STAT 506: Homework 6

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. Changing Variable Types

Run the code below (exactly as shown -- don't add any extra spaces or anything) to create a table named **bad:**

```
data bad;
input date $10. zip;
datalines;
2023/10/31 47907
2023/11/23 47906
;
```

Write a new DATA step (don't edit the one above) using **bad** as the input table to create a new table named **better** which has the same two columns (same names too) as **bad**, but with **date** saved as a numeric SAS date column and with **zip** saved as a character column.

Write steps to display both the descriptor portion and the data portion of **better**.

Show all your new code and screenshots of both (i) the last table from the descriptor portion and (ii) the data portion.

2. Creating Custom Permanent Formats

Write a PROC FORMAT step to create two new permanent formats in the **pg2.hwformats** catalog:

- The SOMEREG format should format character values as follows:
  - MW => "Midwest"
  - NE => "Northeast"
  - SE => "Southeast"
  - other codes => "All Other Regions"
- The PARKSIZE format should format numeric values as follows:
  - Less than 30,000 => "Small"
  - 30,000 through less than 600,000 => "Average"
  - 600,000 and more => "Large"

Write a PROC FREQ step to create a two-way table comparing the columns **Reg** and **Acres** from **pg2.np_summary**, applying the SOMEREG format to the **Reg** column and the PARKSIZE format to the **Acres** column.

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

3. Creating a Custom Date Format Using an Existing Format

Look up the SAS 9.4 documentation for the VALUE statement in PROC FORMAT. Look specifically at the syntax (and examples) that reference *using existing SAS formats as labels* (these are called “nested formats”).

Write a PROC FORMAT step to create a temporary format named DECADE that categorizes dates as identified below:

- January 1, 2000 - December 31, 2009 => the text string “2000-2009”
- January 1, 2010 - December 31, 2017 => the text string “2010-2017”
- January 1, 2018 - March 31, 2018 => the text string “1st Quarter 2018”
- April 1, 2018 and beyond => the actual date value displayed using the MMDDYY10. format (nested)

Write a PROC FREQ step to create a one-way table of the column **Date** from **pg2.np_weather**, including only rows where the column **Prcp** is above 0.35. Format **Date** with the DECADE format.

Show your code and a screenshot of the PROC FREQ output.

4. Creating a Custom Format from a Table

- Run the program below and review the results. Notice that some of the park types are still displayed as abbreviated codes because the custom format does not include a label for those values.

```
proc format cntlin=pg2.np_types_regions;
run;

proc freq data=pg2.np_summary;
    table Type / nocum;
    format Type $TypCode.;
run;
```

- Create an output table named **typfmtout** from the existing $TypCode format. To do this, you’ll need to use the CNTLOUT= option and a SELECT statement (See a great example of these here: https://blogs.sas.com/content/sgf/2017/12/04/controlling-your-formats/). The **typfmtout** table contains several extra columns, but the critical columns for this problem are **FmtName**, **Start**, and **Label**. Notice that the values for **FmtName** do not include the $ as a prefix.
- Open the **pg2.np_newcodes** table. Notice that it contains the format name, the Type values, and the labels in the **FmtName**, **Start**, and **Label** columns, respectively.
- Write a DATA step that creates a table named **typfmt_update** by concatenating the **typfmtout** table and the **pg2.np_newcodes** table. Update all the values of **FmtName** to be equal to “$TypCode”. Keep only the **FmtName**, **Start**, and **Label** columns.
- Write a PROC FORMAT step that re-creates the $TypCode format using the CNTLIN= option to read the new **typfmt_update** table that contains the updated format values.
- Run the original PROC FREQ step again and verify that all **Type** codes are displayed with labels.

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

5. **Merging Tables**

The table **pg2.np_2016traffic** contains the columns:

- **Code**
- **Year**
- **Month** [For this problem, we will only be interested in rows from this table where **Month** = 11]
- **MonthCount**

The table **pg2.np_acres** contains the columns:

- **Region**
- **ParkCode**
- **ParkName**
- **State**
- **GrossAcres**

The table **pg2.np_types_regions** contains the columns:

- **FmtName**
- **Start**
- **Label** [*Hint*: check out the rows of this column corresponding to a **FmtName** of "$RegCode"]

Examine these tables to see which columns they have in common (see the *Hint* above; it's not immediately obvious).
Use these common columns to create a new merged table named **nov_merged** (see the subsetting criterion above).
Do not use PROC SQL. *Hint*: this whole process might take a few PROC and DATA steps.
Only include rows which appeared in all three original tables.
Drop the columns **Year**, **Month**, and **FmtName**.
Then write a PROC PRINT to display the first 10 rows of **nov_merged**.
Show all your code and a screenshot of the PROC PRINT output.