SP201-202 Doing More with SAS Programming

201 Controlling DATA Step Processing

202 Creating Accumulating Column and Processing Data in Groups

```
/*********************************************/
/* This code sets the PATH macro variable to   */
/* your EPG294/data folder and assigns the     */
/* PG2 library to that path. You must run      */
/* this code each time you start SAS OnDemand  */
/* for Academics to access your practice data  */
/*********************************************/

%let path=~/EPG2V2/data;
libname PG2 "&path";

*************************************************************;
*  p201a03.sas Activity 1.03                                *;
*  1) Run the program and examine the log, PROC CONTENTS  *;
*     report and output table.                          *;
*  2) Move the DROP statement to the end of the DATA step,*;
*     just before the RUN statement. Run the program and  *;
*     examine the log, PROC CONTENTS report, and output   *;
*     table. Did the results change?                    *;
*  3) Move the LENGTH statement between the DROP and RUN  *;
*     statements. Run the program and examine the log,    *;
*     PROC CONTENTS report, and output table. Did the     *;
*     results change?                                   *;
*************************************************************;
```

```
data storm_complete;
        set pg2.storm_summary_small;
        length Ocean $ 8;
        drop EndDate;
        where Name is not missing;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        if substr(Basin,2,1)="I" then Ocean="Indian";
        else if substr(Basin,2,1)="A" then Ocean="Atlantic";
        else Ocean="Pacific";
run;

proc contents data=storm_complete;
run;



data storm_complete;
        set pg2.storm_summary_small;
        length Ocean $ 8;
        where Name is not missing;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        if substr(Basin,2,1)="I" then Ocean="Indian";
        else if substr(Basin,2,1)="A" then Ocean="Atlantic";
        else Ocean="Pacific";
        drop EndDate;
run;
```

```
proc contents data=storm_complete;
run;



data storm_complete;
        set pg2.storm_summary_small;
        where Name is not missing;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        if substr(Basin,2,1)="I" then Ocean="Indian";
        else if substr(Basin,2,1)="A" then Ocean="Atlantic";
        else Ocean="Pacific";
        drop EndDate;
        length Ocean $ 8;
run;

proc contents data=storm_complete;
run;
```

The CONTENTS Procedure

Data Set Name	WORK.STORM_COMPLETE	Observations	3038
Member Type	DATA	Variables	6
Engine	V9	Indexes	0
Created	04/27/2021 00:24:18	Observation Length	56
Last Modified	04/27/2021 00:24:18	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

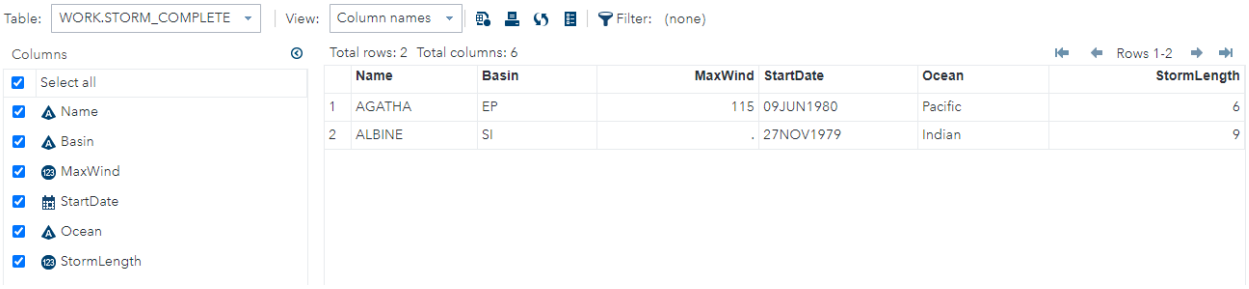
Engine/Host Dependent Information	
Data Set Page Size	131072
Number of Data Set Pages	2
First Data Page	1
Max Obs per Page	2334
Obs in First Data Page	2289
Number of Data Set Repairs	0
Filename	/saswork/SAS_workE72700018B2D_odaws04-usw2.oda.sas.com/SAS_workD01400018B2D_odaws04-usw2.oda.sas.com/storm_complete.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	1074795290
Access Permission	rw-r--r--
Owner Name	u58304328
File Size	384KB
File Size (bytes)	393216

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
2	Basin	Char	2	
3	MaxWind	Num	8	
1	Name	Char	15	
5	Ocean	Char	8	
4	StartDate	Num	8	DATE9.
6	StormLength	Num	8	

```
*************************************************************;
*  p201a04.sas Activity 1.04                                  *;
*  1) Examine the PUTLOG statements that are in the DATA  *;
*    step.                                  *;
*  2) Add two PUTLOG statements before the RUN statement  *;
*    to print "PDV before RUN statement" and write all   *;
*    columns in the PDV to the log. Run the program.     *;
*  3) View the log. What is the value of StormLength at   *;
*    the end of the second iteration of the DATA step?   *;
*  4) Type NOTE: (use uppercase and include the colon)    *;
*    inside the quotation marks of the following PUTLOG  *;
*    statement. Run the program. What changes in the     *;
*    log?                                   *;
*        putlog "NOTE: PDV before RUN statement";      *;
*************************************************************;
*  Syntax                                    *;
*    PUTLOG _ALL_;                              *;
*    PUTLOG column=;                             *;
*    PUTLOG "message";                           *;
*************************************************************;

data storm_complete;
          set pg2.storm_summary_small(obs=2);
   putlog "PDV after SET statement";
          putlog _all_;
          length Ocean $ 8;
          drop EndDate;
          where Name is not missing;
          Basin=upcase(Basin);
```

```
        StormLength=EndDate-StartDate;
        putlog StormLength=;
        if substr(Basin,2,1)="I" then Ocean="Indian";
        else if substr(Basin,2,1)="A" then Ocean="Atlantic";
        else Ocean="Pacific";
        *Add PUTLOG statements;
        putlog "NOTE: PDV before RUN statement";
run;


data storm_complete;
        set pg2.storm_summary_small(obs=2);
   putlog "NOTE: PDV after SET statement";
        putlog _all_;
        length Ocean $ 8;
        drop EndDate;
        where Name is not missing;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        putlog StormLength=;
        if substr(Basin,2,1)="I" then Ocean="Indian";
        else if substr(Basin,2,1)="A" then Ocean="Atlantic";
        else Ocean="Pacific";
        *Add PUTLOG statements;
        putlog "NOTE: PDV before RUN statement";
run;
```



