Jason Michaels
Math510
Homework 7 (SAS HW 1)

3.2 - (Levels 1, 2, Challenge)

LEVEL 1:
4. Accessing a Permanent Data Set

a. Use an interactive facility to explore Orion and answer the following questions:

How many observations are in the **orion.country** data set? **7**
How many variables are in the **orion.country** data set? **6**
What is the name of the last country in the data set? **South Africa**

b. Submit a PROC CONTENTS step to generate a list of all members in the orion library. What is the name of the last member listed?

```
proc contents data=orion._all_ nods;
run;
```

The last member listed is US_SUPPLIERS

LEVEL II
5. Viewing General Data Set Properties

a. Examine the general data set properties of orion.staff.

```
proc contents data=orion.staff;
run;
```

b. What sort of information is stored in this data set?

The variables are Employee ID, Start Date, End Date, Job Title, Salary, Gender, Birth Data, Hire Date, Termination Data, Manager ID. It is not sorted.

CHALLENGE
6. SAS Autoexec File

Use the Help facility or product documentation to investigate the SAS autoexec file and answer the following questions. (used documentation at https://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#a000104286.htm#autoexec)
What is the name of the file? autoexec.sas
What is its purpose? The autoexec file contains commands that are executed automatically when you start up SAS.

How is it created? You create it using a text editor (preferably an SAS text editor) and then use the save as dialogue box.
How could this be useful in a SAS session? One of the uses for it could be to set up a proper path and to automatically submit a LIBNAME statement. In our case, it could be used in lieu of running the libname.sas program.

Chapter 4.1 - Levels 1, 2

LEVEL I

1. **Displaying orion.order_Fact with the PRINT Procedure**

a. Retrieve the starter program p104e01. Run the program and view the output. Observe that there are 617 observations. Observations might be displayed over two lines, depending on output settings.

b. Add a SUM statement to display the sum of Total_Retail_Price.

```
sum Total_Retail_Price;
```

c. Add a WHERE statement to select only the observations with Total_Retail_Price more than 500. Submit the program. Verify that 35 observations were displayed.

```
where Total_Retail_Price > 500;
```

What do you notice about the Obs column? The observation column still contains the original values (i.e. it doesn't start at 1 and move upwards in increments of 1).

Did the sum of Total_Retail_Price change to reflect only the subset? Yes

d. Add an option to suppress the Obs column. Verify that there are 35 observations in the results.

```
proc print data=orion.order_fact noobs;
```

How can you verify the number of observations in the results? You can check the log tab to verify that there are 35 observations.

e. Add an ID statement to use Customer_ID as the identifying variable. Submit the program. The results contain 35 observations.

```
id Customer_ID;
```

How did the output change? Customer_ID becomes the leftmost column.

f. Add a VAR statement to display Customer_ID, Order_ID, Order_Type, Quantity, and Total_Retail_Price.

var Customer_ID Order_ID Order_Type Quantity Total_Retail_Price;

What do you notice about Customer_ID?

There is a duplicate Customer_ID column.

g. Modify the VAR statement to address the issue with Customer_ID

var Order_ID Order_Type Quantity Total_Retail_Price;

Our program ends up looking like this:

```
proc print data=orion.order_fact noobs;
        where Total_Retail_Price > 500;
        id Customer_ID;
        var Order_ID Order_Type Quantity Total_Retail_Price;
        sum Total_Retail_Price;
run;
```

LEVEL II

2. Displaying orion.customer_dim with the PRINT Procedure.

a. Write a PRINT step to display orion.customer_dim

```
proc print data=orion.customer_dim;
run;
```

b. Modify the program to display a subset of orion.customer_dim by selecting only the observations for customers between the ages of 30 and 40. Also, suppress the Obs column. The resulting report should contain 17 observations.

```
proc print data=orion.customer_dim noobs;
        where Customer_Age between 30 and 40;
run;
```

c. Add a statement to use Customer_ID instead of Obs as the identifying column. Submit the program and verify the results.

id Customer_ID;

d. Add a statement to limit the variables to those shown in the report below.

var Customer_Name Customer_Age Customer_Type;

Our final code looks like:

```
proc print data=orion.customer_dim noobs;
        where Customer_Age between 30 and 40;
        id Customer_ID
        var Customer_Name Customer_Age Customer_Type;
run;
```

Chapter 4.2 - Levels 1, 2

LEVEL 1

5. Sorting orion.employee_payroll and Displaying the New Data Set

a. open p104e05. Add a PROC SORT step before the PROC PRINT step to sort orion.employee_payroll by Salary, placing the sorted observations into a temporary data set named sort_salary.

```
proc sort data=orion.employee_payroll
        out=work.sort_salary;
        by Salary;
run;
```

b. Modify the PROC PRINT step to display the new data set. Verify that your output matches the report below.

```
proc print data=work.sort_salary;
run;
```

