SP203-204 Doing More with SAS Programming

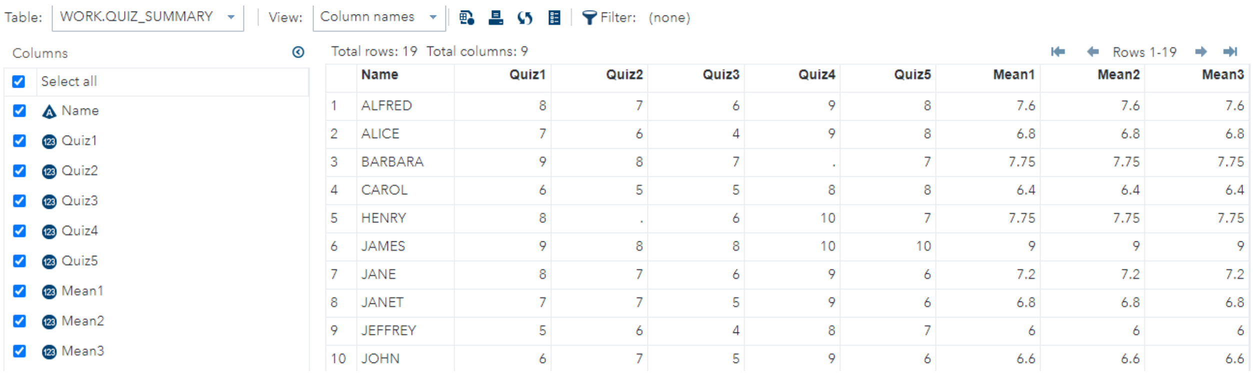
203 Understanding SAS Functions and Routines

204 Creating and Using Custom Formats

```
***********************************************************;
*  p203a01.sas Activity 3.01                              *;
*  1) Run the program. Why does the DATA step fail?       *;
*     Correct the error by overwriting the value of the   *;
*     column Name in uppercase.                           *;
*  2) Examine the expressions for Mean1, Mean2, and       *;
*     Mean3. Each one is a method for specifying a list   *;
*     of columns as arguments in a function. Run the      *;
*     program and verify that the values in these three   *;
*     columns are the same.                               *;
*  3) In the expression for Mean2, delete the keyword OF  *;
*     and run the program. What do the values in Mean2    *;
*     represent?                                          *;
***********************************************************;
```

```
data quiz_summary;
        set pg2.class_quiz;
        name=upcase(Name);
        Mean1=mean(Quiz1, Quiz2, Quiz3, Quiz4, Quiz5);
        /* Numbered Range: col1-coln where n is a sequential number */
        Mean2=mean(of Quiz1-Quiz5);
        /* Name Prefix: all columns that begin with the specified character string */
        Mean3=mean(of Q:);
run;
```

```
*Original;

data quiz_summary;

        set pg2.class_quiz;

        upcase(Name);

        Mean1=mean(Quiz1, Quiz2, Quiz3, Quiz4, Quiz5);

        /* Numbered Range: col1-coln where n is a sequential number */

        Mean2=mean(of Quiz1-Quiz5);

        /* Name Prefix: all columns that begin with the specified character string */

        Mean3=mean(of Q:);

run;
```

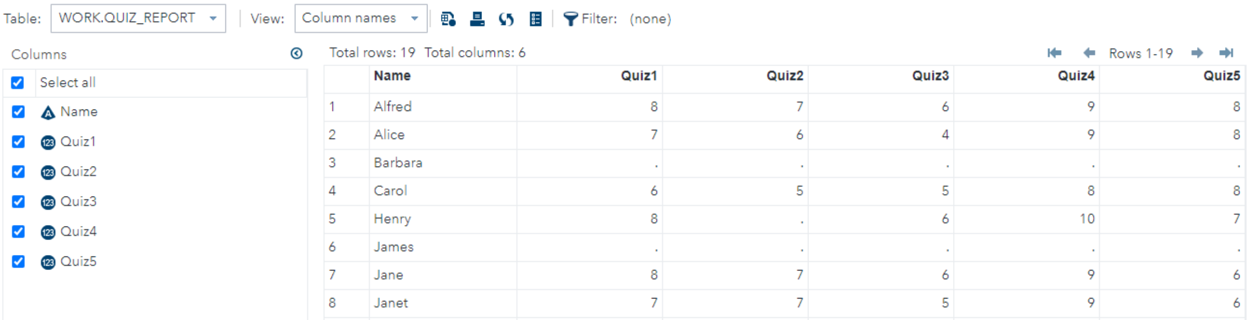


Total rows: 19 Total columns: 9

	Name	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5	Mean1	Mean2	Mean3
1	ALFRED	8	7	6	9	8	7.6	7.6	7.6
2	ALICE	7	6	4	9	8	6.8	6.8	6.8
3	BARBARA	9	8	7	.	7	7.75	7.75	7.75
4	CAROL	6	5	5	8	8	6.4	6.4	6.4
5	HENRY	8	.	6	10	7	7.75	7.75	7.75
6	JAMES	9	8	8	10	10	9	9	9
7	JANE	8	7	6	9	6	7.2	7.2	7.2
8	JANET	7	7	5	9	6	6.8	6.8	6.8
9	JEFFREY	5	6	4	8	7	6	6	6
10	JOHN	6	7	5	9	6	6.6	6.6	6.6

```
***********************************************************;

*  p203a02.sas Activity 3.02                          *;

*  1) Examine the program and notice that all quiz scores *;

*     for two students are changed to missing values.    *;

*     Highlight the first DATA step and submit the       *;

*     selected code.                                  *;

*  2) In a web browser, access SAS Help at             *;

*     http://support.sas.com/documentation. In the Syntax *;

*     Shortcuts section, click the Programming: SAS 9.4   *;

*     and Viya link.                                  *;

*  3) In the Syntax � Quick Links section, click CALL     *;
*    Routines. Use the documentation to read about the   *;
*    CALL MISSING routine.                          *;
*  4) Simplify the second DATA step by using CALL MISSING *;
*    to assign missing values for the two students' quiz *;
*    scores. Run the step.                          *;
*********************************************************;

/* Step 1 */
data quiz_report;
   set pg2.class_quiz;
          if Name in("Barbara", "James") then do;
                    Quiz1=.;
                    Quiz2=.;
                    Quiz3=.;
                    Quiz4=.;
                    Quiz5=.;
          end;
run;

/* Step 4 */
data quiz_report;
   set pg2.class_quiz;
          if Name in("Barbara", "James") then call missing(of Quiz1-Quiz5);
run;

/* Original Step 4 */
data quiz_report;
   set pg2.class_quiz;
          if Name in("Barbara", "James") then call missing(/*provide arguments*/);
```

```
run;
```