	Name	Basin	MaxWind	StartDate	Ocean	StormLength
1	AGATHA	EP	115	09JUN1980	Pacific	6
2	ALBINE	SI	.	27NOV1979	Indian	9

```
************************************************************;
*  p201a06.sas Activity 1.06                                *;
*  1) Add an explicit OUTPUT statement after each           *;
*     ProjectedSales assignment statement. Run the program*;
*     How many rows are in the output table?                *;
*  2) Comment the final OUTPUT statement and run the        *;
*     program again. ARe rows where Year=3 written to the *;
*     new table?                                           *;
************************************************************;


data forecast;
        set sashelp.shoes;
        keep Region Product Subsidiary Year ProjectedSales;
        format ProjectedSales dollar10.;
   Year=1;
        ProjectedSales=Sales*1.05;
        Output;
        Year=2;
        ProjectedSales=ProjectedSales*1.05;
        Output;
        Year=3;
        ProjectedSales=ProjectedSales*1.05;
        Output;
run;
```

Table: WORK.FORECAST | View: Column names | Filter: (none)

Columns

- Select all
- Region
- Product
- Subsidiary
- ProjectedSales
- Year

Total rows: 1185 Total columns: 5 Rows

	Region	Product	Subsidiary	ProjectedSales
1	Africa	Boot	Addis Ababa	$31,249
2	Africa	Boot	Addis Ababa	$32,812
3	Africa	Boot	Addis Ababa	$34,452
4	Africa	Men's Casual	Addis Ababa	$70,604
5	Africa	Men's Casual	Addis Ababa	$74,134
6	Africa	Men's Casual	Addis Ababa	$77,841

```
************************************************************;
* Controlling Row Output                                   *;
************************************************************;
* Syntax and Example                                       *;
*                                                          *;
*   Explicit output                                        *;
*      OUTPUT;
*;
*                                                          *;
*   Conditional output to multiple tables                  *;
*      DATA table1 table2...;
*;
*      ...                                                 *;
*         IF expression then OUTPUT table1;
*;
*         ELSE expression OUTPUT table2;
*;
*      ...                                                 *;
************************************************************;


data sales_high sales_low;
          set sashelp.shoes;


          if Sales>100000 then
```

```
                output sales_high;
        else
                output sales_low;
run;

*************************************************************;
*  p201d02.sas Demo                                         *;
*  1) Modify the DATA statement to create three tables      *;
*     named indian, atlantic, and pacific.                  *;
*  2) Modify the IF-THEN/ELSE conditional statements to     *;
*     write output to the appropriate table.                *;
*  3) Add a DROP statement to remove MaxWindMPH.            *;
*     Highlight the DATA step, run the selected code, and   *;
*     examine the output tables. Notice that MaxWindMPH     *;
*     has been dropped from all three tables.               *;
*************************************************************;

data indian atlantic pacific;
        set pg2.storm_summary;
        length Ocean $ 8;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;

        if substr(Basin, 2, 1)="I" then
                do;
                        Ocean="Indian";
                        output indian;
                end;
```

```
        else if substr(Basin, 2, 1)="A" then
                do;
                        Ocean="Atlantic";
                        output atlantic;
                end;
        else
                do;
                        Ocean="Pacific";
                        output pacific;
                end;
        drop MaxWindMPH;
run;

*Original;

data storm_complete;
        set pg2.storm_summary;
        length Ocean $ 8;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;

        if substr(Basin, 2, 1)="I" then
                Ocean="Indian";
        else if substr(Basin, 2, 1)="A" then
                Ocean="Atlantic";
        else
                Ocean="Pacific";
run;
```

Table: WORK.SALES_HIGH | View: Column names | Filter: (none)

Columns

- Select all
- Region
- Product
- Subsidiary
- Stores
- Sales
- Inventory
- Returns

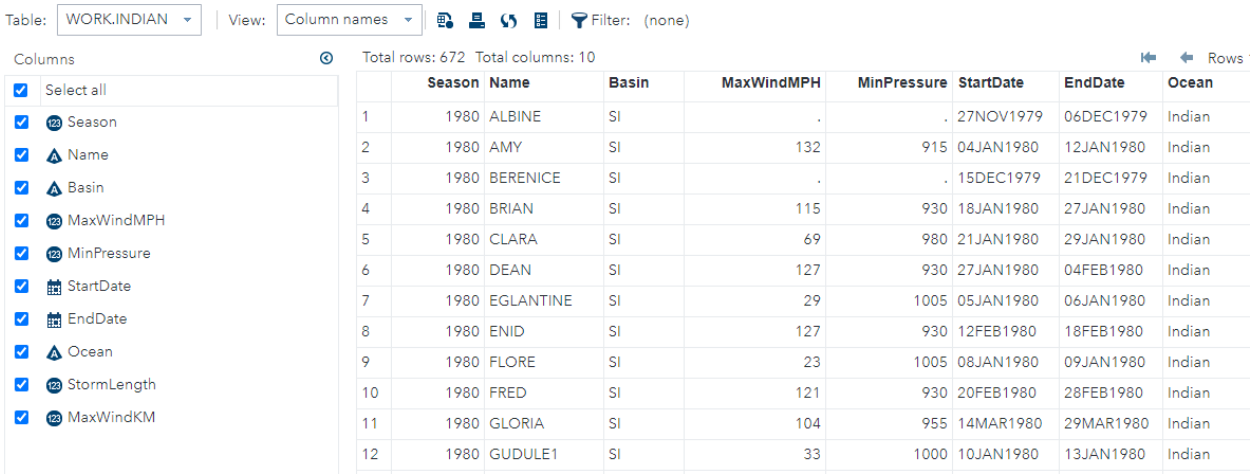
Total rows: 107 Total columns: 7 Rows 1-100

	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Women's Dress	Addis Ababa	12	$108,942	$311,017	$3,233
2	Africa	Men's Dress	Algiers	13	$123,743	$428,575	$3,621
3	Africa	Men's Casual	Cairo	25	$360,209	$1,063,251	$9,424
4	Africa	Women's Casual	Cairo	14	$328,474	$940,851	$10,124
5	Asia	Men's Dress	Seoul	7	$116,333	$251,803	$2,443
6	Asia	Slipper	Seoul	21	$149,013	$469,007	$2,941
7	Canada	Men's Dress	Montreal	11	$112,009	$355,170	$3,713
8	Canada	Slipper	Montreal	24	$135,305	$500,053	$3,395
9	Canada	Women's Dress	Montreal	12	$132,638	$483,637	$2,880

```
************************************************************;
* p201d03.sas Controlling Column Output                    *;
************************************************************;
* Syntax                                                   *;
*                                                          *;
*   table(DROP=col1 col2...)                               *;
*   table(KEEP=col1 col2...)                               *;
************************************************************;


************************************************************;
* Demo                                                     *;
* 1) Use the DROP= data set option to drop MaxWindMPH      *;
*    from the INDIAN table and MaxWindKM from the          *;
*    ATLANTIC table. Do not drop any columns from the      *;
*    PACIFIC table.                                        *;
* 2) Start the DATA step debugger. Note that MaxWindMPH    *;
*    and MaxWindKM are included in the PDV.                *;
* 3) Close the debugger, run the program, and examine      *;
*    the three output tables. MaxWindMPH has been          *;
*    dropped from the INDIAN table, MaxWindKM has been     *;
*    dropped from the ATLANTIC table, and the PACIFIC      *;
*    table has all columns.                                *;
* 4) Add a DROP= data set option in the SET statement to   *;
```

```
*    drop MinPressure. Start the debugger. Notice that   *;
*    MinPressure is not included in the PDV.            *;
*  5) Close the debugger, run the program, and examine    *;
*    the three output tables. Confirm that MinPressure   *;
*    has been dropped from each table.                  *;
************************************************************;

data indian(drop=MaxWindMPH) atlantic(drop=MaxWindKM) pacific;
        set pg2.storm_summary(drop=MinPressure);
        length Ocean $ 8;
        Basin=upcase(Basin);
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;
        if substr(Basin,2,1)="I" then do;
                Ocean="Indian";
                output indian;
        end;
        else if substr(Basin,2,1)="A" then do;
                Ocean="Atlantic";
                output atlantic;
        end;
        else do;
                Ocean="Pacific";
                output pacific;
        end;
run;
```