Our code looks like:

```
proc sort data=orion.employee_payroll
        out=work.sort_salary;
        by Salary;
run;

proc print data=work.sort_salary;
run;
```

6. Sorting orion.employee_payroll and Displaying Grouped Observations

a. Open p104e06. Add a PROC SORT step before the PROC PRINT step to sort orion.employee_payroll by Employee_Gender, and within gender by Salary in descending order. Place the sorted observations into a temporary data set named sort_salary2.

```
proc sort data=orion.employee_payroll
        out=work.sort_salary2;
        by Employee_Gender descending Salary;
run;
```

b. Modify the PROC PRINT step to display the new data set with the observations grouped by Employee_Gender.

```
proc print data=work.sort_salary2;
        by Employee_Gender;
run;
```

LEVEL II

7. Sorting orion.employee_payroll and Displaying a Subset of the New Data Set

a. Sort orion.employee_payroll by Employee_Gender, and by descending Salary within gender. Place the sorted observations into a temporary data set named sort_sal.

```
proc sort data=orion.employee payroll
        out=work.sort_sal;
        by Employee_Gender descending Salary;
run;
```

b. Print a subset of the sort_sal data set. Select only the observations for active employees (those without a value for Employee_Term_Data) who earn more than $65,000. Group the report by Employee_Gender, and include a total and subtotals for Salary. Suppress the Obs column. Display only Employee_ID, Salary, and Marital_Status. The results contain the 18 observations.

```
proc print data=work.sort_sal noobs;
        by Employee_Gender;
        sum Salary;
        where Employee_Term_Date is missing and Salary > 65000;
        var Employee_ID Salary Marital_Status;
run;
```

Chapter 4.3 - Levels 1, 2

LEVEL 1

9. Displaying Titles and Footnotes in a Detail Report

a. **Open and submit p104e09 to display all observations for Australian Sales Rep IVs.**

b. **Add a VAR statement to display only the variables shown in the report below.**

var Employee_ID First_Name Last_Name Gender Salary;

c. **Add TITLE and FOOTNOTE statements to include the titles and footnotes shown in the report below.**

title1 ‘Australian Sales Employees’;
title2 ‘Senior Sales Representatives’;
footnote1 ‘Job_Title: Sales Rep. IV’;

d. **Submit the program and verify the output. The results contain five observations as shown below.**

e. **Submit a null TITLE and null FOOTNOTE statement to clear all titles and footnotes.**

title;
footnote;

Final code looks like this:

```
title1 'Australian Sales Employees';
title2 'Senior Sales Representatives';
footnote1 'Job_Title: Sales Rep. IV';

proc print data=orion.sales noobs;
  where Country='AU' and Job_Title contains 'Rep. IV';
  var Employee_ID First_Name Last_Name Gender Salary;
run;
title;
footnote;
```

10. Displaying Column Headings in a Detail Report

a. Open and submit p104e10. Modify the program to define and use the following labels:

Employee_ID	**Employee ID**
First_Name	**First Name**
Last_Name	**Last Name**
Salary	**Annual Salary**

```
label Employee_ID = "Employee ID"
      First_Name = "First Name"
      Last_Name = "Last Name"
      Salary = "Annual Salary";
```

Final code looks like:

```
title 'Entry-level Sales Representatives';
footnote 'Job_Title: Sales Rep.I';

proc print data=orion.sales noobs label;
      where Country = 'US' and Job_Title='Sales Rep. I';
      var Employee_ID First_Name Last_Name Gender Salary;

      label Employee_ID = "Employee ID"
      First_Name = "First Name"
      Last_Name = "Last Name"
      Salary = "Annual Salary";
run;

title;
footnote;
```

Level 2

11. Writing an Enhanced Detail Report

a. Write a program to display a subset of orion.employee_addresses as shown below. The program should sort the observations by State, City, and Employee_Name and then display the sorted observations grouped by State. The resulting report should contain 311 observations.

```
proc sort data=orion.employee_addresses
      out=work.address;
      where Country='US';
      by State City Employee_Name;
run;

title "US Employees by State";
```

```
proc print data = work.address noobs split = ' ';
        var Employee_ID Employee_Name City Postal_Code;
        label Employee_ID = 'Employee ID'
                Employee_Name = 'Name'
                Postal_Code = 'Zip Code';
        by State;
run;
```