Total rows: 19 Total columns: 6

	Name	Quiz1	Quiz2	Quiz3	Quiz4	Quiz5
1	Alfred	8	7	6	9	8
2	Alice	7	6	4	9	8
3	Barbara	.	.	.	.	.
4	Carol	6	5	5	8	8
5	Henry	8	.	6	10	7
6	James	.	.	.	.	.
7	Jane	8	7	6	9	6
8	Janet	7	7	5	9	6

```
***********************************************************;
*  p203a03.sas Activity 3.03                              *;
*  1) Notice that the expressions for WindAvg1 and        *;
*     WindAvg2 are the same. Run the program and examine  *;
*     the output table.                                   *;
*  2) Modify the WindAvg1 expression to use the ROUND     *;
*     function to round values to the nearest tenth (.1). *;
*  3) Add a FORMAT statement to format WindAvg2 with the  *;
*     5.1 format. Run the program. What is the difference *;
*     between using a function and a format?              *;
***********************************************************;

data wind_avg;
	set pg2.storm_top4_wide;
	WindAvg1=round(mean(of Wind1-Wind4), .1);
	WindAvg2=mean(of Wind1-Wind4);
	format WindAvg2 5.1;
run;

*Original;
data wind_avg;
```

```
        set pg2.storm_top4_wide;
        WindAvg1=mean(of Wind1-Wind4);
        WindAvg2=mean(of Wind1-Wind4);
run;
```

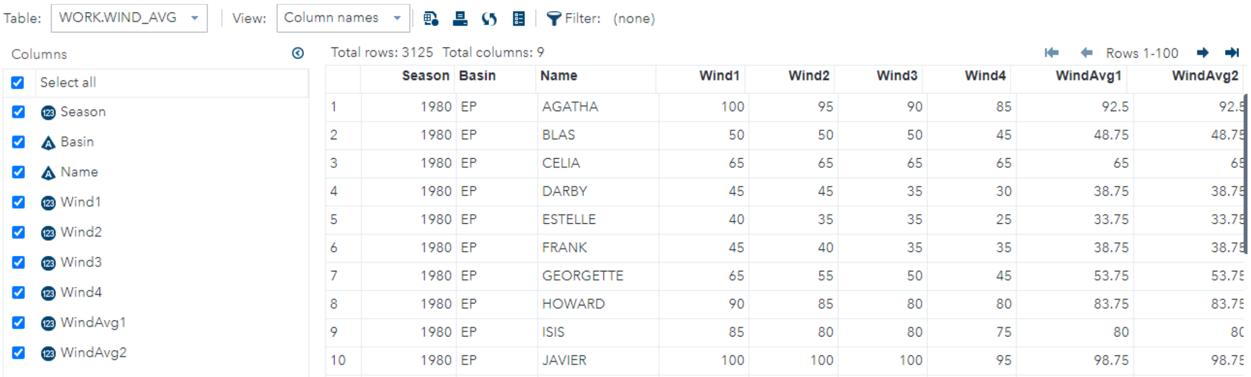


Table: WORK.WIND_AVG | View: Column names | Filter: (none)

Columns

☑ Select all
☑ Season
☑ Basin
☑ Name
☑ Wind1
☑ Wind2
☑ Wind3
☑ Wind4
☑ WindAvg1
☑ WindAvg2

Total rows: 3125 Total columns: 9 Rows 1-100

	Season	Basin	Name	Wind1	Wind2	Wind3	Wind4	WindAvg1	WindAvg2
1	1980	EP	AGATHA	100	95	90	85	92.5	92.5
2	1980	EP	BLAS	50	50	50	45	48.75	48.75
3	1980	EP	CELIA	65	65	65	65	65	65
4	1980	EP	DARBY	45	45	35	30	38.75	38.75
5	1980	EP	ESTELLE	40	35	35	25	33.75	33.75
6	1980	EP	FRANK	45	40	35	35	38.75	38.75
7	1980	EP	GEORGETTE	65	55	50	45	53.75	53.75
8	1980	EP	HOWARD	90	85	80	80	83.75	83.75
9	1980	EP	ISIS	85	80	80	75	80	80
10	1980	EP	JAVIER	100	100	100	95	98.75	98.75

```
**********************************************************;
*  p203a04.sas Activity 3.04                           *;
*  1) Notice that the INTCK function does not include the *;
*     optional method argument, so the default discrete   *;
*     method is used to calculate the number of weekly    *;
*     boundaries (ending each Saturday) between StartDate *;
*     and EndDate.                                     *;
*  2) Run the program and examine rows 8 and 9. Both      *;
*     storms were two days, but why are the values        *;
*     assigned to Weeks different?                     *;
*  3) Add 'c' as the fourth argument in the INTCK         *;
*     function to use the continuous method. Run the      *;
*     program. Are the values for Weeks in rows 8 and 9   *;
*     different?                                       *;
**********************************************************;
*  Syntax Help                                         *;
*     INTCK('interval', start-date, end-date, <'method'>) *;
*        Interval: WEEK, MONTH, YEAR, WEEKDAY, HOUR, etc.*;
*       Method: DISCRETE (D) or CONTINUOUS (C)          *;
*****************************************************;

data storm_length;
        set pg2.storm_final(obs=10);
        keep Season Name StartDate Enddate StormLength Weeks;
        Weeks=intck('week', StartDate, EndDate);
run;

data storm_length;
        set pg2.storm_final(obs=10);
        keep Season Name StartDate Enddate StormLength Weeks;
        Weeks=intck('week', StartDate, EndDate, 'C');
run;
```

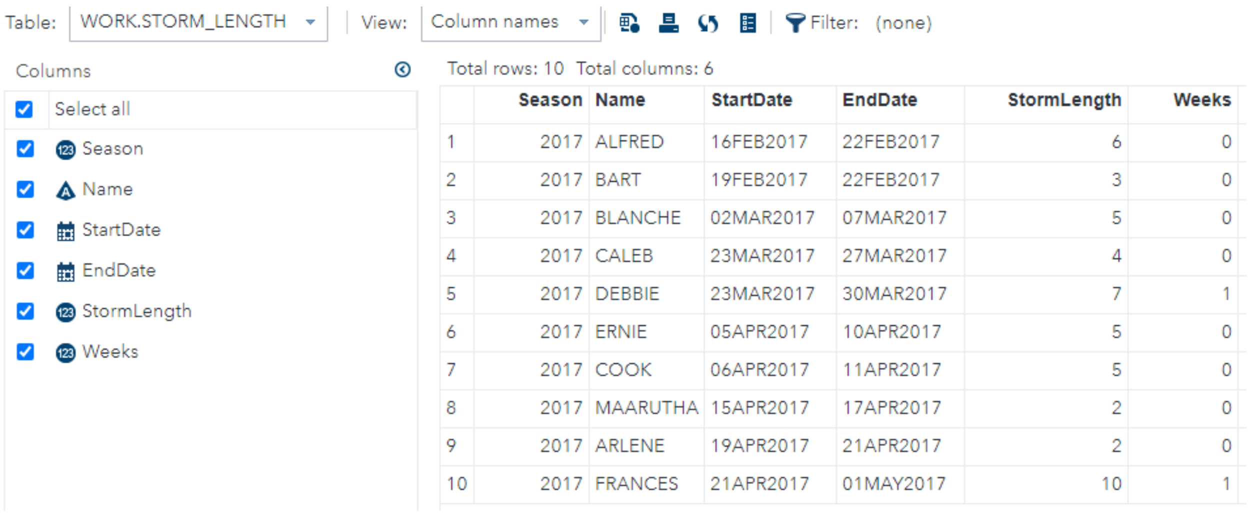


Table: WORK.STORM_LENGTH   View: Column names   Filter: (none)

