

Homework #6-Controlling Input and Output; Summarizing Data

Directions: Please submit one program file, one output file, and one log file for the entire assignment. Use comment statements to separate your answers. For questions that do not require a SAS program use comment statements. For example:

```
/*
```

```
Question #1d: my answer
```

```
Question #2a: my answer
```

```
*/
```

```
/*Question #4b: */
```

```
--SAS program--
```

```
/*Question #5*/
```

Please make sure the log and output file contain only one run. For example, clear the screen for the log and output file and submit your program one last time before you upload your solutions to **Blackboard**. See lab 1 for the instructions on how to clear your output and log files.

Part I: Controlling Input and Output

1. Writing Observations Explicitly

The data set **orion.discount** contains information about various discounts that Orion Star runs on its products.

Partial **orion.discount**

Product_ID	Start_ Date	End_Date	Unit_Sales_ Price	Discount
210100100027	01MAY2011	31MAY2011	\$17.99	70%
210100100030	01AUG2011	31AUG2011	\$32.99	70%
210100100033	01AUG2011	31AUG2011	\$161.99	70%
210100100034	01AUG2011	31AUG2011	\$187.99	70%
210100100035	01MAY2011	31MAY2011	\$172.99	70%

- a. Due to excellent sales, all discounts from December 2011 are repeated in July 2012. Both the December 2011 and the July 2012 discounts are called the Happy Holidays promotion.
- Create a new data set named **work.extended** that contains all discounts for the Happy Holidays promotion.
 - Use a WHERE statement to read-only observations with a start date of 01Dec2011.
 - Create a new variable, **Promotion**, which has the value *Happy Holidays* for each observation.
 - Create another new variable, **Season**, that has a value of *Winter* for the December observations
 - Use an explicit OUTPUT statement to write the December observations.
 - Specify a new start date of 01Jul2012 and an end date of 31Jul2012 for the July 2012 discounts.
 - For the July observations, overwrite value for **Season** with *Summer*.
 - Use an explicit OUTPUT statement to write the July observations.
 - Drop the **Unit_Sales_Price** variable.



Use the date constant to subset

- b. Print the new data set.
- Add an appropriate title
 - Verify the results.

Partial PROC PRINT (332 Total Observations)

Obs	Product_ID	Start_ Date	End_Date	Discount	Promotion	Season
1	210200100007	01DEC2011	31DEC2011	50%	Happy Holidays	Winter
2	210200100007	01JUL2012	31JUL2012	50%	Happy Holidays	Summer
3	210200300013	01DEC2011	31DEC2011	50%	Happy Holidays	Winter
4	210200300013	01JUL2012	31JUL2012	50%	Happy Holidays	Summer
5	210200300025	01DEC2011	31DEC2011	50%	Happy Holidays	Winter

2. Creating Multiple SAS Data Sets with Derived Values

The data set **orion.orders** contains information about in-store, catalog, and Internet orders as well as delivery dates.

Partial **orion.orders** (490 Total Observations)

Order_ID	Order_ Type	Employee_ID	Customer_ID	Order_ Date	Delivery_ Date
1230058123	1	121039	63	11JAN2007	11JAN2007
1230080101	2	99999999	5	15JAN2007	19JAN2007
1230106883	2	99999999	45	20JAN2007	22JAN2007
1230147441	1	120174	41	28JAN2007	28JAN2007
1230315085	1	120134	183	27FEB2007	27FEB2007

- a. Orion Star wants to study catalog and Internet orders that were delivered quickly, as well as those that went slowly.
- Create three data sets named **work.fast**, **work.slow**, and **work.veryslow**.
 - Write a WHERE statement to read only the observations with **Order_Type** equal to 2 (catalog) or 3 (Internet).
 - Create a variable named **ShipDays** that is the number of days between when the order is placed and when the order is delivered.
 - Handle the output as follows:
 - Output to **work.fast** when the value of **ShipDays** is less than 3.
 - Output to **work.slow** when the value of **ShipDays** is 5 to 7.
 - Output to **work.veryslow** when the value of **ShipDays** is greater than 7.
 - Do not output an observation when the value of **ShipDays** is 3 or 4.
 - Drop the variable **Employee_ID**.
 - There should be 80 observations in **work.fast**, 69 observations in **work.slow**, and 5 observations in **work.veryslow**.



Of the 490 observations in **orion.orders**, only 230 are read due to the WHERE statement.

- b. Print your results from **work.veryslow** with an appropriate title.

work.veryslow

Orders taking more than 7 days to deliver						
Obs	Order_ID	Order_ Type	Customer_ID	Order_ Date	Delivery_ Date	Ship Days
1	1231305521	2	16	27AUG2007	04SEP2007	8
2	1236483576	2	70108	22JUL2009	02AUG2009	11
3	1236965430	3	70165	08SEP2009	18SEP2009	10
4	1237165927	3	79	27SEP2009	08OCT2009	11

3. Specifying Variables and Observations

The data set **orion.orders** contains information about in-store, catalog, and Internet orders as well as delivery dates.

