STAT 506: Homework 7

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.

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. **Iterative and Conditional DO Loops**

   Using the data in **pg2.eu_sports**, write a DATA step to create a table named **MoreExports** by doing the following:

   - Subset the input data to only look at rows where **Year** is 2015, **Country** is "Malta", and **Sport_Product** is "FISHING" or "GOLF".
   - In a DO loop, increase **Year** from 2017 in 2-year jumps until either (1) **Year** is 2031 or (2) **Amt_Export** exceeds 1/4 of **Amt_Import**, whichever comes first.
     - In each iteration of the loop, increase **Amt_Export** by 31%.
     - Output during each iteration of the loop.

   Write a PROC PRINT step to display **MoreExports**.

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

2. **Using a Character Variable as an Index and Terminating a DO Loop Early**

   Using the data in **pg2.np_summary**, we wish to estimate each park's visitors over (up to) the next three years. Create a table named **VisitProj** in a DATA step by doing the following:

   - Create an index variable in a DO loop named **Projected**, which iterates through the *character* values "1st Year", "2nd Year", and "3rd Year". (see a helpful blog here, especially the first example in the section discussing "foreach loops": https://blogs.sas.com/content/iml/2011/09/07/loops-in-sas.html).
     - In each iteration of the DO loop, increase **DayVisits** by 4%.
     - Following that increase, use an IF condition to terminate the DO loop early if **DayVisits** exceeds 16000. You can exit the loop using the LEAVE statement (see a helpful blog here: https://blogs.sas.com/content/iml/2017/03/15/leave-continue-sas.html).
     - Do not include any explicit output statements.
   - Drop all columns except **ParkName**, **Projected**, and **DayVisits**.

   Write a PROC PRINT step to display the first 6 rows of **VisitProj**.

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

3. **Using a DATA step to Convert a Wide Table to a Narrow Table**

   The table **pg2. np_2017camping** is currently stored as a wide table. Write a DATA step to convert it to a narrow table named **CampNarrow**. **CampNarrow** should only contain three columns:

   - **ParkName** (which has the same values as the input table)
   - **CampType** (which has the values "Tent", "RV", or "Backcountry": ensure that the entire value is stored)
   - **CampCount** (which has the numerical counts from the input table)

   Write a PROC PRINT step to display the first 6 rows of **CampNarrow**.

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

4. **Using a DATA step to Convert a Narrow Table to a Wide Table**

   The table **pg2. np_2016camping** is currently stored as a narrow table. Note that the table is currently sorted by **ParkName**. Write a DATA step to convert it to a wide table named **CampWide**. **CampWide** should only contain four columns:

   - **ParkName** (which has the same values as the input table: there should be one row per park)
   - **Tent** (which has the numerical counts from the input table when **CampType** is "Tent")
   - **RV** (which has the numerical counts from the input table when **CampType** is "RV")
   - **Backcountry** (which has the numerical counts from the input table when **CampType** is "Backcountry")

   Write a PROC PRINT step to display the first 6 rows of **CampWide**.

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

5. **Using PROC TRANSPOSE**

   The table **pg2.weather_highlow** contains both the high and low monthly temperatures for 4 locations.

   - Sort the table by **Location**, and then by **Month** in descending alphabetical order, and then by **Temp** in ascending order.
     - Save the sorted table as **sort_lowhigh**.
     - Do not do any subsetting of the table – it should still include all its original rows and columns.
   - Use PROC TRANSPOSE to create a table named **HighTemps** from **sort_lowhigh**.
     - **HighTemps** should contain only four columns: **Location**, **Jun**, **Jul**, and **Aug**. Those month columns should contain the high temperatures for the respective months (*e.g.*, for "Black Canyon Of The Gunnison, CO", the highs for Jun, Jul, and Aug are 84, 92, and 87, respectively).
     - Note that by default, PROC TRANSPOSE will not appreciate there being multiple occurrences of the same ID value in the same BY group. To override this, see the SAS documentation for the LET option in the PROC TRANSPOSE statement.
   - Write a PROC PRINT step to display **HighTemps**.

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