Columns

☑ Select all
☑ Season
☑ Name
☑ StartDate
☑ EndDate
☑ StormLength
☑ Weeks

Total rows: 10 Total columns: 6

	Season	Name	StartDate	EndDate	StormLength	Weeks
1	2017	ALFRED	16FEB2017	22FEB2017	6	0
2	2017	BART	19FEB2017	22FEB2017	3	0
3	2017	BLANCHE	02MAR2017	07MAR2017	5	0
4	2017	CALEB	23MAR2017	27MAR2017	4	0
5	2017	DEBBIE	23MAR2017	30MAR2017	7	1
6	2017	ERNIE	05APR2017	10APR2017	5	0
7	2017	COOK	06APR2017	11APR2017	5	0
8	2017	MAARUTHA	15APR2017	17APR2017	2	0
9	2017	ARLENE	19APR2017	21APR2017	2	0
10	2017	FRANCES	21APR2017	01MAY2017	10	1

```
*****************************************************;
*  p203a06.sas Activity 3.06                          *;
*  1) Complete the NewLocation assignment statement to    *;
*     use the COMPBL function to read Location and        *;
*     convert each occurrence of two or more consecutive  *;
*     blanks into a single blank.                     *;
*  2) Complete the NewStation assignment to use the      *;
*     COMPRESS function with Station as the only          *;
*     argument. Run the program. Which characters are     *;
*     removed in the NewStation column?                   *;
*  3) Add a second argument in the COMPRESS function to   *;
*     remove both the space and hyphen. Both characters   *;
*     should be enclosed in a single set of quotation     *;
*     marks. Run the program.                             *;
****************************************************;
*  Syntax Help                                            *;
*     COMPBL(string)                                      *;
*     COMPRESS (string <, characters>)                    *;
****************************************************;


data weather_japan_clean;
   set pg2.weather_japan;
   NewLocation=compbl(location);
   NewStation=compress(station,"- ");
run;
```

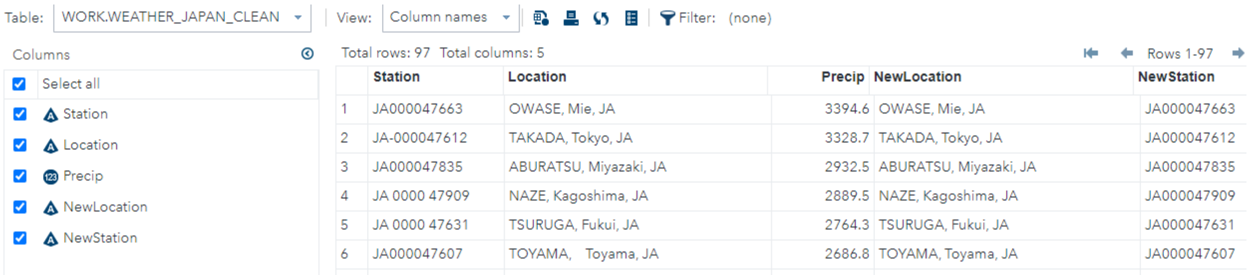


Table: WORK.WEATHER_JAPAN_CLEAN   View: Column names   Filter: (none)

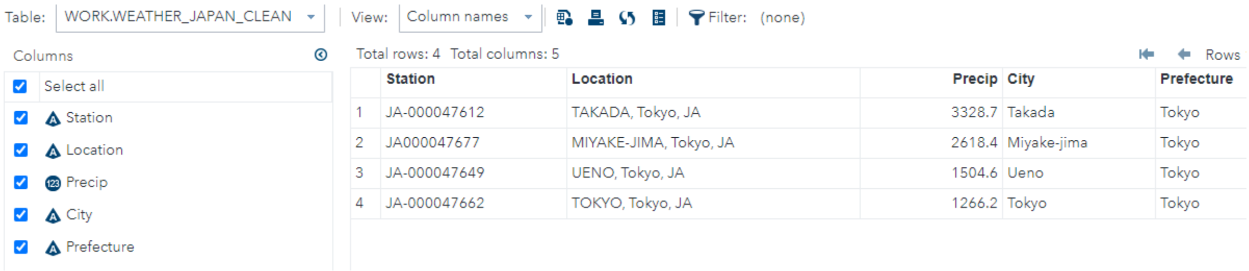
Columns

☑ Select all
☑ Station
☑ Location
☑ Precip
☑ NewLocation
☑ NewStation

Total rows: 97 Total columns: 5   Rows 1-97

	Station	Location	Precip	NewLocation	NewStation
1	JA000047663	OWASE, Mie, JA	3394.6	OWASE, Mie, JA	JA000047663
2	JA-000047612	TAKADA, Tokyo, JA	3328.7	TAKADA, Tokyo, JA	JA000047612
3	JA000047835	ABURATSU, Miyazaki, JA	2932.5	ABURATSU, Miyazaki, JA	JA000047835
4	JA 0000 47909	NAZE, Kagoshima, JA	2889.5	NAZE, Kagoshima, JA	JA000047909
5	JA 0000 47631	TSURUGA, Fukui, JA	2764.3	TSURUGA, Fukui, JA	JA000047631
6	JA000047607	TOYAMA, Toyama, JA	2686.8	TOYAMA, Toyama, JA	JA000047607