Partial **orion.orders** (490 Total Observations)

Order_ID	Type	Order_ Employee_ID	Customer_ID	Date	Order_ Date	Delivery_
1230058123	1	121039	63	11JAN2007	11JAN2007	
1230080101	2	99999999	5	15JAN2007	19JAN2007	
1230106883	2	99999999	45	20JAN2007	22JAN2007	
1230147441	1	120174	41	28JAN2007	28JAN2007	
1230315085	1	120134	183	27FEB2007	27FEB2007	
1230333319	2	99999999	79	02MAR2007	03MAR2007	

- Create two data sets, **work.instore** and **work.delivery**, to analyze in-store sales.
 - Use a WHERE statement to read-only observations with **Order_Type** equal to 1.
 - Create a variable **ShipDays** that is the number of days between when the order was placed and when the order was delivered.
 - Output to **work.instore** when **ShipDays** is equal to 0.
 - Output to **work.delivery** when **ShipDays** is greater than 0.
 - The **work.instore** data set should contain three variables (**Order_ID**, **Customer_ID**, and **Order_Date**). The **work.delivery** data set should contain four variables (**Order_ID**, **Customer_ID**, **Order_Date**, and **ShipDays**).
 - Test this program by reading the first 30 observations that satisfy the WHERE statement. Check the SAS log to verify that no warnings or errors were reported.
- Modify the program to read the full **orion.orders** data set. Of the 490 observations in **orion.orders**, only 260 are read due to the WHERE statement.
- Print your results from **work.delivery** with an appropriate title.

Partial **work.delivery** (10 Total Observations)

Deliveries from In-store Purchases					
Obs	Order_ID	Customer_ID	Order_ Date	Ship Days	
1	1231468750	52	25SEP2007	5	
2	1231657078	63	29OCT2007	4	
3	1232648239	49	07APR2008	8	
4	1241063739	89	03JAN2011	1	
5	1241235281	171	23JAN2011	7	

- Use PROC FREQ to display the number of orders per year in **work.instore**. Add an appropriate title.
Hint: Format the variable **Order_Date** with a YEAR. format. Restrict the analysis to the variable **Order_Date** with a TABLES statement.

PROC FREQ Output

In-stock Store Purchases, By Year			
The FREQ Procedure			
Date Order was placed by Customer			
		Cumulative	Cumulative

Order_Date	Frequency	Percent	Frequency	Percent
2007	43	17.20	43	17.20
2008	50	20.00	93	37.20
2009	27	10.80	120	48.00
2010	67	26.80	187	74.80
2011	63	25.20	250	100.00



Part II-Summarizing Data

1. Creating Accumulating Totals with Conditional Logic

The data set **orion.order_fact** contains a group of orders across several years, sorted by **Order_Date**.

Partial Listing of **orion.order_fact** (617 Total Observations, 12 Total Variables)

Order_ID	Type	Order_Date	Order_Quantity
1230058123	1	11JAN2007	1
1230080101	2	15JAN2007	1
1230106883	2	20JAN2007	1
1230147441	1	28JAN2007	2
1230315085	1	27FEB2007	3

- Orion Star would like to analyze 2009 data by creating accumulating totals for the number of items sold from retail, catalog, and Internet channels.
 - The value of **Order_Type** indicates whether the sale was retail (=1), catalog (=2), or Internet (=3).
 - Create a data set named **work.typedtotals** with accumulating totals for **TotalRetail**, **TotalCatalog**, and **TotalInternet**, as described above.
 -  The variable **Quantity** contains the number of items sold for each order.
 - For testing your program in this step, read only the first 10 observations that satisfy the WHERE statement.
 -  Remember to process only those rows where **Order_Date** occurs in 2009.
- Continue testing your program by printing the results from part **a**. Print all the variables and verify that the program is correctly calculating values for the accumulating totals.

PROC PRINT Output

Obs	Customer_ID	Employee_ID	Street_ID	Date	Date	Order_Delivery_ Order_ID Type	Product_ID	Order_
1	195	120150	1600101663	02JAN2009	02JAN2009	1234437760 1	230100600028	
2	36	99999999	9260128237	11JAN2009	14JAN2009	1234534069 3	240800100026	
3	183	120121	1600100760	12JAN2009	12JAN2009	1234537441 1	240100200001	
4	16	99999999	3940105865	12JAN2009	14JAN2009	1234538390 2	220200300015	
...								
10	183	120179	1600100760	31JAN2009	31JAN2009	1234727966 1	240700400004	
Obs	Quantity	Total_Retail_ Price	CostPrice_ Per_Unit	Discount	Total Retail	Total Catalog	Total Internet	
1	2	\$193.40	\$48.45	.	2	0	0	
2	4	\$525.20	\$58.55	.	2	0	4	
3	1	\$16.00	\$6.35	.	3	0	4	
4	1	\$115.00	\$52.40	.	3	1	4	
...								
10	1	\$13.20	\$5.95	.	4	6	6	

- When the results from parts **a** and **b** are correct, do the following:
 - Modify the program to read all observations satisfying the WHERE statement. The resulting report contains 90 observations.