Chapter 5.1-Levels 1,2:

LEVEL I

1. Displaying Formatted Values in a Detail Report

a. **Open p105e01 and submit. Review the output.**

b. **Modify the PROC PRINT step to display only Employee_ID, Salary, Birth_Date, and Employee_Hire_Date.**

```
var Employee_ID Salary Birth_Date Employee_Hire_Date
```

c. **Add a FORMAT statement to display Salary in a dollar format, Birth_Date in 01/31/2012 date style, and Employee_Hire_Date in the 01JAN2012 date style, as shown in the report below.**

```
format Salary dollar11.2 Birth_Date mmddyy10. Employee_Hire_Date date_9.;
```

Our final code looks like:

```
proc print data=orion.employee_payroll;

  var Employee_ID Salary Birth_Date Employee_Hire_Date

format Salary dollar11.2 Birth_Date mmddyy10. Employee_Hire_Date

date_9.;
```

```
run;
```

LEVEL II

2. Displaying Formatted Values in a Detail Report

a. Write a PROC PRINT step to display the report below using orion.sales as input. Subset the observations and variables to produce the report shown below. Include titles, labels, and formats. The results contain 13 observations.

```
title1 'US Sales Employees';
title2 'Earning Under $26,000';
proc print data=orion.sales label noobs;
        where Country='US' and Salary<26000
        label First_Name='First Name'
                Last_Name='Last Name'
                Job_Title='Title'
                Hire_Date='Date Hired';
        var Employee_ID First_Name Last_Name Job_Title Salary Hire_Date;
        format Salary dollar10. Hire_Date monyy7.;
run;
title;
```

Chapter 5.2 - Levels 1, 2

LEVEL I

Level 1

4. Creating User-Defined Formats

a. Retrieve the starter program p105e04.

b. Create a character format named $GENDER that displays gender codes as follows:

```
proc format;
   value $gender
         'M' = 'Male'
         'F' = 'Female'
run;
```

c. Create a numeric format named MNAME that displays month numbers as follows:

```
value mname
   1 = 'January'
   2 = 'February'
   3 = 'March';
```

d. Add a PROC PRINT step to display the data set, applying these two user-defined formats to the Employee_Gender and BirthMonth variables, respectively.

```
proc print data=Q1Birthdays
   var Employee_ID Employee_Gender BirthMonth;
   format Employee_Gender $gender.
         BirthMonth mname.;
```

```
run;
```

e. Submit the program to produce the following report. The results contain 113 observations.

```
data Q1Birthdays;
   set orion.employee_payroll;
   BirthMonth=month(Birth_Date);
   if BirthMonth le 3;
run;
proc format;
   value $gender
         'M' = 'Male'
         'F' = 'Female'
value mname
   1 = 'January'
   2 = 'February'
   3 = 'March';
run;
title 'Employees with Birthdays in Q1';
proc print data=Q1Birthdays
   var Employee_ID Employee_Gender BirthMonth;
```

```
   format Employee_Gender $gender.
   BirthMonth mname.;
run;
run;
title;
```

5. Defining Ranges in User-Defined Formats

a. Retrieve the starter program p105e05.

b. Create a character format named $GENDER that displays gender codes as follows:

```
proc format;
   value $gender
                'M' = 'Male'
                'F' = 'Female'
                other = 'Invalid code';
```

c. Create a numeric format named SALRANGE that displays salary ranges as follows:

```
value salrange .='Missing salary'
        20000-<100000='Below $100,000'
        100000-500000='$100,000 or more'
        other='Invalid salary';
```

```
run;
```

d. In the PROC PRINT step, apply these two user-defined formats to the Gender and Salary variables, respectively. Submit the program to produce the following report:

```
proc print data=orion.nonsales;
        var Employee_ID Job_Title Salary Gender;
        format Salary salrange. Gender $gender.;
run;
```

Our overall program looks like this:

```
proc format;
   value $gender
        'M' = 'Male'
        'F' = 'Female'
        other = 'Invalid code';
    value salrange .='Missing salary'
        20000-<100000='Below $100,000'
        100000-500000='$100,000 or more'
        other='Invalid salary';
run;

title1 'Salary and Gender Values';
title2 'for Non-Sales Employees';
```

```
proc print data=orion.nonsales;
        var Employee_ID Job_Title Salary Gender
        format Salary salrange. Gender $gender.;
run;
title;
```

Chapter 6.2 - Level 2

5. Subsetting Observations Based on Three Conditions

a. Write a DATA step to create work.delays using orion.orders as input.