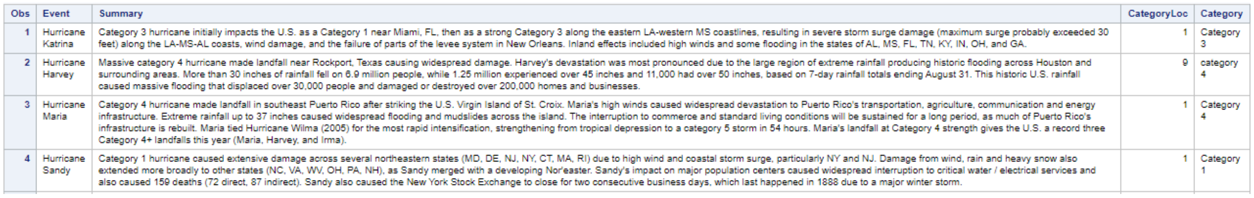
```
****************************************************;
*  p203a07.sas Activity 3.07                              *;
*  1) Notice the subsetting IF statement that writes rows *;
*     to output only if Prefecture is Tokyo. Run the      *;
*     program and notice that the output table does not   *;
*    include any rows.                    *;
*  2) Either use the DATA step debugger in Enterprise    *;
*    Guide or uncomment the PUTLOG statement to view the *;
*    values of Prefecture as the step executes. Why is   *;
*    the subsetting IF condition always false?        *;
*  3) Modify the program to correct the logic error. Run  *;
*    the program and confirm that four rows are        *;
*    returned.                              *;
********************************************************;


data weather_japan_clean;
        set pg2.weather_japan;
        Location=compbl(Location);
        City=propcase(scan(Location, 1, ','), ' ');
        Prefecture=strip(scan(Location, 2, ','));
        putlog Prefecture $quote20.;
        if Prefecture="Tokyo";
run;
```



Total rows: 4 Total columns: 5

	Station	Location	Precip	City	Prefecture
1	JA-000047612	TAKADA, Tokyo, JA	3328.7	Takada	Tokyo
2	JA000047677	MIYAKE-JIMA, Tokyo, JA	2618.4	Miyake-jima	Tokyo
3	JA-000047649	UENO, Tokyo, JA	1504.6	Ueno	Tokyo
4	JA-000047662	TOKYO, Tokyo, JA	1266.2	Tokyo	Tokyo

```
********************************************************;
*  p203a08.sas Activity 3.08                  *;
*  1) Notice that the assignment statement for        *;
*    CategoryLoc uses the FIND function to search for   *;
*    category within each value of the Summary column.   *;
*    Run the program.                       *;
*  2) Examine the PROC PRINT report. Why is CategoryLoc   *;
*    equal to 0 in row 1? Why is CategoryLoc equal to 0  *;
*    in row 15?                                *;
*  3) Modify the FIND function to make the search case    *;
*    insensitive. Uncomment the IF-THEN statement to     *;
*    create a new column named Category. Run the program *;
*    and examine the results.                       *;
********************************************************;
*  Syntax Help                                 *;
*    FIND(string, substring <, 'modifiers'>)            *;
*       Modifiers:                             *;
*          'I'=case insensitive search                *;
*          'T'=trim leading and training blanks from   *;
*                                 string and substring                  *;
********************************************************;


data storm_damage2;
	set pg2.storm_damage;
	drop Date Cost;
	CategoryLoc=find(Summary, 'category', 'I');
	if CategoryLoc > 0 then Category=substr(Summary,CategoryLoc, 10);
run;


proc print data=storm_damage2;
	var Event Summary Cat:;
run;
```



Obs	Event	Summary	CategoryLoc	Category
1	Hurricane Katrina	Category 3 hurricane initially impacts the U.S. as a Category 1 near Miami, FL, then as a strong Category 3 along the eastern LA-western MS coastlines, resulting in severe storm surge damage (maximum surge probably exceeded 30 feet) along the LA-MS-AL coasts, wind damage, and the failure of parts of the levee system in New Orleans. Inland effects included high winds and some flooding in the states of AL, MS, FL, TN, KY, IN, OH, and GA.	1	Category 3
2	Hurricane Harvey	Massive category 4 hurricane made landfall near Rockport, Texas causing widespread damage. Harvey's devastation was most pronounced due to the large region of extreme rainfall producing historic flooding across Houston and surrounding areas. More than 30 inches of rainfall fell on 6.9 million people, while 1.25 million experienced over 45 inches and 11,000 had over 50 inches, based on 7-day rainfall totals ending August 31. This historic U.S. rainfall caused massive flooding that displaced over 30,000 people and damaged or destroyed over 200,000 homes and businesses.	9	category 4
3	Hurricane Maria	Category 4 hurricane made landfall in southeast Puerto Rico after striking the U.S. Virgin Island of St. Croix. Maria's high winds caused widespread devastation to Puerto Rico's transportation, agriculture, communication and energy infrastructure. Extreme rainfall up to 37 inches caused widespread flooding and mudslides across the island. The interruption to commerce and standard living conditions will be sustained for a long period, as much of Puerto Rico's infrastructure is rebuilt. Maria tied Hurricane Wilma (2005) for the most rapid intensification, strengthening from tropical depression to a category 5 storm in 54 hours. Maria's landfall at Category 4 strength gives the U.S. a record three Category 4+ landfalls this year (Maria, Harvey, and Irma).	1	Category 4
4	Hurricane Sandy	Category 1 hurricane caused extensive damage across several northeastern states (MD, DE, NJ, NY, CT, MA, RI) due to high wind and coastal storm surge, particularly NY and NJ. Damage from wind, rain and heavy snow also extended more broadly to other states (NC, VA, WV, OH, PA, NH), as Sandy merged with a developing Nor'easter. Sandy's impact on major population centers caused widespread interruption to critical water / electrical services and also caused 159 deaths (72 direct, 87 indirect). Sandy also caused the New York Stock Exchange to close for two consecutive business days, which last happened in 1888 due to a major winter storm.	1	Category 1

```
*****************************************************;
*  p203a09.sas Activity 3.09                        *;
*  1) Examine the assignment statements that use the CAT  *;
*     and CATS functions to create StormID1 and StormID2. *;
*     Run the program. How do the two columns differ?     *;
*  2) Add an assignment statement to create StormID3 that *;
*     uses the CATX function to concatenate Name, Season, *;
*     and Day with a hyphen inserted between each value.  *;
*     Run the program.                            *;
*  3) Modify the StormID2 assignment statement to insert  *;
*     a hyphen only between Name and Season.            *;
*****************************************************;

data storm_id;
	set pg2.storm_final;
	keep StormID: ;
	Day=StartDate-intnx('year', StartDate, 0);
	StormID1=cat(Name, Season, Day);
	StormID2=cats(Name, '-', Season, Day);
	StormID3=catx('-', Name, Season, Day);
run;
```

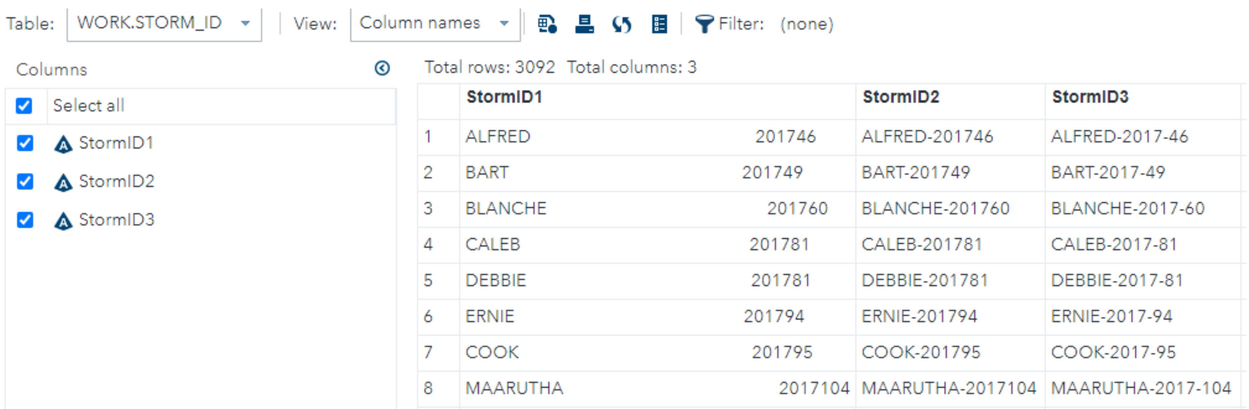


Table: WORK.STORM_ID | View: Column names | Filter: (none)