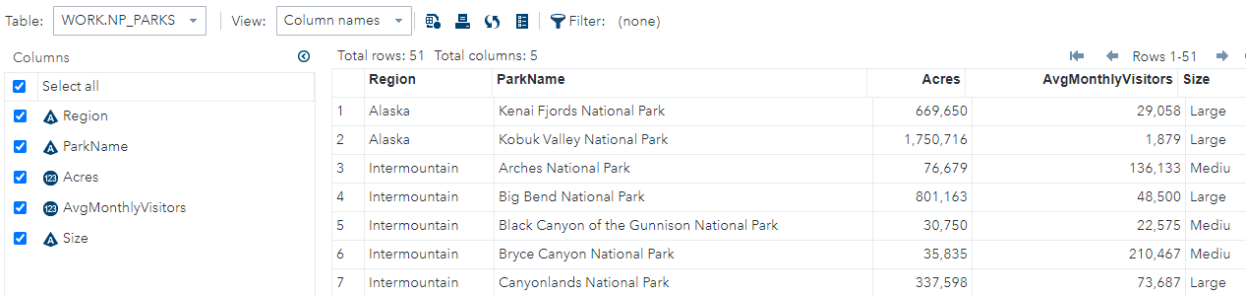
```
*original;
data indian atlantic pacific;
	set pg2.storm_summary;
	length Ocean $ 8;
	Basin=upcase(Basin);
	StormLength=EndDate-StartDate;
	MaxWindKM=MaxWindMPH*1.60934;
	if substr(Basin,2,1)="I" then do;
		Ocean="Indian";
		output indian;
	end;
	else if substr(Basin,2,1)="A" then do;
		Ocean="Atlantic";
		output atlantic;
	end;
	else do;
		Ocean="Pacific";
		output pacific;
	end;
run;
```



	Season	Name	Basin	MaxWindMPH	MinPressure	StartDate	EndDate	Ocean
1	1980	ALBINE	SI	.	.	27NOV1979	06DEC1979	Indian
2	1980	AMY	SI	132	915	04JAN1980	12JAN1980	Indian
3	1980	BERENICE	SI	.	.	15DEC1979	21DEC1979	Indian
4	1980	BRIAN	SI	115	930	18JAN1980	27JAN1980	Indian
5	1980	CLARA	SI	69	980	21JAN1980	29JAN1980	Indian
6	1980	DEAN	SI	127	930	27JAN1980	04FEB1980	Indian
7	1980	EGLANTINE	SI	29	1005	05JAN1980	06JAN1980	Indian
8	1980	ENID	SI	127	930	12FEB1980	18FEB1980	Indian
9	1980	FLORE	SI	23	1005	08JAN1980	09JAN1980	Indian
10	1980	FRED	SI	121	930	20FEB1980	28FEB1980	Indian
11	1980	GLORIA	SI	104	955	14MAR1980	29MAR1980	Indian
12	1980	GUDULE1	SI	33	1000	10JAN1980	13JAN1980	Indian

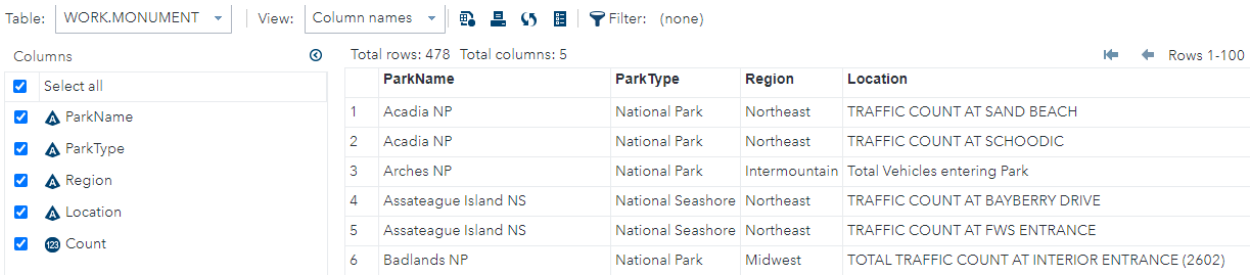
```
*************************************************************;
*  p201p02.sas LESSON 1, PRACTICE 2                         *;
*  a) Examine the program and answer the following          *;
*     questions.                                            *;
*     1) Which statements are compile-time only?            *;
*     2) What will be assigned for the length of Size?      *;
*  b) Run the program and examine the results.              *;
*  c) Modify the program to resolve the truncation of       *;
*     Size. Read the first 5 rows from the input table.     *;
*  d) Add PUTLOG statements to provide the following        *;
*     information in the log:                               *;
*     1) Immediately after the SET statement, write START   *;
*        DATA STEP ITERATION to the log as a color-coded    *;
*        note.                                              *;
*     2) After the Type= assignment statement, write the    *;
*        value of Type to the log.                          *;
*     3) At the end of the DATA step, write the contents    *;
*        of the PDV to the log.                             *;
*  e) Run the program and read the log to examine the       *;
*     messages written during execution.                    *;
*************************************************************;


data np_parks;
          set pg2.np_final(obs=5);
          putlog "NOTE: START DATA STEP ITERATION";
          length Size $ 6;
          keep Region ParkName AvgMonthlyVisitors Acres Size;
   where Type="PARK";
          format AvgMonthlyVisitors Acres comma10.;
```

```
   Type=propcase(Type);
   putlog Type=;
        AvgMonthlyVisitors=sum(DayVisits,Campers,OtherLodging)/12;
        if Acres<1000 then Size="Small";
        else if Acres<100000 then Size="Medium";
        else Size="Large";
        putlog _all_;
run;

/* Original */
data np_parks;
        set pg2.np_final;
        keep Region ParkName AvgMonthlyVisitors Acres Size;
   where Type="PARK";
        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;
```



