Homework Data Step and Sql Interface

The data for these exercises are found in the mac1 subdirectory.

Level 1

1. **Creating Macro Variables with the SYMPUTX Routine**
	1. Copy the program shown below into the Editor window. Submit the program and examine the output that it creates.

**%macro emporders(idnum=121044);**

 **proc print data=orion.orders noobs;**

 **var Order\_ID Order\_Type Order\_Date Delivery\_Date;**

 **where Employee\_ID=&idnum;**

 **title "Orders Received by Employee &idnum";**

 **run;**

**%mend emporders;**

**%emporders()**

* 1. Modify the macro to include a DATA step that creates a macro variable named NAME based on the variable **Employee\_Name** found in the **orion.employee\_addresses** data set.
	2. Modify the TITLE statement to display the name of the employee instead of the employee’s ID number. Call the macro using the default parameter value.
	3. Call the macro again, but with a parameter value of 121066.



Level 2

1. **Creating Macro Variables with the SYMPUTX Routine**
	1. Copy the program shown below into the Editor window. This program creates a summary data set named **customer\_sum** that summarizes **Total\_Retail\_Price** by **Customer\_ID** and sorts the data set by descending **CustTotalPurchase**. Submit the program and examine the output that it creates.

**proc means data=orion.order\_fact nway noprint;**

 **var Total\_Retail\_Price;**

 **class Customer\_ID;**

 **output out=customer\_sum sum=CustTotalPurchase;**

**run;**

**proc sort data=customer\_sum;**

 **by descending CustTotalPurchase;**

**run;**

**proc print data=customer\_sum(drop=\_type\_);**

**run;**

* 1. Create a macro variable named TOP that contains the ID number for the top customer. Then modify the program to print only the orders for Orion’s top customer.



* 1. Modify the program to print the customer’s name instead of the customer’s ID in the TITLE statement. Customer names are found in the **orion.customer\_dim** data set.



Level 3

1. **Creating Macro Variables with the SYMPUTX Routine**
	1. Copy the program shown below into the Editor window. Submit the program and examine the output that it creates.

**proc means data=orion.order\_fact nway noprint;**

 **var Total\_Retail\_Price;**

 **class Customer\_ID;**

 **output out=customer\_sum sum=CustTotalPurchase;**

**run;**

**proc sort data=customer\_sum ;**

 **by descending CustTotalPurchase;**

**run;**

**proc print data=customer\_sum(drop=\_type\_);**

**run;**

* 1. Using the **customer\_sum** data set, create a single macro variable, TOP3, that contains the customer IDs of the top three customers by revenue.

Note that **Customer\_ID** is a numeric variable.

* 1. Using the **orion.customer\_dim** data set, print a listing of the top three customers.



Level 1

1. **Creating Multiple Macro Variables with the SYMPUTX Routine**
	1. Copy the program shown below into the Editor window.

**%macro memberlist(id=1020);**

 **%put \_user\_;**

 **title "A List of &id";**

 **proc print data=orion.customer;**

 **var Customer\_Name Customer\_ID Gender;**

 **where Customer\_Type\_ID=&id;**

 **run;**

**%mend memberlist;**

**%memberlist()**

* 1. The **orion.customer\_type** data set contains the variable **Customer\_Type\_ID**, which uniquely identifies the customer membership level and activity level. Modify the macro to include a DATA step to create a series of macro variables named TYPE*xxxx*, where *xxxx* is the value of **Customer\_Type\_ID**. The value of each TYPE macro variable should be the value of **Customer\_Type**.
	2. Modify the TITLE statement so that it displays the appropriate customer type. Use an indirect macro variable reference to one of the TYPE variables based on the current value of ID. Submit the modified program.
	3. Call the macro again, but with a parameter value of 2030.



Level 2

1. **Using Indirect References in a Macro Call**
	1. Copy the program shown below into the Editor window. Submit the program and examine the results.

**data \_null\_;**

 **set orion.customer\_type;**

 **call symputx('type'||left(Customer\_Type\_ID), Customer\_Type);**

**run;**

**%put \_user\_;**

**%macro memberlist(custtype);**

 **proc print data=orion.customer\_dim;**

 **var Customer\_Name Customer\_ID Customer\_Age\_Group;**

 **where Customer\_Type="&custtype";**

 **title "A List of &custtype";**

 **run;**

**%mend memberlist;**

* 1. Create a macro variable named NUM with the value of 2010. Execute the macro so that the value of CUSTTYPE resolves to Orion Club members low activity in the macro call.



Level 3

1. **Using a Table Lookup Application**
	1. Using **orion.country**, create a series of macro variables in which the name of the macro variable is the country abbreviation (**Country**) and the value of the macro variable is the country name (**Country\_Name**). Use a %put statement to display the macro variables.
	2. Copy the program shown below into the Editor window.

**%let code=AU;**

**proc print data=Orion.Employee\_Addresses;**

 **var Employee\_Name City;**

 **where Country="&code";**

 **title "A List of *xxxxx* Employees";**

**run;**

* 1. Use indirect macro variable referencing to replace the ***xxxxx*** with the appropriate country name.