b. Create a new variable, Order_Month, and set it to the month of Order_Date. Hint: Use the MONTH function.

```
Order_Month = month(Order_Date);
```

c. Use a WHERE statement and a subsetting IF statement to select only the observations that meet all of the following conditions:

Delivery_Date values that are more than four days beyond Order_Date

Employee_ID values that are equal to 999999999

Order_Month values occurring in August

```
where Order_Date=4 < Delivery_Date
        and Employee_ID=99999999;
```

d. The new data set should include only Employee_ID, Customer_ID, Order_Date, Delivery_Date, and Order_Month.

```
keep Employee_ID Customer_ID Order_Date Delivery_Date Order_Month;
```

e. Add parmanent labels for Order_Date, Delivery_Date, and Order_Month as shown below.

label Order_Date='Date Ordered';

Delivery_date = 'Date Delivered'

Order_Month='Month Ordered'

f. Add permanent formats to display Order_Date and Delivery_Date as MM/DD/YY

format Order_Date Delivery_Date mmddyy10.;

g. Add a PROC CONTENTS step to verify that the labels and formats were stored permanently.

proc contents data=work.delays;

run;

h. Write a PROC PRINT step to create the report below. Results should contain nine observations.

proc print data=work.delays;

run;

Final code looks like:

data work.delays;

set.orion.orders;

where Order_Date+4<Delivery_Date

and Employee_ID=99999999;

Order_Month = month(Order_Date);

if Order_Month=8;

label Order_Date='Date Ordered';

Delivery_Date='Date Delivered'

```
        Order_Month='Month Ordered'
        format Order_Date Delivery_Date mmddyy10.;
    keep Emplyee_ID Customer_ID Order_Date Delivery_Date
        Order_Month;
run;
proc contents data=work.delays;
run;

proc print data=work.delays;
run;
```

Chapter 9.1 - Level 2

2. Creating New Variables

a. Write a DATA step that reads orion.customer to create work.birthday.

```
data work.birthday
    set orion.customer
run;
```

b. In the DATA step, create three new variables: Bday2012 (combo of the month of birth_Date, the day of Birth_Date, and the constant of 2012 in the MDY function), BdayDOW2012 (day of the week of Bday2012), and Age2012 (age of the customer in 2012. Subtract Birth_Date from Bday2012, and divide the result by 365.25).

```
Bday2012 = mdy(month(Birth_Date), day(Birth_Date), 2012);
BdayDOW2012 = weekday(Bday2012);
Age2012 = (Bday2012-Birth_Date)/365.25
```

c. Include only the following variables in the new data set: Customer_Name, Birth_Date, Bday2012, BdayDOW2012, and Age2012.

keep Customer_Name Birth_Date Bday2012 BdayDOW2012 Age2012

d. Format Bday2012 to display in the form 01Jan2012. Age2012 should be formatted to display with no decimal places.

format Bday2012 date9. Age2012 3.;

e. Write a PROC PRINT step to create the report below. The result should contain 77 observations.

proc print data=work.birthday;

run;

Final code looks like:

```
data work.birthday
        set orion.customer
        Bday2012 = mdy(month(Birth_Date), day(Birth_Date), 2012);
        BdayDOW2012 = weekday(Bday2012);
        Age2012 = (Bday2012-Birth_Date)/365.25;
        keep Customer_Name Birth_Date Bday2012 BdayDOW2012 Age2012;
        format Bday2012 date9. Age2012 3.;
run;
proc print data=work.birthday;
run;
```

Chapter 9.2 - Level 2

6. Creating Multiple Variables in Conditional Processing

a. Write a DATA step that reads orion.customer_dim to create work.season

b. Create two new variables Promo and Promo2

The value of Promo is based on the quarter in which the customer was born. If the customer was born in the first quarter, than = Winter, if 2nd quarter, Spring, 3rd, Summer, 4th Fall. The value of Promo2 is based on the customer's age. 18-24 set = YA, 65+ = Senior, missing for all others.

c. The new data set should include only Customer_FirstName, Customer_LastName, Customer_BirthDate, Customer_Age, Promo, and Promo2.

d. Create the report below. The results should include 77 observations.

```
data work.season;
        set orion.customer_dim;
        length Promo2 $ 6;
        Quarter=qtr(Customer_BirthDate);
        if Quarter=1 then Promo='Winter';
        else if Quarter=2 then Promo='Spring';
        else if Quarter=3 then Promo='Summer';
        else if Quarter=4 then Promo='Fall';
        if Customer_Age>18 and Customer_Age<=25 then Promo2='YA'
        else if Customer_Age>=65 then Promo2='Senior';
        keep Customer_FirstName Customer_LastName Customer_BirthDate
                Customer_Age Promo Promo2;
run;

proc print data=work.season;
run;
```