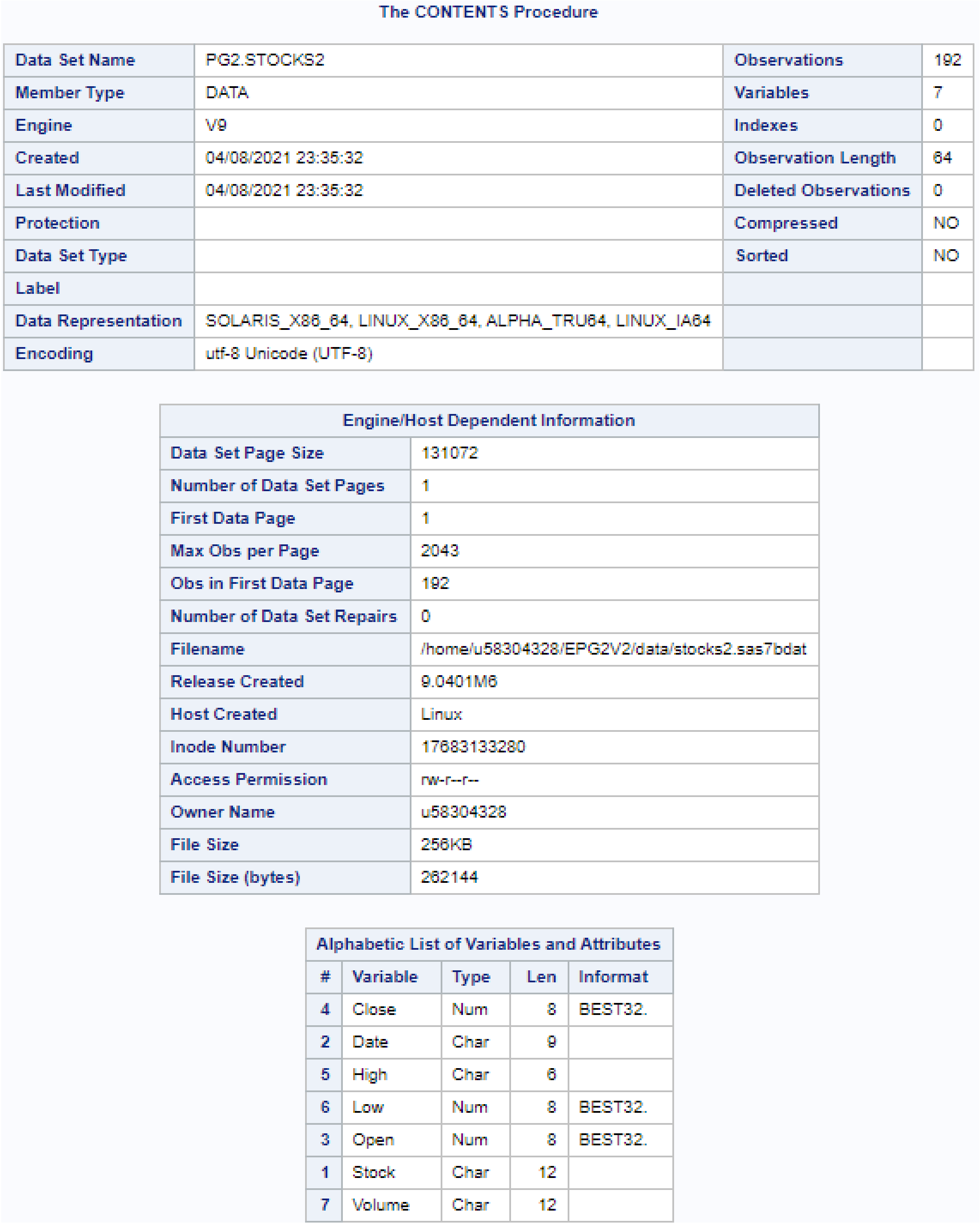
Columns

☑ Select all
☑ StormID1
☑ StormID2
☑ StormID3

Total rows: 3092 Total columns: 3

	StormID1	StormID2	StormID3
1	ALFRED 201746	ALFRED-201746	ALFRED-2017-46
2	BART 201749	BART-201749	BART-2017-49
3	BLANCHE 201760	BLANCHE-201760	BLANCHE-2017-60
4	CALEB 201781	CALEB-201781	CALEB-2017-81
5	DEBBIE 201781	DEBBIE-201781	DEBBIE-2017-81
6	ERNIE 201794	ERNIE-201794	ERNIE-2017-94
7	COOK 201795	COOK-201795	COOK-2017-95
8	MAARUTHA 2017104	MAARUTHA-2017104	MAARUTHA-2017-104

```
***********************************************************;
*  p203a10.sas Activity 3.10                              *;
*  1) Highlight the PROC CONTENTS step and run the        *;
*     selected code. What is the type of High, Low, and   *;
*     Volume?                                             *;
*  2) Highlight the DATA and PROC PRINT steps and run the *;
*     selected code. Notice that although High is a       *;
*     character column, the Range column is accurately    *;
*     calculated.                                         *;
*  3) Open the log. Read the note printed immediately     *;
*     after the DATA step.                                *;
*  4) Uncomment the DailyVol assignment statement and run *;
*     the program. Is DailyVol created successfully?      *;
***********************************************************;

proc contents data=pg2.stocks2;
run;

data work.stocks2;
  set pg2.stocks2;
  Range=High-Low;
  *DailyVol=Volume/30;
run;


proc print data=stocks2(obs=10);
run;
```