(Partial output):



Level 1

1. **Resolving Macro Variables with the SYMGET Function**
	1. Copy the program shown below into the Editor window to create a series of macro variables containing the customer type. Submit the first DATA step and the %PUT statement
	and examine the results.

**data \_null\_;**

 **set orion.customer\_type;**

 **call symputx('type'||left(Customer\_Type\_ID), Customer\_Type);**

**run;**

**%put \_user\_;**

**data us;**

 **set orion.customer;**

 **where Country="US";**

 **keep Customer\_ID Customer\_Name Customer\_Type\_ID;**

**run;**

**proc print data=us noobs;**

 **title "US Customers";**

**run;**

* 1. Modify the second DATA step to create a new variable named **CustType** that contains the value of the macro variable TYPE*xxxx* created in part **a**. Add the new variable to the KEEP statement.

Level 2

1. **Investigating Macro Variable Storage and Resolution**

Determine the type, length, and value of the DATA step variables in the program below.

**%let var1=cat;**

**%let var2=3;**

**data test;**

 **length s1 s4 s5 $ 3;**

 **call symputx('var3','dog');**

 **r1="&var1";**

 **r2=&var2;**

 **r3="&var3";**

 **s1=symget('var1');**

 **s2=symget('var2');**

 **s3=input(symget('var2'),2.);**

 **s4=symget('var3');**

 **s5=symget('var'||left(r2));**

**run;**

|  |  |  |  |
| --- | --- | --- | --- |
| Name  | Type | Length | Value |
| R1 |  |  |  |
| R2 |  |  |  |
| R3 |  |  |  |
| S1 |  |  |  |
| S2 |  |  |  |
| S3 |  |  |  |
| S4 |  |  |  |
| S5 |  |  |  |

Hint: Mimic SAS processing by making *two* passes through the program: a word scanning/compilation pass and an execution pass.

Hint: Draw a symbol table, updating it as each macro variable is created and populated.

Level 1

1. **Creating Macro Variables Using SQL**
	1. Copy the program shown below into the Editor window. Submit the program and review the results.

**%let start=01Jan2007;**

**%let stop=31Dec2007;**

**proc means data=orion.order\_fact noprint;**

 **var Total\_Retail\_Price;**

 **output out=stats n=count mean=avg;**

 **run;**

**data \_null\_;**

 **set stats;**

 **call symputx('orders',count);**

 **call symputx('average',avg);**

**run;**

**proc gchart data=orion.order\_fact;**

 **vbar3d Order\_Type**

 **/ patternid=midpoint cframe=w shape=c discrete**

 **sumvar=Total\_Retail\_Price type=mean ref=&average;**

 **format Total\_Retail\_Price dollar4.;**

 **label Total\_Retail\_Price='Average Order';**

 **title1 h=1 "Report from &start to &stop";**

 **title2 h=1 f=swiss "Orders this period: " c=b "&orders";**

 **footnote1 h=1 f=swiss "Overall Average: " c=b**

 **"%sysfunc(putn(&average,dollar4.))";**

**run;**

**quit;**

* 1. Replace the PROC MEANS step and the DATA step with a PROC SQL step. Create the macro variables ORDERS and AVERAGE based on the variable **Total\_Retail\_Price** in the **orion.order\_fact** data set.
	2. Remove any extraneous blanks from the values. Use the %PUT statement to display the values.
	3. Submit the PROC GHART step and verify that the output is the same.
	4. Modify the PROC SQL step to create a third macro variable named FMTAVG that contains the formatted version of AVERAGE. Remove any extraneous blanks from the value.
	5. In the FOOTNOTE statement, replace **%sysfunc(putn(&average, dollar4.))** with
	the macro variable FMTAVG.

Level 2

1. **Creating a List of Values in a Macro Variable Using SQL**
	1. Copy the program below into the Editor window and modify the SQL procedure to create a macro variable named TOP3 that contains the customer ID numbers of the top three customers by **Total\_Retail\_Price** in the **orion.order\_fact** data set. Separate each of the values with a comma and a blank. Use the OUTOBS= option to limit the number of output rows.

**proc sql;**

 **select customer\_id**

 **from orion.order\_fact**

 **group by Customer\_ID**

 **order by total descending;**

**quit;**

**Remember:** A GROUP BY clause is used to summarize the data by customer ID number, and an ORDER BY clause is used to sort the data.

* 1. Submit the program and review the results.

Level 3

1. **Creating Multiple Macro Variables Using SQL**
	1. The **orion.customer\_type** data set contains the variable **Customer\_Type\_ID**, which holds the unique customer type codes. Use the SQL procedure to create a series of macro variables named CTYPE1 through CTYPE*xx*, where *xx* resolves to the number of rows that the query will return.

You will need two queries, one to return the number of rows that the query will return and the other to create CTYPE1 through CTYPE*xx.*

* 1. Open the program shown below to display only the macro variables that begin with CTYPE.

**proc sql;**

 **select name, value**

 **from dictionary.macros**

 **where name like "CTYPE%";**

**quit;**