	Region	ParkName	Acres	AvgMonthlyVisitors	Size
1	Alaska	Kenai Fjords National Park	669,650	29,058	Large
2	Alaska	Kobuk Valley National Park	1,750,716	1,879	Large
3	Intermountain	Arches National Park	76,679	136,133	Mediu
4	Intermountain	Big Bend National Park	801,163	48,500	Large
5	Intermountain	Black Canyon of the Gunnison National Park	30,750	22,575	Mediu
6	Intermountain	Bryce Canyon National Park	35,835	210,467	Mediu
7	Intermountain	Canyonlands National Park	337,598	73,687	Large

```
************************************************************;
*  p201p03.sas LESSON 1, PRACTICE 3                        *;
*  a) Modify the DATA step to create three tables:       *;
*     monument, park, and other. Use the value of        *;
*     ParkType as indicated above to determine which     *;
*     table the row is output to.                        *;
*  b) Drop ParkType from the monument and park tables.   *;
*     Drop Region from all three tables.                 *;
*  c) Submit the program and verify the output.          *;
************************************************************;

data monument(drop=ParkType) park(drop=ParkType) other;
        set pg2.np_yearlytraffic(drop=Region);
        if ParkType="National Monument" then do;
                output monument;
        end;
        else if ParkType="National Park" then do;
                output park;
        end;
        else
                output other;
run;



*original;
data monument;
        set pg2.np_yearlytraffic;
run;
```



	ParkName	ParkType	Region	Location
1	Acadia NP	National Park	Northeast	TRAFFIC COUNT AT SAND BEACH
2	Acadia NP	National Park	Northeast	TRAFFIC COUNT AT SCHOODIC
3	Arches NP	National Park	Intermountain	Total Vehicles entering Park
4	Assateague Island NS	National Seashore	Northeast	TRAFFIC COUNT AT BAYBERRY DRIVE
5	Assateague Island NS	National Seashore	Northeast	TRAFFIC COUNT AT FWS ENTRANCE
6	Badlands NP	National Park	Midwest	TOTAL TRAFFIC COUNT AT INTERIOR ENTRANCE (2602)

/*p201p04.sas Question 1

The pg2.np_2017 table contains monthly public use figures for national parks.

Create two tables, one that contains data about campers and one that contains data about visitors who used other lodging.

If necessary, start SAS Studio before you begin.

Reminder: If you restarted your SAS session, submit your libname.sas program to access the practice data.

Open a new program window and write a DATA step that reads the pg2.np_2017 table and creates temporary SAS tables named camping and lodging.

Compute a new column, CampTotal, that is the sum of CampingOther, CampingTent, CampingRV, and CampingBackcountry.

Format CampTotal so that values are displayed with commas.

In the camping table, include only rows that have CampTotal greater than zero.

The camping table should only have the columns ParkName, Month, DayVisits, and CampTotal columns.

In the lodging table, include only rows that have LodgingOther greater than zero.

The lodging table should only have the columns ParkName, Month, DayVisits, and LodgingOther columns.

Submit the program and verify the output.

Question 1

How many rows are in the camping table?

Question 2

How many rows are in the lodging table?

*/

data camping lodging;

```
        set pg2.np_2017;
        CampTotal = sum(CampingOther, CampingTent, CampingRV, CampingBackcountry);
        format CampTotal comma15.;
        if CampTotal>0 then do;
                output camping;
                keep ParkName Month DayVisits CampTotal;
        end;
        if LodgingOther>0 then do;
                output lodging;
                keep ParkName Month DayVisits LodgingOther;
        end;
run;

*Solution;
data camping(keep=ParkName Month DayVisits CampTotal)
   lodging(keep=ParkName Month DayVisits LodgingOther);
   set pg2.np_2017;
   CampTotal=sum(of Camping:);
   if CampTotal > 0 then output camping;
   if LodgingOther > 0 then output lodging;
   format CampTotal comma15.;
run;

*sample EDA;
data camping;
        set pg2.np_2017;
run;
```

Table: WORK.CAMPING | View: Column names | Filter: (none)

Columns

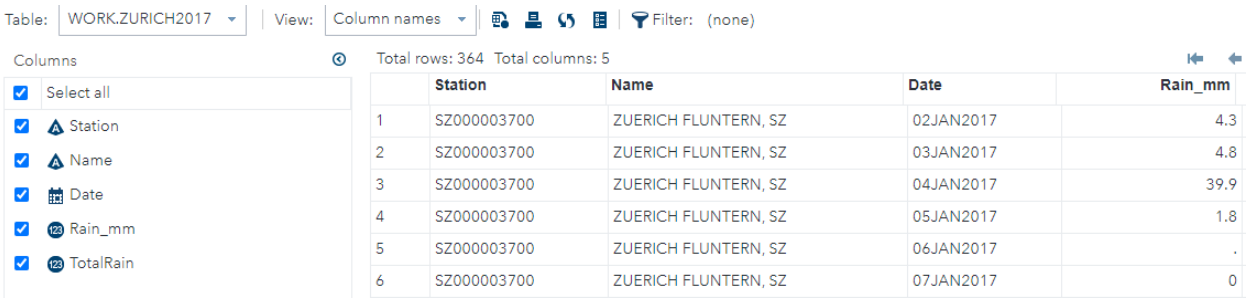
- Select all
- ParkName
- UnitCode
- ParkType
- Region
- State
- Year
- Month
- DayVisits
- LodgingOther
- CampingOther
- CampingTent
- CampingRV
- CampingBackcountry

Total rows: 4548 Total columns: 13 Rows 1-100

	ParkName	UnitCode	ParkType	Region	State	Year	Month	DayVisits
1	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	1	4,716
2	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	2	4,918
3	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	3	17,492
4	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	4	26,788
5	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	5	27,530
6	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	6	40,216
7	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	7	44,086
8	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	8	37,215
9	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	9	31,035
10	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	10	20,685
11	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	11	10,644
12	Abraham Lincoln Birthplace NHP	ABLI	National Historical Park	Southeast	KY	2017	12	4,255
13	Acadia NP	ACAD	National Park	Northeast	ME	2017	1	15,520
14	Acadia NP	ACAD	National Park	Northeast	ME	2017	2	14,342

```
**********************************************************;
* p202a01.sas  Activity 2.01                             *;
*  1) Modify the program to retain TotalRain and set the  *;
*     initial value to 0.                                *;
*  2) Run the program and examine the results. Why are    *;
*     all values for TotalRain missing after row 4?      *;
*  3) Change the assignment statement to use the SUM      *;
*     function instead of the plus symbol. Run the       *;
*     program again. Why are the results different?      *;
**********************************************************;


data zurich2017;
        set pg2.weather_zurich;
        *Add a RETAIN statement;
        retain TotalRain 0 DayNum 0;
        TotalRain=sum(TotalRain, Rain_mm);
        DayNum=sum(DayNum,1);
        DayNo+1;
run;
```

```
data zurich2017;
   set pg2.weather_zurich;
   YTDRain_mm+Rain_mm;
   DayNum+1;
run;



*Original;
data zurich2017;
        set pg2.weather_zurich;
        *Add a RETAIN statement;
        TotalRain=TotalRain+Rain_mm;
run;
```



