957B33B			Distribution Application	199	2	AK
10	392S41269	3	18	4850B48E		HID	Distribution Analysis	137	25	MA
11	292K87387	0	19	1623A68F			One-Way Frequencies	72	53	WA
12	771K04832	0	19	9062E11E			Table Analysis	281	44	RI
13	440S79859	1	17	5104F89C		17JU	JL2015 80000	202	6	CA

Figure 16. Initiating the One-Way Frequencies task

When the One-Way Frequencies Task opens, drag **Region** and drop it on the Analysis variables slot. The **Run** button becomes 'un-grayed', so select it.

Data Statistics	Data				
Plots Results Titles Properties	Data source: SASApp:WORK.APPEND_TABLE Task filter: None				
	Variables to assign: Name pol_id_nbr pol_id_nbr pol_expr_yr clm_nbr clm_npt_dt adj_expns state statecode region Task roles: Analysis variables variable required> Frequency count (Limit Group analysis by Frequency count (Limit Region Frequency count (Limit Frequency count (Limit Frequency count (Limit Frequency count (Limit Region Frequency count (Limit Frequency count (Limit Fre				
Preview code Run Save Cancel Help The "Analysis variables" role must have at least 1 variable assigned to it.					

Figure 17. The One Way Frequencies task

When this task runs, it generates the output seen in Figure 18.

One-Way Frequencies Results							
The FREQ Procedure							
region Frequency Percent Frequency Percent							
North Central	114	20.25	114	20.25			
Northeast	94	16.70	208	36.94			
Ohio Valley	73	12.97	281	49.91			
Southeast	107	19.01	388	68.92			
Southwest	75	13.32	463	82.24			
West Coast	100	17.76	563	100.00			

Figure 18. One-Way Frequencies task output

There are six regions identified in this report and that is how many regions are in the data. Next open the **Append_Table** data set again. From the open data set, select **Describe --> Summary Statistics...**

Арр	Append_Table +									
9	🐺 Filter and Sort 🏥 Query Builder 🍸 Where Data 🔹 Describe 🔹 Graph 👻 Analyze 👻 Export 👻 Send To 👻 📳									
	💩 pol_id_nbr	🔌 renw_sfx_nbr	\land pol_expr_yr		List Data	aim_paid	🔞 adj_expns	12		
1	089Y22189	4	19	Σ Summary Statistics Wizard		624.34	116			
2	916Q54846	0	16			1875.9	93			
3	631R31755	9	19	Σ	Summary Statistics	1306.22	149			
4	193S25625	0	17	Summary Tables Wizard		1166.22	160			
5	816R17804	7	19				194			
6	429L68921	9	19	Summary Tables		2371.48	42			

Figures 19. Initiating the Summary Statistics task

From the Data panel of the Summary Statistics task, make the screen look like Figure 20.



Figure 20. The Summary Statistics task

Make **Total_Paid** the Analysis variable, and **Region** the Classification variable, then select the **Basic Statistics** panel. Deselect all the default statistics and select the **Sum** statistic. Next, select **Run**. Look at Figure 21 to see the output.

🕄 Modify Task Export 🝷 Send To 👻 Create 👻 Publish 🔡 Properties						
Summary Statistics						
Results						
	The MEANS	Procedure				
	Analysis Variab	le : total_paid				
	region	Sum				
	North Central	2506774.90				
	Northeast	2211807.47				
	Ohio Valley 1051987.78					
	Southeast	1868571.65				
	Southwest	1323929.22				
	West Coast	1554472.53				

Figure 21. Summary Statistics task output screen

The only way to 'format' this output is to control the number of decimal places. Since Total_Paid is a monetary value, we want to add commas and a dollar sign to the output. Select **Modify Task** at the top of the output screen. Select 'Results' from the list on the left side of the window (see red arrow below). From the Results panel, (1.) select **Save statistics to a data set** and (2.) uncheck Show statistics. Select **Run**.

Data Statistics Basic	Results	
Percentiles	Save statistics to data set 1.	
Plots	SASApp:WORK.MEANSummaryStats	Browse
Results		
Properties	Show statistics 2.	
	Value to copy for Copy Variables role:	Maximum
	Combinations of classification variables:	N-way only
	Specify ways:	0, 1
		Examples: 1, 3, 14, 0 to 4 by 2

Figure 22. The modified Summary Statistics task.

The resulting data set is shown in Figure 23.

Summary Statistics 👻								
📰 Input Data 🗒 Code 📋 Log 📰 Output Data								
😏 🕟 Modify Task 🐺 Filter and Sort 🏨 Query Builder 🍸 Where Data 🔹 Describe 👻 Graph								
🔌 region	10 _WAY_	▲ _TYPE_	FREQ_	🛗 total_paid_Sum				
North Central	1	1	114	\$2,506,774.90				
Northeast	1	1	94	\$2,211,807.47				
Ohio Valley	1	1	73	\$1,051,987.78				
Southeast	1	1	107	\$1,868,571.65				
Southwest	1	1	75	\$1,323,929.22				
West Coast	1	1	100	\$1,554,472.53				
	Input Data Modify Task region North Central Northeast Ohio Valley Southeast Southeast Southwest West Coast	Input Data Code Log Modify Task Teller and So North Central 1 Northeast 1 Ohio Valley 1 Southeast 1 Southwest 1 West Coast 1	Input Data Code Log Output Data Modify Task Filter and Sort Query Built region Image: Code Image: Code Image: Code North Central 1 1 North Central 1 1 Ohio Valley 1 1 Southeast 1 1 Southwest 1 1 West Coast 1 1	Input Data Code Log Output Data Modify Task Imput Data Imput Data Imput Data Modify Task Imput Data Imput Data Imput Data Imput Data Imput Data Imput	Input DataCodeLogOutput DataModify TaskImage: Filter and SortQuery BuilderImage: WhereData • Describe •regionImage: WAY_Image: TYPE_Image: FREQ_Image: total_paid_SumNorth Central11114\$2,506,774.90Northeast1194\$2,211,807.47Ohio Valley1173\$1,051,987.78Southeast11107\$1,868,571.65Southwest1175\$1,323,929.22West Coast11100\$1,554,472.53			