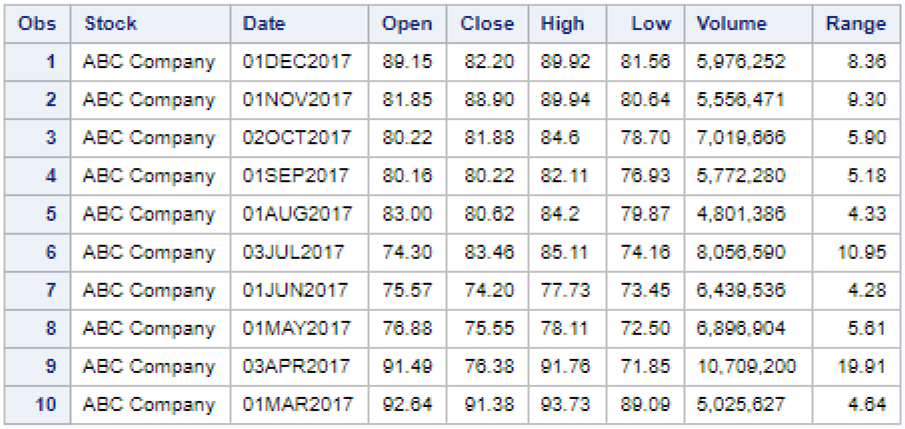
The CONTENTS Procedure

Data Set Name	PG2.STOCKS2	Observations	192
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	04/08/2021 23:35:32	Observation Length	64
Last Modified	04/08/2021 23:35:32	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	1
First Data Page	1
Max Obs per Page	2043
Obs in First Data Page	192
Number of Data Set Repairs	0
Filename	/home/u58304328/EPG2V2/data/stocks2.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	17683133280
Access Permission	rw-r--r--
Owner Name	u58304328
File Size	256KB
File Size (bytes)	262144

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Informat
4	Close	Num	8	BEST32.
2	Date	Char	9	
5	High	Char	6	
6	Low	Num	8	BEST32.
3	Open	Num	8	BEST32.
1	Stock	Char	12	
7	Volume	Char	12	



Obs	Stock	Date	Open	Close	High	Low	Volume	Range
1	ABC Company	01DEC2017	89.15	82.20	89.92	81.56	5,976,252	8.36
2	ABC Company	01NOV2017	81.85	88.90	89.94	80.64	5,556,471	9.30
3	ABC Company	02OCT2017	80.22	81.88	84.6	78.70	7,019,666	5.90
4	ABC Company	01SEP2017	80.16	80.22	82.11	76.93	5,772,280	5.18
5	ABC Company	01AUG2017	83.00	80.62	84.2	79.87	4,801,386	4.33
6	ABC Company	03JUL2017	74.30	83.46	85.11	74.16	8,056,590	10.95
7	ABC Company	01JUN2017	75.57	74.20	77.73	73.45	6,439,536	4.28
8	ABC Company	01MAY2017	76.88	75.55	78.11	72.50	6,896,904	5.61
9	ABC Company	03APR2017	91.49	76.38	91.76	71.85	10,709,200	19.91
10	ABC Company	01MAR2017	92.64	91.38	93.73	89.09	5,025,627	4.64

```
*********************************************************;
*  p203a11.sas Activity 3.11                            *;
*  1) Examine and run the program. In the output table,  *;
*     verify that Date2 is created as numeric. Notice    *;
*     that the table contains a character column named   *;
*     Volume.                                            *;
*  2) Add an assignment statement to create a column     *;
*     named Volume2. Use the INPUT function to read      *;
*     Volume using the COMMA12. informat. Run the program *;
*     and verify that Volume2 is created as a numeric    *;
*     column.                                            *;
*  3) In the assignment statement, change Volume2 to     *;
*     Volume so that you update the value of the existing *;
*     column.                                            *;
*  4) Run the program and notice that Volume is still    *;
*     character. Why is the assignment statement not     *;
*     changing the column type?                          *;
*********************************************************;

data work.stocks2;
   set pg2.stocks2;
```

```
  Date2=input(Date,date9.);
          *Add an assignment statement;
          Volume2=input(Volume,comma12.);
          *Volume=Volume2;
run;
```

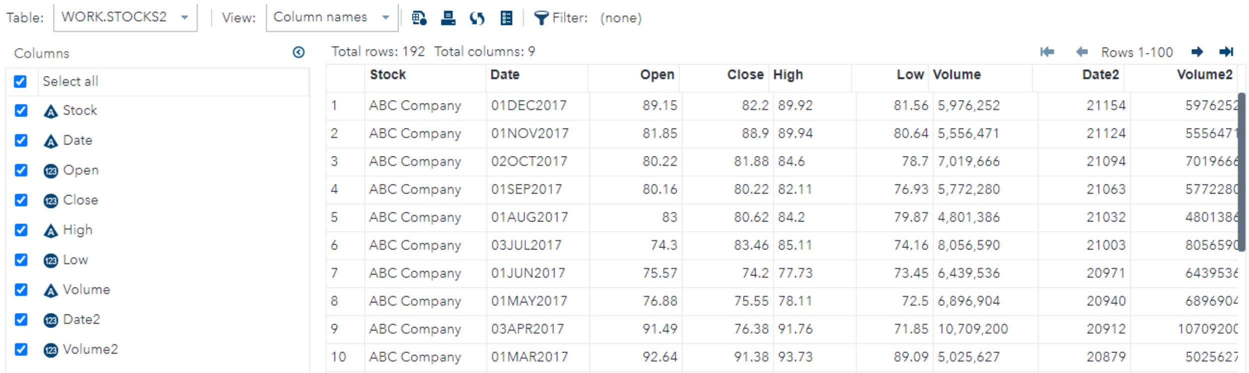


Table: WORK.STOCKS2 | View: Column names | Filter: (none)