	Station	Name	Date	Rain_mm
1	SZ000003700	ZUERICH FLUNTERN, SZ	02JAN2017	4.3
2	SZ000003700	ZUERICH FLUNTERN, SZ	03JAN2017	4.8
3	SZ000003700	ZUERICH FLUNTERN, SZ	04JAN2017	39.9
4	SZ000003700	ZUERICH FLUNTERN, SZ	05JAN2017	1.8
5	SZ000003700	ZUERICH FLUNTERN, SZ	06JAN2017	.
6	SZ000003700	ZUERICH FLUNTERN, SZ	07JAN2017	0

```
**********************************************************;
*  p202a03.sas Activity 2.03                             *;
*  1) Modify the PROC SORT step to sort the rows within  *;
*     each value of Basin by MaxWindMPH. Highlight the   *;
*     PROC SORT step and run the selected code. Which row *;
*     within each value of Basin represents the storm    *;
*     with the highest wind?                             *;
*  2) Add the following WHERE statement immediately after *;
*     the BY statement in the DATA step. The intent is to *;
```

```
*    include only the last row within each value of     *;
*    Basin. Does the program run successfully?          *;
*      where last.Basin=1;                              *;
************************************************************;

proc sort data=pg2.storm_2017 out=storm2017_sort;
        by Basin MaxWindMPH;
run;

*Original;
proc sort data=pg2.storm_2017 out=storm2017_sort;
        by Basin;
run;

data storm2017_max;
        set storm2017_sort;
        by Basin;
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;
run;
```

Table: WORK.STORM2017_SORT | View: Column names | Filter: (none)

Columns

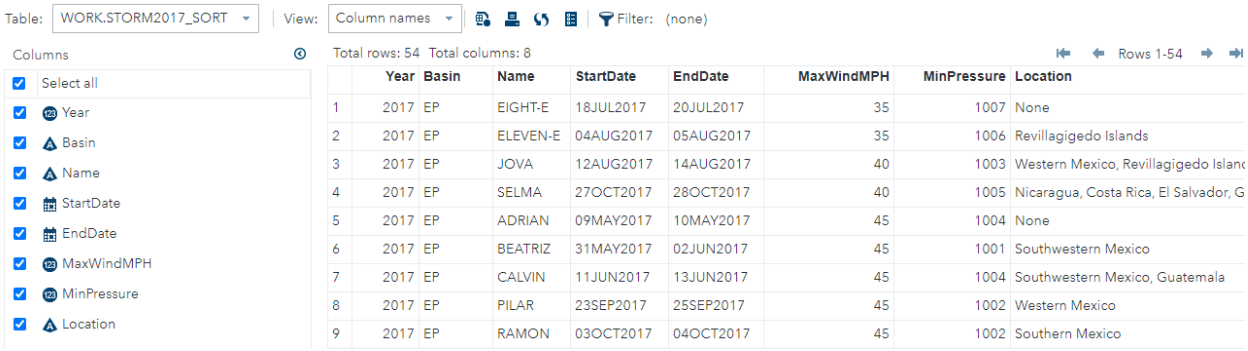
- Select all
- Year
- Basin
- Name
- StartDate
- EndDate
- MaxWindMPH
- MinPressure
- Location

Total rows: 54 Total columns: 8 Rows 1-54

	Year	Basin	Name	StartDate	EndDate	MaxWindMPH	MinPressure	Location
1	2017	EP	ADRIAN	09MAY2017	10MAY2017	45	1004	None
2	2017	EP	BEATRIZ	31MAY2017	02JUN2017	45	1001	Southwestern Mexico
3	2017	EP	CALVIN	11JUN2017	13JUN2017	45	1004	Southwestern Mexico, Guatemala
4	2017	EP	DORA	25JUN2017	28JUN2017	105	974	Southwestern Mexico, Revillagigedo
5	2017	EP	EUGENE	07JUL2017	12JUL2017	115	966	Baja California Peninsula, California
6	2017	EP	FERNANDA	12JUL2017	22JUL2017	145	947	Hawaii
7	2017	EP	GREG	17JUL2017	26JUL2017	60	1001	None
8	2017	EP	EIGHT-E	18JUL2017	20JUL2017	35	1007	None
9	2017	EP	HILARY	21JUL2017	31JUL2017	105	972	Southwestern Mexico
10	2017	EP	IRWIN	22JUL2017	01AUG2017	90	980	None

```
************************************************************;
*  p202a04.sas Activity 2.04                                *;
*  1) Change the WHERE statement to a subsetting IF         *;
*     statement and submit the program. How many rows are *;
*     included in the output table?                         *;
*  2) Move the subsetting IF statement just before the      *;
*     RUN statement and submit the program. How many rows *;
*     are included in the output table?                     *;
*  3) Consider the sequence of the statements in the        *;
*     execution phase. Where is the optimal placement of  *;
*     the subsetting IF statement?                          *;
************************************************************;
proc sort data=pg2.storm_2017 out=storm2017_sort;
        by Basin MaxWindMPH;
run;

data storm2017_max;
        set storm2017_sort;
        by Basin;
        if last.Basin=1;
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;
run;



data storm2017_max;
        set storm2017_sort;
        by Basin;
        StormLength=EndDate-StartDate;
```

```
        MaxWindKM=MaxWindMPH*1.60934;
        if last.Basin=1;
run;



*Original;
data storm2017_max;
        set storm2017_sort;
        by Basin;
        where last.Basin=1;
        StormLength=EndDate-StartDate;
        MaxWindKM=MaxWindMPH*1.60934;
run;
```



	Year	Basin	Name	StartDate	EndDate	MaxWindMPH	MinPressure	Location
1	2017	EP	EIGHT-E	18JUL2017	20JUL2017	35	1007	None
2	2017	EP	ELEVEN-E	04AUG2017	05AUG2017	35	1006	Revillagigedo Islands
3	2017	EP	JOVA	12AUG2017	14AUG2017	40	1003	Western Mexico, Revillagigedo Islanc
4	2017	EP	SELMA	27OCT2017	28OCT2017	40	1005	Nicaragua, Costa Rica, El Salvador, G
5	2017	EP	ADRIAN	09MAY2017	10MAY2017	45	1004	None
6	2017	EP	BEATRIZ	31MAY2017	02JUN2017	45	1001	Southwestern Mexico
7	2017	EP	CALVIN	11JUN2017	13JUN2017	45	1004	Southwestern Mexico, Guatemala
8	2017	EP	PILAR	23SEP2017	25SEP2017	45	1002	Western Mexico
9	2017	EP	RAMON	03OCT2017	04OCT2017	45	1002	Southern Mexico

```
************************************************;
*  p202a05.sas Activity 2.05                   *;
*    Add a subsetting IF statement to output   *;
*    only the final day of each month.         *;
************************************************;


data houston_monthly;
        set pg2.weather_houston;
        keep Date Month DailyRain MTDRain;
        by Month;
        if first.Month=1 then MTDRain=0;
        MTDRain+DailyRain;
        if last.Month=1;
run;
```