Figure 23. The data set created from the Summary Statistics task

Notice that when you look at the data set this way, you see the effects of the dollar format on the variable **total_paid_sum.** We can turn this view of the data set into a report by selecting **Describe --> List Data**. This opens the List Data task. From the Data panel, drag the variables region **and total_paid_sum** and drop them as the two List variables.

🛅 List Data for SA	SApp:WORK.MEANSUMMARYSTATS
Data Options	Data
Intles Properties	Data source: SASApp:WORK.MEANSUMMARYSTATS Task filter: None Edit
	Variables to assign: Task roles: Name Ist variables Image: WAY_ Image: WAY_ Image: WAY_
Preview code	Run 💌 Save Cancel Help

Figure 24. The List Data task

Next, select Run. This generates a PROC PRINT report which reflects the effects of the formats.

Row numbe	r region	total_paid_Sum
	North Central	\$2,506,774.90
	2 Northeast	\$2,211,807.47
	3 Ohio Valley	\$1,051,987.78
	4 Southeast	\$1,868,571.65
-	5 Southwest	\$1,323,929.22
	6 West Coast	\$1,554,472.53

Figure 25. The List Data task output

Next, lets generate a vertical bar chart showing the amount of claims paid by region and year. From the open data set, select Graphs --> Bar Chart... as shown in Figure 26.

Арр	Append_Table +							
🐺 F	ilter and Sort 🖷	Query Builder 🏾 🍸	Describe +	Graph - Analyze - Export - Send To				
	🔌 pol_id_nbr	💩 renw_sfx_nbr	💩 pol_expr_yr	💧 clm_n	📶 🛛 Bar Chart Wizard			
1	089Y22189	4	19	3800F14A	Bar Chart			
2	916Q54846	0	16	2637D19B				
3	631R31755	9	19	5466A57D	Pie Chart Wizard			
4	193S25625	0	17	5995B76F	Pie Chart			
5	816R17804	7	19	4710A78D				
6	429L68921	9	19	0090B48B	Line Plot Wizard			
7	915O64546	3	19	8409A10B	Line Plot			

Figure 26. Initiating the Bar Chart task





Figure 27. The Bar Chart task gallery

Next, select the **Data** panel. When the Data panel opens, drag paid_year and drop it on 'Columns to Chart'. Drag region and drop it on 'Group bars by'. Finally, drag total_paid and drop it on 'Sum of'. See the next page.

III Bar Chart for SASApp	WORK.APPEND_TABLE
Bar Chart Data Appearance	Data
Bars Layout Axes General	Data source: SASApp:WORK.APPEND_TABLE Task filter: None Edit
Horizontal Axis Axis Vertical Axis Axis Major Ticks Minor Ticks Depth Axis Axis Chart Area Advanced Titles Properties	Columns to assign: Task roles: Name Image: Column to chart (Limit: Image: Columnto to chart (Limit: Image: Columnto to cha

Figure 28. Bar Chart data panel

Next, select the **Bars** panel. On this screen, select 'One bar for each unique data value'. This is necessary because the charting variable **paid_yr** is numeric. Undesired results would be generated if this is not done.



Figure 29. The Bars panel of the Bar Chart task.

Next, select the Advanced panel.

Bar Chart Data	Appearance > Advanced	
Appearance Bars Layout	Statistic used to calculate bar:	Accept missing values
Axes	Average 🗸 🗸	Skip zero values
Horizontal Axis Axis Vertical Axis Axis Major Ticks	Error bars are valid only when the chart statistic is Average or Percentage and the chart type is not a 3D grouped bar chart or a type that has a stacked column assigned.	Calculate percentages and cumulative percentages for each group (G100)
Minor Ticks Depth Axis Axis Chart Area Advanced Titles Properties	Display error bars Type: CLM: Both 95 Error bar color:	 Specify one statistical value to show for bars Percentage Display statistical value inside the bar
,		

From the Advanced panel, make sure the statistic is Average, then select Run.

Figure 30. The Advanced Panel of the Bar Chart task

The report looks like this...



Figure 31. The Results of the Bar Chart task

Many more reports will be generated in the Hands-on-Workshop as exercises.

CONCLUSION

The results of almost every SAS program can be accomplished by dragging and dropping and pointing and clicking your way through Enterprise Guide.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

The author can be reached at:

Ben Cochran The Bedford Group (919) 741-0370 bentcochran@gmail.com

