STAT 506: Homework 5

For these problems you will need to access the data in the PG2/data folder. Use the `libname` statement we learned to load this each time you work on your assignments. You should call it '`pg2`' to be consistent with the SAS materials. Note that it's not PG1 anymore -- Make sure you've downloaded the new folder and run cre8data.sas. I tried to *italicize* the parts where I expect you to actually show me something in your homework solutions if it is not obvious.

1. **Debugging the PDV**

   Modify the code below to do the following:

   - Ensure that the values of **Size** won't get truncated.
   - Add PUTLOG statements to provide the following information in the log:
     - Immediately after the SET statement, write "START DATA STEP ITERATION" to the log as a color-coded note.
     - Before the Type= assignment statement, write the value of **Type** to the log.
     - After the Type= assignment statement, write the value of **Type** to the log.
     - At the end of the DATA step, write the contents of the entire PDV to the log.

```
data np_monuments;
  set pg2.np_final;
  keep Region ParkName AvgMonthlyVisitors Acres Size;
  where Type="MONUMENT";
  format AvgMonthlyVisitors Acres comma10.;
  Type=propcase(Type);
  AvgMonthlyVisitors=sum(DayVisits,Campers,OtherLodging)/12;
  if Acres<1000 then Size="Small";
  else if Acres<100000 then Size="Medium";
  else Size="Large";
run;
```

   Show a screenshot of the log showing the last four iterations of the DATA step.

2. **Directing Output to Multiple Tables**

   Based on the data in **pg2.np_yearlytraffic**, create three new tables named **monument**, **park**, and **other**.

   - Use SELECT and WHEN statements (see https://v8doc.sas.com/sashtml/lgref/z0201966.htm or elsewhere) to direct the output:
     - If **ParkType** is "National Monument", then the row should be output to the **monument** table
     - If **ParkType** is "National Park", then the row should be output to the **park** table
     - If **ParkType** is anything else, the row should be output to the **other** table
   - Only read the columns **ParkName**, **ParkType**, and **Location** into the PDV
   - Drop **ParkType** from the **monument** and **park** tables

   Show your code and a screenshot of the corresponding log notes.

3. **Creating Summed Columns**

Open the **pg2.np_yearlytraffic** table. Notice the **Count** column records the number of cars that have passed through a particular **Location** at each **Park**.

Write a PROC SORT step to create a new table named **traffic_sort** based on the data in **pg2.np_yearlytraffic**:

- The table should be sorted by **ParkType**, and then by **ParkName** within **ParkType** (both in ascending order).
- The table should only include rows where **ParkType** is "National Preserve", "National River", or "National Seashore".

Write a DATA step to create a new table named **total_traffic** based on **traffic_sort**:

- Group the data by **ParkType**, and then by **ParkName** within **ParkType** (both in ascending order).
- Create a new column named **TypeCount** that is the running total of **Count** within each **ParkType**. Reset the count for each new value of **ParkType**.
- Create a new column named **NameCount** that is the running total of **Count** within each **ParkName**. Reset the count for each new value of **ParkName**.
- Format **TypeCount** and **NameCount** so values are displayed with commas.
- Keep only the **ParkType**, **ParkName**, **TypeCount**, and **NameCount** columns.
- Write only the last row for each **ParkName** to the output table.

Write a PROC PRINT to display **total_traffic**.

Show all your code and a screenshot of the PROC PRINT output.

4. **Using Numeric Functions**

Modify the following code to do the following:

- Create a new column named **MonthlyRainTotalMM** which is **MonthlyRainTotal** multiplied by 25.4 and rounded to the nearest half millimeter.
- Create a new column named **Date** which is the date portion of the **DateTime** column.
- Create a new column named **MonthEnd** which is the date corresponding to the last day of that month.
- Format **Date** and **MonthEnd** as date values.
- Keep only the **StationName**, **MonthlyRainTotal**, **MonthlyRainTotalMM**, **Date**, and **MonthEnd** columns.
- Only output a row if it is the last row within its **Month**.
- Add a PROC PRINT to display **rainsummary**.

```
data rainsummary;
  set pg2.np_hourlyrain;
  by Month;
  if first.Month=1 then MonthlyRainTotal=0;
  MonthlyRainTotal+Rain;
run;
```

Show all your code and a screenshot of the PROC PRINT output.

5. Using Character Functions

Write a DATA step to create a new table named **clean_traffic** based on **pg2.np_monthlytraffic**:

- Add a LENGTH statement to create a new five-character-long column named **Type**. Add an assignment statement that uses the SCAN function to extract the last word from the **ParkName** column and assign the resulting value to **Type**.
- Use the SUBSTR function to create a new column named **Park** that reads each **ParkName** value and excludes the **Type** code at the end of the string. *Note*: Use the FIND function to identify the position number of the **Type** string (with any trailing blanks removed from **Type**). That value can be used in calculating the third argument of the SUBSTR function to specify how many characters to read.
- Add an assignment statement to use the UPCASE and COMPRESS functions to change the case of **Region** and remove any blanks.
- Add an assignment statement to change the case of **Location** to proper case. Use the COMPBL function to remove any extra blanks between words.
- Use the TRANWRD function to create a new column named **Gate** that reads **Location** (after performing the previous step) and converts the string "Traffic Count At" to a blank.
- Create a new column named **GateCode** that concatenates **ParkCode** and **Gate** together with a colon (:) between the strings.

Then write a PROC PRINT step to display the first 10 rows of **clean_traffic** where **Month** is equal to 3. Only display the columns **Type**, **Park**, **Region**, **Location**, **GateCode**, and **Count**, in that order. Include a descriptive title.

Show all your code and a screenshot of the PROC PRINT output.

[Question 6 on next page]

6. Using Functions to Optimize Column Length

- Run the program below. Notice that the **Column1** column contains raw data with values separated by various symbols. The SCAN function is being used to extract the **ParkCode** and **ParkName** values. Also examine the PROC CONTENTS report. Notice that **ParkCode** and **ParkType** have a length of 200, which is the same as **Column1**. *Note*: When the SCAN function creates a new column, the new column will have the same length as the column listed as the first argument.
- The **ParkCode** column should include only the first four characters in the string. Add a LENGTH statement in the proper place in the DATA step to define the length of **ParkCode** as 4.
- The length for the **ParkName** column could be optimized by determining the longest string and setting an appropriate length. Modify the DATA step to create a new column named **NameLength** that uses the LENGTH function to return the position of the last non-blank character for each value of **ParkName**.
- Use a RETAIN statement to create a new numeric column named **MaxLength** that has an initial value of 0. Use an assignment statement and the MAX function to set the value of **MaxLength** to either the current value of **NameLength** or **MaxLength**, whichever is larger.
- Use the END= option in the SET statement (see https://v8doc.sas.com/sashtml/lgref/z0173782.htm, especially Example 11) to create a temporary variable in the PDV named **LastRow**. **LastRow** will be 0 for all rows until the last row of the table, when it will be 1. Add an IF-THEN statement to write the value of **MaxLength** to the log if the value of **LastRow** is 1.

```
data parklookup;
  set pg2.np_unstructured_codes;
  ParkCode=scan(Column1, 2, '{}:,"()-');
  ParkName=scan(Column1, 4, '{}:,"()');
run;
proc print data=parklookup(obs=10); run;
proc contents data=parklookup; run;
```

*Show your final DATA step code and include a screenshot of the log that shows the resulting value of **MaxLength**.*

Note: Although it would be the next logical step, you don't actually need to end up changing the length of **ParkName** to **MaxLength**. I know you know how to do that.