Table: WORK.HOUSTON_MONTHLY | View: Column names | Filter: (none)

Columns

- Select all
- Date
- Month
- DailyRain
- MTDRain

Total rows: 12 Total columns: 4 Rows 1-12

	Date	Month	DailyRain	MTDRain
1	31JAN2017	1	0	6.1
2	28FEB2017	2	0	2.42
3	31MAR2017	3	0	5.63
4	30APR2017	4	0	1.68
5	31MAY2017	5	0.09	2.41
6	30JUN2017	6	0	7.19
7	31JUL2017	7	0	6.29
8	31AUG2017	8	0	39.11
9	30SEP2017	9	0	1.23
10	31OCT2017	10	0.26	3.42
11	30NOV2017	11	0	0.5
12	31DEC2017	12	0.01	3.72

```
************************************************;
*  p202d01.sas Creating an Accumulating Column          *;
************************************************;
*  Demo                                *;
*  Refer to the course notes for detailed steps. *;
************************************************;


data houston_rain;
        set pg2.weather_houston;
        retain YTDRain 0;
        keep Date DailyRain YTDRain;
        YTDRain=YTDRain+DailyRain;
run;



*Original;
data houston_rain;
        set pg2.weather_houston;
        keep Date DailyRain YTDRain;
        YTDRain=YTDRain+DailyRain;
run;
```

Table: WORK.HOUSTON_RAIN | View: Column names | Filter:

Columns

- Select all
- Date
- DailyRain
- YTDRain

Total rows: 365 Total columns: 3

	Date	DailyRain	YTDRain
1	01JAN2017	0.01	0.01
2	02JAN2017	1.29	1.3
3	03JAN2017	0	1.3
4	04JAN2017	0	1.3
5	05JAN2017	0	1.3
6	06JAN2017	0.27	1.57
7	07JAN2017	0	1.57
8	08JAN2017	0	1.57

```
************************************************************;
* p202d03.sas Identifying the First and Last Row in Each Group     *;
************************************************************;
* Syntax                                  *;
*                                         *;
*   PROC SORT DATA=input-table                    *;
*      <OUT=output-table>;                        *;
*      BY <DESCENDING> col-name (s);              *;
*   RUN;                                  *;
*   DATA output-table;                            *;
*      SET input-table;                           *;
*      BY <DESCENDING> col-name (s);              *;
*   RUN;                                  *;
*   FIRST.bycol                             *;
*   LAST.bycol                              *;
************************************************************;
* Demo                                    *;
*    Refer to the course notes for detailed steps.      *;
```

**********;

```
proc sort data=pg2.storm_2017 out=storm2017_sort(keep=Basin Name);
        by Basin;
run;

data storm2017_max;
        set storm2017_sort;
        by Basin;
        First_Basin=first.basin;
        Last_Basin=last.basin;
run;

*Original;
data storm2017_max;
        set storm2017_sort;
        by Basin;
        *First_Basin=first.basin;
        *Last_Basin=last.basin;
run;
```

Table: WORK.STORM2017_SORT | View: Column names | Filter: (none)

Columns

- Select all
- Basin
- Name

Total rows: 54 Total columns: 2

	Basin	Name
1	EP	ADRIAN
2	EP	BEATRIZ
3	EP	CALVIN
4	EP	DORA
5	EP	EUGENE
6	EP	FERNANDA

```
************************************************************;
*  p202d03.sas Creating an Accumulating Column within Groups        *;
************************************************************;
*  Syntax and Example                          *;
*                                        *;
*    Subsetting IF statement:                    *;
*       IF expression;                          *;
*    FIRST.bycol                               *;
*    LAST.bycol                                *;
************************************************************;

proc sort data=pg2.storm_2017 out=storm2017_sort;
            by Basin;
run;

data storm2017_max;
   set storm2017_sort;
   by Basin;
   if last.Basin=1;
   StormLength=EndDate-StartDate;
   MaxWindKM=MaxWindMPH*1.60934;
run;

************************************************************;
*  Demo                                        *;
*  1) Highlight the DATA step and run the selected code.  *;
*     Notice that YTDRain is an accumulating column that  *;
*     creates a running total of DailyRain. Also notice   *;
*     that the data is sorted by Month and Date.          *;
```

```
*  2) Add a BY statement to process the rows by groups    *;
*     based on the values of Month.                       *;
*  3) Change the new accumulating column to MTDRain in    *;
*     the KEEP and sum statements.                        *;
*  4) Reset MTDRain to 0 each time that SAS reaches the   *;
*     first row within a new Month group. Highlight the   *;
*     DATA step and run the selected code.                *;
************************************************************;

data houston_monthly;
        set pg2.weather_houston;
        keep Date Month DailyRain MTDRain YTDRain;
        by Month;
        YTDRain+DailyRain;
        if first.Month=1 then MTDRain=0;
        MTDRain+DailyRain;
run;


*Original;
data houston_monthly;
        set pg2.weather_houston;
        keep Date Month DailyRain YTDRain;
        YTDRain+DailyRain;
run;
```

Table: WORK.STORM2017_SORT | View: Column names | Filter: (none)

Columns

- Select all
- Year
- Basin
- Name
- StartDate
- EndDate
- MaxWindMPH
- MinPressure
- Location

Total rows: 54 Total columns: 8 Rows 1-54

	Year	Basin	Name	StartDate	EndDate	MaxWindMPH	MinPressure	Location
1	2017	EP	ADRIAN	09MAY2017	10MAY2017	45	1004	None
2	2017	EP	BEATRIZ	31MAY2017	02JUN2017	45	1001	Southwestern Mexico
3	2017	EP	CALVIN	11JUN2017	13JUN2017	45	1004	Southwestern Mexico, Guatemala
4	2017	EP	DORA	25JUN2017	28JUN2017	105	974	Southwestern Mexico, Revillagiged
5	2017	EP	EUGENE	07JUL2017	12JUL2017	115	966	Baja California Peninsula, California
6	2017	EP	FERNANDA	12JUL2017	22JUL2017	145	947	Hawaii
7	2017	EP	GREG	17JUL2017	26JUL2017	60	1001	None
8	2017	EP	EIGHT-E	18JUL2017	20JUL2017	35	1007	None
9	2017	EP	HILARY	21JUL2017	31JUL2017	105	972	Southwestern Mexico

```
**************************************************************;
*  p202p01.sas LESSON 2, PRACTICE 1                          *;
*  a) Open the PG2.NP_YEARLYTRAFFIC table. Notice the        *;
*     Count column records the number of cars that have      *;
*     passed through a particular Location.                  *;
*  b) Modify the DATA step to create a column, totTraffic,*;
*     that is the running total of Count.                    *;
*  b) Keep the ParkName, Location, Count, and                *;
*     totTraffic columns in the output table.                *;
*  c) Format totTraffic so values are displayed with         *;
*     commas.                                                *;
**************************************************************;


data totalTraffic;
   set pg2.np_yearlyTraffic;
   keep ParkName Location Count totTraffic;
        totTraffic+Count;
        format totTraffic comma15.;
run;


/*Question 1
The pg2.np_yearlytraffic table contains annual traffic counts at locations in national parks.
```

Parks are classified as one of five types: National Monument, National Park, National Preserve, National River, and National Seashore.