- Keep only the variables **Order_Date**, **Order_ID**, **TotalRetail**, **TotalCatalog**, and **TotalInternet**.
- Print your results with an appropriate title.


2. Summarizing and Grouping Data Using the DATA Step

The data set **orion.order_qtrsum** contains information about sales in a particular year for each customer, separated by month.

- For a given customer, there might be some months (and quarters) that the customer did not place an order.
- The variable **Order_Qtr** contains the appropriate quarter.

Partial **orion.order_qtrsum** (101 Total Observations)

Customer_ID	Qtr	Order_ Month	Order_ Sale_Amt
69	4	10	3.2
70187	4	11	8.2
10	2	6	12.2
70079	4	10	14.6
70165	3	7	16.6

 The data set is not sorted by **Customer_ID** and **Order_Qtr**.

- Create a data set named **work.qtrcustomers** that summarizes sales based on customer and quarter.
 - The variable **Total_Sales** should contain the total sales for each quarter within each **Customer_ID** value.
 - Create a variable named **Num_Months** that counts the total months within each quarter that the customer had an order.
- Print your results.
 - Display **Total_Sales** with a DOLLAR11.2 format.
 - Add an appropriate title.

Partial PROC PRINT Output (74 Total Observations)

Total Sales to each Customer for each Quarter					
Obs	Customer_ID	Order_ Qtr	Total_Sales	Num_ Months	
1	5	2	\$604.80	2	
2	5	3	\$52.50	1	
3	5	4	\$33.80	1	
4	10	1	\$32.60	1	
5	10	2	\$342.80	3	

Part III- Supplemental exercises for STAT 625 and Honors credit

1. Using Conditional Logic to Output Multiple Observations

The data set **orion.country** contains information about country names as well as various lookup codes.

orion.country

Country	Country_Name	Population	Country_ ID	Continent_ ID	Country_Former Name
AU	Australia	20,000,000	160	96	
CA	Canada	.	260	91	
DE	Germany	80,000,000	394	93	East/West Germany

IL	Israel	5,000,000	475	95
TR	Turkey	70,000,000	905	95
US	United States	280,000,000	926	91
ZA	South Africa	43,000,000	801	94

- Create a new data set that contains one observation for each current country name as well as one observation for each former country name.
 - Use conditional logic and explicit OUTPUT statements to create a data set named **work.lookup**.
 - If a country has a former country name, write two observations: one with the current name in the **Country_Name** variable and another with the former country name in the **Country_Name** variable.
 - Drop the variables **Country_FormerName** and **Population**.
 - Create a new variable named **Outdated** with values of either *Y* or *N* to indicate whether the observation represents the current country name.
- Print the new data set with an appropriate title.

PROC PRINT Output

Current and Outdated Country Name Data					
Obs	Country	Country_Name	Country_ID	Continent_ID	Outdated
1	AU	Australia	160	96	N
2	CA	Canada	260	91	N
3	DE	Germany	394	93	N
4	DE	East/West Germany	394	93	Y
5	IL	Israel	475	95	N
6	TR	Turkey	905	95	N
7	US	United States	926	91	N
8	ZA	South Africa	801	94	N

2. Identifying Extreme Values in Each Group of Data

The data set **orion.customer_dim** contains information about Orion Star customers.

Partial **orion.customer_dim** (77 Total Observations, 11 Total Variables)

Customer_ID	Customer_Name	Customer_Type	BirthDate	Customer_
4	James Kvarniq	Orion Club members low activity	27JUN1978	
5	Sandrina Stephano	Orion Club Gold members medium activity	09JUL1983	
9	Cornelia Krah	Orion Club Gold members medium activity	27FEB1978	
10	Karen Ballinger	Orion Club members high activity	18OCT1988	
11	Elke Wallstab	Orion Club members high activity	16AUG1978	

Use First./Last. processing to create the report below. Show data on the oldest and youngest customers for each **Customer_Type**.

- The variable **o_ID** is the **Customer_ID** value of the oldest customer and **y_ID** is the **Customer_ID** value of the youngest customer for each group.
- Create a variable named **agerange** to indicate the spread between these oldest and youngest customers.
- Use **Customer_BirthDate**, rather than **Customer_Age**, for all age determinations because this is more accurate.

Partial PROC PRINT Output (7 Total Observations)

Oldest and Youngest Customers of each Customer Type					
Customer_Type	oldest	youngest	o_ID	y_ID	agerange
Internet/Catalog Customers	08JUL1938	18AUG1973	29	54655	35.1
Orion Club members high activity	28SEP1938	24OCT1990	89	46966	52.1
Orion Club members medium activity	20JAN1938	16SEP1992	70059	2806	54.7
Orion Club Gold members high activity	16JAN1938	25JUL1988	50	39	50.5
Orion Club Gold members low activity	19DEC1973	21JUL1992	70201	13	18.6



There are several ways to obtain the number of years between two dates. Two possible techniques are dividing days by 365.25 or using the YRDIF function.