Columns

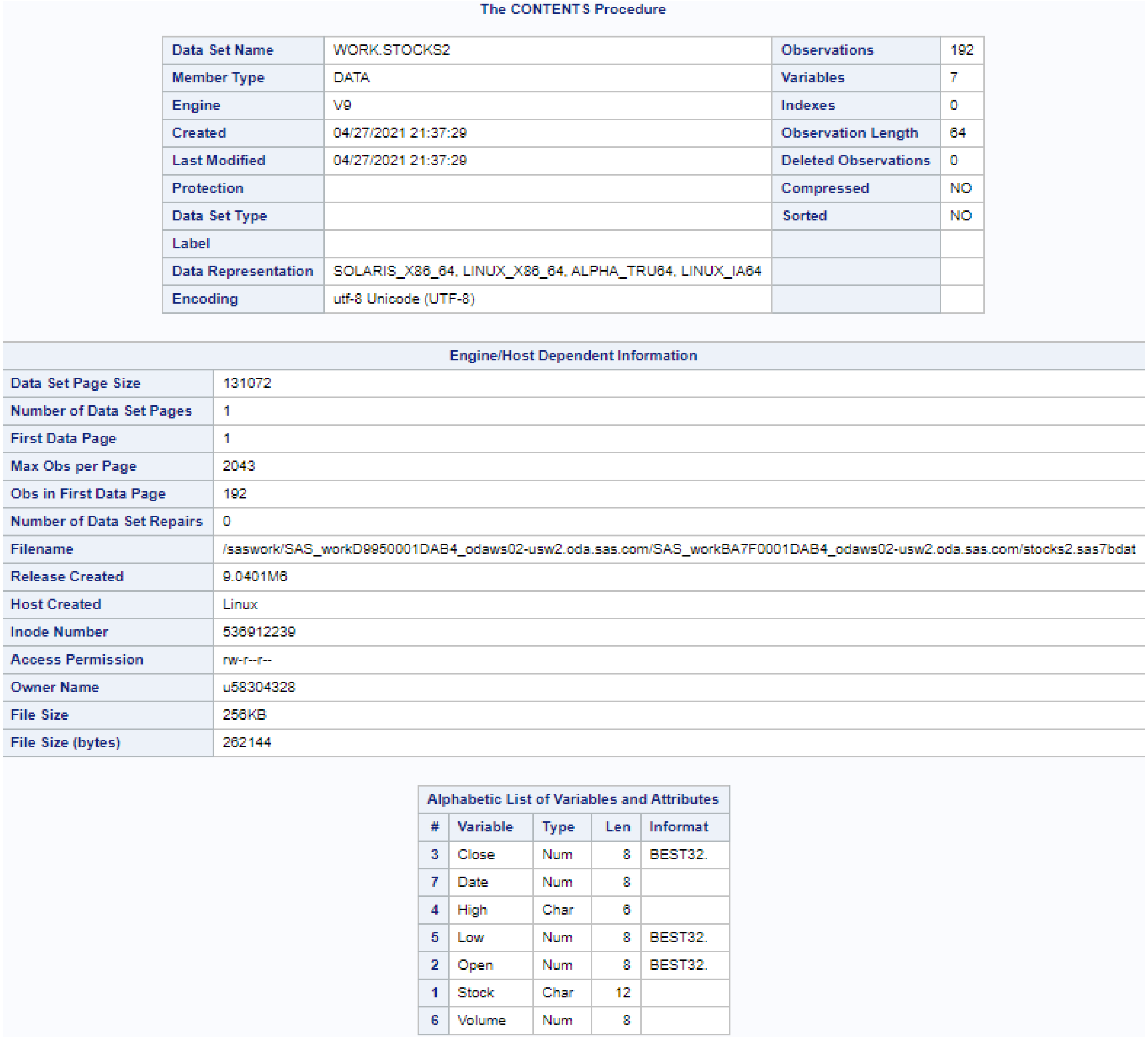
☑ Select all
☑ Stock
☑ Date
☑ Open
☑ Close
☑ High
☑ Low
☑ Volume
☑ Date2
☑ Volume2

Total rows: 192 Total columns: 9 Rows 1-100

	Stock	Date	Open	Close	High	Low	Volume	Date2	Volume2
1	ABC Company	01DEC2017	89.15	82.2	89.92	81.56	5,976,252	21154	5976252
2	ABC Company	01NOV2017	81.85	88.9	89.94	80.64	5,556,471	21124	5556471
3	ABC Company	02OCT2017	80.22	81.88	84.6	78.7	7,019,666	21094	7019666
4	ABC Company	01SEP2017	80.16	80.22	82.11	76.93	5,772,280	21063	5772280
5	ABC Company	01AUG2017	83	80.62	84.2	79.87	4,801,386	21032	4801386
6	ABC Company	03JUL2017	74.3	83.46	85.11	74.16	8,056,590	21003	8056590
7	ABC Company	01JUN2017	75.57	74.2	77.73	73.45	6,439,536	20971	6439536
8	ABC Company	01MAY2017	76.88	75.55	78.11	72.5	6,896,904	20940	6896904
9	ABC Company	03APR2017	91.49	76.38	91.76	71.85	10,709,200	20912	10709200
10	ABC Company	01MAR2017	92.64	91.38	93.73	89.09	5,025,627	20879	5025627

```
*****************************************************;
*  p203a13.sas Activity 3.13                        *;
*  1) Add to the RENAME= option to rename the input      *;
*     column Date as CharDate.                      *;
*  2) Add an assignment statement to create a numeric     *;
*     column Date from the character column CharDate. The *;
*     values of CharDate are stored as 01JAN2018.        *;
*  3) Modify the DROP statement to eliminate all columns  *;
*     that begin with Char from the output table.        *;
*  4) Run the program and verify that Volume and Date are *;
*     numeric columns.                              *;
*****************************************************;


data stocks2;
   set pg2.stocks2(rename=(Volume=CharVolume Date=CharDate));
   Volume=input(CharVolume,comma12.);
   Date=input(CharDate, date9.);
```

```
  drop Char:;
run;


proc contents data=stocks2;
run;
```



The CONTENTS Procedure

Data Set Name	WORK.STOCKS2	Observations	192
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	04/27/2021 21:37:29	Observation Length	64
Last Modified	04/27/2021 21:37:29	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	1
First Data Page	1
Max Obs per Page	2043
Obs in First Data Page	192
Number of Data Set Repairs	0
Filename	/saswork/SAS_workD9950001DAB4_odaws02-usw2.oda.sas.com/SAS_workBA7F0001DAB4_odaws02-usw2.oda.sas.com/stocks2.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	536912239
Access Permission	rw-r--r--
Owner Name	u58304328
File Size	256KB
File Size (bytes)	262144

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Informat
3	Close	Num	8	BEST32.
7	Date	Num	8	
4	High	Char	6	
5	Low	Num	8	BEST32.
2	Open	Num	8	BEST32.
1	Stock	Char	12	
6	Volume	Num	8	

```
***********************************************************;
*  p203d01.sas Using Numeric Functions                *;
***********************************************************;
*  Syntax                                   *;
*                                          *;
*   RAND ('distribution', parameter1, ...parameterk)    *;
*   LARGEST (k, value-1 <, value-2 ...>)             *;
*   ROUND (number <,rounding-unit>)                 *;
***********************************************************;

***********************************************************;
*  Demo                                     *;
*  1) Copy and paste the Quiz1st assignment statement    *;
*    twice and modify the statements to create columns   *;
*    named Quiz2nd and Quiz3rd.                     *;
*  2) Create a new column named Top3Avg that uses the     *;
*    MEAN function with the top three quiz scores as the *;
*    arguments.                              *;
*  3) Add Name in the DROP statement.                 *;
*  4) Before the SET statement, create a new column named *;
*    StudentID. Use the RAND function with 'INTEGER' as  *;
*    the first argument. This generates random integers  *;
*    between the values specified in the second and      *;
*    third arguments. To create a four-digit number, use *;
*    1000 as the lower limit and 9999 as the upper       *;
*    limit. Highlight the DATA step and run the selected *;
*    code.                                  *;
*  5) Modify the Top3Avg assignment statement to use the  *;
*    ROUND function to round the values returned by the  *;
*    MEAN function to the nearest integer. Highlight the *;
*    DATA step and run the selected code.              *;
*  6) Add a second argument in the ROUND function to     *;
*    round values to the nearest .1. Highlight the DATA  *;
*    step and run the selected code.                  *;
********************************************************;

data quiz_analysis;
            StudentID=rand("integer", 1000, 9999);
            set pg2.class_quiz;
            drop Quiz1-Quiz5 name;
   Quiz1st=largest(1, of Quiz1-Quiz5);
   Quiz2nd=largest(2, of Quiz1-Quiz5);
   Quiz3rd=largest(3, of Quiz1-Quiz5);
   Top3Avg=round(mean(Quiz1st,Quiz2nd,Quiz3rd), .1);
run;
```