Suppose you want to create separate running totals for two of the park types. If necessary, start SAS Studio before you begin.

Reminder: If you restarted your SAS session, submit your libname.sas program to access the practice data.

Create a table, parkTypeTraffic, from the pg2.np_yearlytraffic table. Use the following specifications:

Read only the rows from the input table where ParkType is National Monument or National Park.

Create two new columns named MonumentTraffic and ParkTraffic.

The value of each column should be increased by the value of Count for that park type.

Format the new columns so that values are displayed with commas.

Create a listing report of parkTypeTraffic as follows:

Use Accumulating Traffic Totals for Park Types as the report title.

Display the columns in this order: ParkType, ParkName, Location, Count, MonumentTraffic, and ParkTraffic.

Submit the program and view the results.

In the output table, which row number has the first nonzero value for MonumentTraffic?

What is the value of ParkTraffic in row 10? Note: Don't use commas in your answer.

*/

```
data parkTypeTraffic;
        set pg2.np_yearlytraffic;
        where ParkType="National Monument" or ParkType="National Park";
        if ParkType="National Monument" then
                MonumentTraffic+Count;
        else
```

```
                    ParkTraffic+Count;
        format MonumentTraffic ParkTraffic comma15.;
        keep ParkType ParkName Location Count MonumentTraffic ParkTraffic;
run;

data work.parktypetraffic;
   set pg2.np_yearlyTraffic;
   where ParkType in ("National Monument", "National Park");
   if ParkType = 'National Monument' then MonumentTraffic+Count;
   else ParkTraffic+Count;
   format MonumentTraffic ParkTraffic comma15.;
run;

title 'Accumulating Traffic Totals for Park Types';
proc print data=work.parktypetraffic;
   var ParkType ParkName Location Count MonumentTraffic
      ParkTraffic;
run;
title;

*Original;
data totalTraffic;
   set pg2.np_yearlyTraffic;

run;
```

Accumulating Traffic Totals for Park Types

Obs	ParkType	ParkName	Location	Count	MonumentTraffic	ParkTraffic
1	National Park	Acadia NP	TRAFFIC COUNT AT SAND BEACH	377,759	0	377,759
2	National Park	Acadia NP	TRAFFIC COUNT AT SCHOODIC	113,601	0	491,360
3	National Park	Arches NP	Total Vehicles entering Park	569,658	0	1,061,018
4	National Park	Badlands NP	TOTAL TRAFFIC COUNT AT INTERIOR ENTRANCE (2602)	120,215	0	1,181,233
5	National Park	Badlands NP	TOTAL TRAFFIC COUNT AT NORTHEAST ENTRANCE (2601)	171,792	0	1,353,025
6	National Park	Badlands NP	TOTAL TRAFFIC COUNT AT PINNACLES ENTRANCE (2603)	125,856	0	1,478,881
7	National Monument	Bandelier NM	TRAFFIC COUNT AT ENTRANCE	0	0	1,478,881
8	National Park	Big Bend NP	TRAFFIC COUNT AT ROUTE 11-PERS.GAP	59,595	0	1,538,476
9	National Park	Big Bend NP	TRAFFIC COUNT AT ROUTE 13-MAVERICK	96,153	0	1,634,629
10	National Park	Black Canyon of the Gunnison NP	TRAFFIC COUNT AT NORTH RIM DRIVE COUNTER	12,345	0	1,646,974
11	National Park	Black Canyon of the Gunnison NP	TRAFFIC COUNT AT SOUTH RIM DRIVE COUNTER	113,373	0	1,760,347
12	National Monument	Booker T. Washington NM	TRAFFIC COUNT AT MAIN ENTRANCE	8,181	8,181	1,760,347
13	National Park	Bryce Canyon NP	TRAFFIC COUNT AT ENTRANCE STATION	623,241	8,181	2,383,588
14	National Monument	Cabrillo NM	TRAFFIC COUNT AT MAIN ENTRANCE	412,691	420,872	2,383,588
15	National Monument	Canyon de Chelly NM	TRAFFIC COUNT AT MAIN ENTRANCE	1,699,696	2,120,568	2,383,588
16	National Monument	Canyon de Chelly NM	TRAFFIC COUNT AT NORTH RIM	407,994	2,528,562	2,383,588
17	National Monument	Canyon de Chelly NM	TRAFFIC COUNT AT SOUTH RIM	408,277	2,936,839	2,383,588
18	National Park	Canyonlands NP	TRAFFIC COUNT AT ELEPHANT HILL	1,877	2,936,839	2,385,465
19	National Park	Canyonlands NP	TRAFFIC COUNT AT HORSESHOE CANYON	2,531	2,936,839	2,387,996
20	National Park	Canyonlands NP	TRAFFIC COUNT AT ISLAND PROPER ENTRANCE	203,973	2,936,839	2,591,969

```
************************************************************;
*  p202p04.sas LESSON 2, PRACTICE 4                        *;
*  a) 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 ParkType.              *;
*     4) Format TypeCount so values are displayed with     *;
*        commas.                                           *;
*     5) Keep only the ParkType and TypeCount columns.     *;
*  c) Run the program and confirm 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.                   *;
************************************************************;

proc sort data=pg2.np_yearlyTraffic
```

```
        out=sortedTraffic(keep=ParkType ParkName
                          Location Count);
    *Insert BY statement;
    by ParkType ParkName;
run;

data TypeTraffic;
    set work.sortedTraffic;
            by ParkType;
            if first.ParkType=1 then TypeCount=0;
            TypeCount+Count;
            format TypeCount comma15.;
            keep ParkType TypeCount;
run;

data TypeTraffic;
    set work.sortedTraffic;
            by ParkType;
            if first.ParkType=1 then TypeCount=0;
            TypeCount+Count;
            format TypeCount comma15.;
            keep ParkType TypeCount;
            if last.ParkType;
run;

*Original;
proc sort data=pg2.np_yearlyTraffic
        out=sortedTraffic(keep=ParkType ParkName
                          Location Count);
```

