Practice

If you restarted your SAS session, open and submit the **libname.sas** program in the course files.

Level 1

4. Generating an Accumulating Column within Groups

The **pg2.np_yearlyTraffic** table contains annual traffic counts at locations in national parks. Park names are grouped into park types.

a. Open the **p202p04.sas** program in the **practices** folder. Complete the PROC SORT step to sort the **pg2.np_yearlyTraffic** table by **ParkType** and **ParkName**.

b. Modify the DATA step as follows:

1) Read the sorted table created in PROC SORT.
2) Add a BY statement to group the data by **ParkType**.
3) Create a column, **TypeCount**, that is the running total of **Count** within each value of **ParkType**.
4) Format **TypeCount** so that values are displayed with commas.
5) Keep only the **ParkType** and **TypeCount** columns.

c. Run the program and confirm that **TypeCount** is reset at the beginning of each **ParkType** group.

d. Modify the program to write only the last row for each **ParkType** to the output table.

	ParkType	TypeCount
1	National Monument	7,042,169
2	National Park	46,643,794
3	National Preserve	1,067,315
4	National River	1,499,496
5	National Seashore	6,622,359

Level 2

5. Generating an Accumulating Column within Multiple Groups

The **sashelp.shoes** table contains sales information for various products in each region and subsidiary. Numbers for sales and returns are recorded for each row. Create a summary table that includes the sum of **Profit** for each region and product.

a. Create a sorted copy of **sashelp.shoes** that is ordered by **Region** and **Product**.

b. Use the DATA step to read the sorted table and create a new table named **profitsummary**. Create a column named **Profit** that is the difference between **Sales** and **Returns**.

c. Create an accumulating column named **TotalProfit** that is a running total of **Profit** within each value of **Region** and **Product**. Reset **TotalProfit** for each new combination of **Region** and **Product**. Run the program and verify that **TotalProfit** is accurate.

d. Modify the DATA step to include only the last row for each **Region** and **Product** combination. Keep **Region**, **Product**, and **TotalProfit**, and format **TotalProfit** as a currency value.

	Region	Product	TotalProfit
1	Africa	Boot	$115,222
2	Africa	Men's Casual	$546,686
3	Africa	Men's Dress	$308,405
4	Africa	Sandal	$181,887
5	Africa	Slipper	$325,667
6	Africa	Sport Shoe	$21,271
7	Africa	Women's Casu...	$405,668
8	Africa	Women's Dress	$363,695
9	Asia	Boot	$61,332
10	Asia	Men's Casual	$10,921

Challenge

6. Creating Multiple Output Tables Based on Group Values

The **pg2.np_acres** table contains acreage amounts for national parks. The park state is also provided. However, some parks span multiple states and therefore have multiple rows of data.

a. Create two tables from the **pg2.np_acres** table:

- **singlestate**, which contains the rows with unique park names
- **multistate**, which contains the rows with park names that appear in multiple states.

The parks should be grouped within their associated regions. When sorting the data, you need to keep only the **Region**, **ParkName**, **State**, and **GrossAcres** columns.

singlestate (5 of 367 rows)

	Region	ParkName	State	GrossAcres
1	Alaska	ALAGNAK WILD RVR	AK	30,665
2	Alaska	ANIAKCHAK N PRESERVE	AK	464,118
3	Alaska	ANIAKCHAK NM	AK	137,176
4	Alaska	BERING LAND BRIDGE N PRES	AK	2,697,391
5	Alaska	CAPE KRUSENSTERN NM	AK	649,096
6	Alaska	DENALI NP & PRES	AK	6,036,899

multistate (5 of 89 rows)

	Region	ParkName	State	GrossAcres
1	Alaska	KLONDIKE GOLD RUSH NHP	AK	12,996
2	Alaska	KLONDIKE GOLD RUSH NHP	WA	12,996
3	Intermountain	BIGHORN CANYON NRA	MT	120,296
4	Intermountain	BIGHORN CANYON NRA	WY	120,296
5	Intermountain	DINOSAUR NM	CO	210,282
6	Intermountain	DINOSAUR NM	UT	210,282

End of Practices