7. Creating Variables Unconditionally and Conditionally

a. Write a DATA step that reads orion.orders to create work.ordertype.

b. Create a new variable, DayOfWeek, that is equal to the weekday of Order_Date.

c. Create the new variable Type, which is equal to Retail Sale if Order_Type = 1, Catalogue Sale if = 2, Internet Sale if 3.

d. Create the new variable SaleAds, which is equal to Mail if Order_Type=2, Email if = 3.

e. Do not include order_Type, Employee_ID, and Customer_ID in the new data set.

f. Create the report below. The results should contain 490 observations.

```
data work.ordertype;
        set orion.orders;
        DayOfWeek=weekday (Order_Date) ;
        if Order_Type = 1 then Type = 'Retail Sale';
        else if Order_Type = 2 then do;
                Type = 'Catalog Sale';
                SaleAds = 'Mail';
        end;
        else if Order_Type=3 then do;
                Type = 'Internet Sale';
                SaleAds='Email';
        drop Order_Type Employee_ID Customer_ID;
end;
```

Chapter 10.1, 10.3, 10.4 - Level 2

3. Concatenating Data Sets with Variables of Different Lengths and Types

a. Open p110e03. Sumbit the PROC CONTENTS steps or explore the data sets interactively to complete the table below by filling in attribute information for each variable in each data set.

b. Write a DATA step to concatenate orion.charities and orion.us_suppliers, creating a temporary data set, contacts.

```
data work.contacts;

    set orion.charities orion.us_suppliers;

run;
```

c. Submit a PROC CONTENTS step to examine work.contacts. From which input data set were the variable attributes assigned?

```
proc contents data=work.contacts;

run;
```

The variable attributes were assigned from orion.charities

d. Write a DATA step to concatenate orion.us_suppliers and orion.charities, creating a temporary data set, contacts2. Note that these are the same data sets as the previous program, but they are in reverse order.

```
data work.contacts2;

    set orion.us_suppliers orion.charities;

run;
```

e. Submit a PROC CONTENTS step to examine work.contacts2. From which input data set were the variable attributes assigned?

```
proc contents data=work.contacts2

run;
```

The variable attributes were assigned from orion.us_suppliers

f. Write a DATA step to concatenate orion.us_suppliers and orion.consultants, creating a temporary data set, contacts3.

```
data work.contacts3;
    set orion.us_suppliers orion.consultants;
run;
```

Why did the DATA step fail? ContactType is defined as both character and numeric.

5. Merging a Sorted Data Set and an Unsorted Data Set in a One-to-Many Merge

a. Sort orion.product_list by Product_Level to create a new data set, work.product_list.

```
proc sort data=orion.product_list
    out=work.product_list
    by Product_Level;
run;
```

b. Merge orion.product_level with the sorted data set. Create a new data set, work.listlevel, which includes only Product_ID, Product_Name, Product_Level, and Product_Level_Name.

```
data work.listlevel;
    merge orion.product_level work.product_list;
    by Product_Level;
    keep Product_ID Product_Name Product_Level Product_Level_Name;
```

c. Create the report below, including only observations with Product Level = 3. The results should contain 13 observations.

```
proc print data=work.listlevel noobs;
        where Product_Level = 3;
run;
```

8. Merging Using the IN=and RENAME= Opptions

a. Write a PROC SORT step to sort orioncustomer by Country to create a new data set, work.customer.

```
proc sort data=orion.customer
    out=work.customer;
    by Country;
run;
```

b. Write a DATA step to merge the resulting data set with orion.lookup_country by Country to create a new data set, work.allcustomer. In the orion.lookup_country data set, rename Start to Country and rename Label to Country_Name. Include only four variables: Customer_ID, Country, Customer_Name, and Country_Name

```
data work.allcustomer;
        merge work.customer
                orion.lookup_country(rename=(Start=Country
                                             Label=Country_Name) ;
        by Country;
        keep Customer_ID, Country, Customer_Name, Country_Name
```

c. Create the report below. The results should contain 308 observations.

d. Modify the DATA step to store only the observations that contain both customer information and country information. A subsetting IF statement that references IN=variables in the MERGE statement must be added.

```
data work.allcustomer;
    merge work.customer (in=Cust)
        orion.lookup_country(rename=(Start=Country
                                     Label=Country_Name) in=Ctry);
    by Country;
    keep Customer_ID, Country, Customer_Name, Country_Name
```

e. Submit the program to create the report below. The results should contain 77 observations.

```
proc print data=work.allcustomer;
run;
```