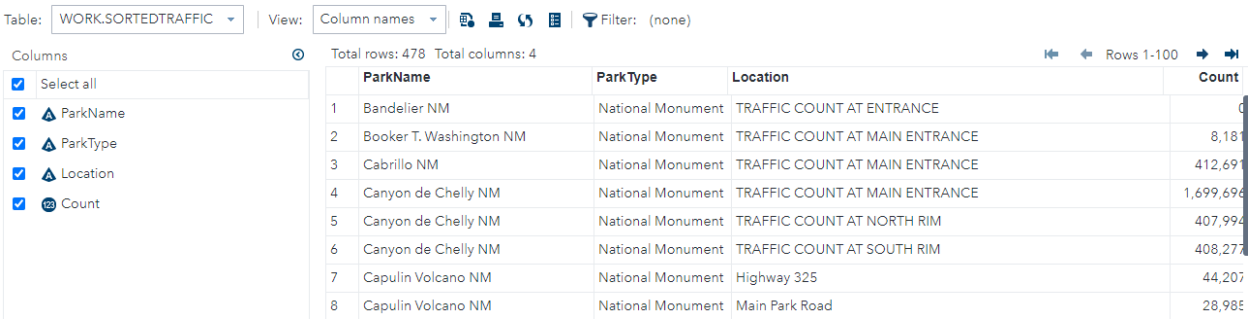
*Insert BY statement;

run;

data TypeTraffic;

set ;

run;



	ParkName	ParkType	Location	Count
1	Bandelier NM	National Monument	TRAFFIC COUNT AT ENTRANCE	
2	Booker T. Washington NM	National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	8,181
3	Cabrillo NM	National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	412,691
4	Canyon de Chelly NM	National Monument	TRAFFIC COUNT AT MAIN ENTRANCE	1,699,696
5	Canyon de Chelly NM	National Monument	TRAFFIC COUNT AT NORTH RIM	407,994
6	Canyon de Chelly NM	National Monument	TRAFFIC COUNT AT SOUTH RIM	408,277
7	Capulin Volcano NM	National Monument	Highway 325	44,207
8	Capulin Volcano NM	National Monument	Main Park Road	28,985

/* p202p05.sas

Level 2 Practice: Generating an Accumulating Column within Multiple Groups

TOTAL POINTS 2

1.

Question 1

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.

If necessary, start SAS Studio before you begin.

Write a program to create a sorted copy of sashelp.shoes that is ordered by Region and Product.

Write a 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.

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.

Submit the program and verify that TotalProfit is accurate.

How many rows are in the profitsummary table?

Question 2

Modify the DATA step to include only the last row for each Region and Product combination.

Include only the columns Region, Product, and TotalProfit in the output table.

Format TotalProfit as a currency value.

Submit the program and examine the output data.

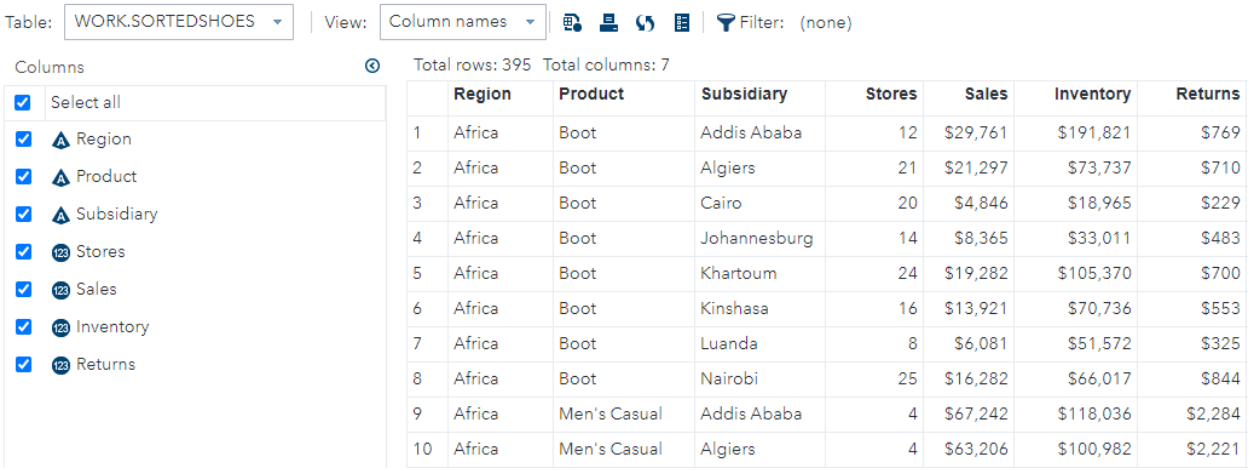
How many rows are in the profitsummary table?

*/

```
proc sort data=sashelp.shoes
        out=sortedShoes;
   *Insert BY statement;
   by Region Product;
run;

data profitSummary;
   set work.sortedShoes;
   Profit=Sales-Returns;
            by Region Product;
            if first.Region=1 and first.Product=1 then TotalProfit=0;
            TotalProfit+Profit;
            format TotalProfit comma15.;
            keep Region Product Profit TotalProfit;
run;


data profitSummary;
```

```
   set work.sortedShoes;
   Profit=Sales-Returns;
         by Region Product;
         if first.Region=1 and first.Product=1 then TotalProfit=0;
         TotalProfit+Profit;
         if last.Product;
         format TotalProfit dollar12.;
         keep Region Product TotalProfit;
run;

*Solution;
proc sort data=sashelp.shoes out=sort_shoes;
   by Region Product;
run;

data profitsummary;
   set sort_shoes;
   by Region Product;
   Profit=Sales-Returns;
   if first.Product then TotalProfit=0;
   TotalProfit+Profit;
   format TotalProfit dollar12.;
run;

data profitsummary;
   set sort_shoes;
   by Region Product;
   Profit=Sales-Returns;
   if first.Product then TotalProfit=0;
```

```
   TotalProfit+Profit;
   if last.Product=1;
   keep Region Product TotalProfit;
   format TotalProfit dollar12.;
run;
```



	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Africa	Boot	Algiers	21	$21,297	$73,737	$710
3	Africa	Boot	Cairo	20	$4,846	$18,965	$229
4	Africa	Boot	Johannesburg	14	$8,365	$33,011	$483
5	Africa	Boot	Khartoum	24	$19,282	$105,370	$700
6	Africa	Boot	Kinshasa	16	$13,921	$70,736	$553
7	Africa	Boot	Luanda	8	$6,081	$51,572	$325
8	Africa	Boot	Nairobi	25	$16,282	$66,017	$844
9	Africa	Men's Casual	Addis Ababa	4	$67,242	$118,036	$2,284
10	Africa	Men's Casual	Algiers	4	$63,206	$100,982	$2,221