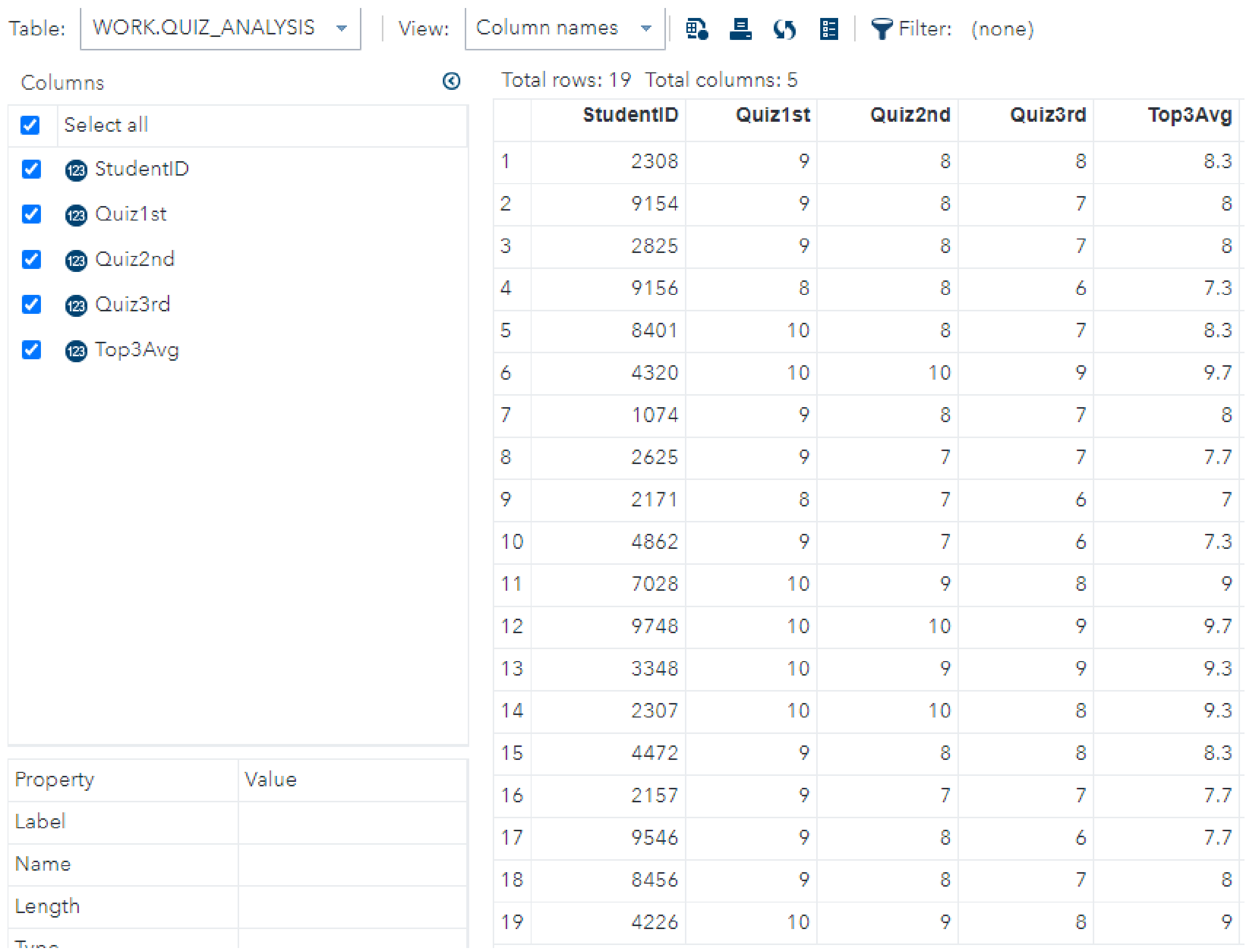


Table: WORK.QUIZ_ANALYSIS View: Column names Filter: (none)

Columns

☑ Select all
☑ StudentID
☑ Quiz1st
☑ Quiz2nd
☑ Quiz3rd
☑ Top3Avg

Property	Value
Label	
Name	
Length	
Type	

Total rows: 19 Total columns: 5

	StudentID	Quiz1st	Quiz2nd	Quiz3rd	Top3Avg
1	2308	9	8	8	8.3
2	9154	9	8	7	8
3	2825	9	8	7	8
4	9156	8	8	6	7.3
5	8401	10	8	7	8.3
6	4320	10	10	9	9.7
7	1074	9	8	7	8
8	2625	9	7	7	7.7
9	2171	8	7	6	7
10	4862	9	7	6	7.3
11	7028	10	9	8	9
12	9748	10	10	9	9.7
13	3348	10	9	9	9.3
14	2307	10	10	8	9.3
15	4472	9	8	8	8.3
16	2157	9	7	7	7.7
17	9546	9	8	6	7.7
18	8456	9	8	7	8
19	4226	10	9	8	9

```
*******************************************************;
*  p203d02.sas Shifting Date Values                    *;
*******************************************************;
*  Syntax                         *;
*                                *;
*   INTNX ('interval', start, increment <, 'alignment'>) *;
*******************************************************;


*******************************************************;
*  Demo                           *;
*  1) Notice that the AssessmentDate column is created by *;
*     using the INTNX function to shift each Date value.  *;
*     Highlight the DATA step and run the selected code.  *;
*    Notice that each Date value has been shifted to the *;
*    first day of the same month.                    *;
*  2) To see the impact of the various arguments in the   *;
*    INTNX function, modify the arguments as directed.   *;
*    Highlight the DATA step, run the selected code, and *;
*    examine the results after each modification.       *;
*    a) Change the increment value to 2.               *;
*    b) Change the increment value to -1. Add 'end' as   *;
*       the optional fourth argument to specify         *;
*       alignment.                              *;
*    c) Change the alignment argument to 'middle'.      *;
*  3) Write an assignment statement to create a new      *;
*    column named Anniversary that is the date of the    *;
*    10-year anniversary for each storm. Add 'same' as   *;
*    the optional fourth argument to specify alignment.  *;
*    Keep the new column in the output table and use the *;
*    DATE9. format to display the values.              *;
********************************************************;
*same month;
data storm_damage2;
	set pg2.storm_damage;
	keep Event Date AssessmentDate;
	AssessmentDate=intnx('month', Date, 0);
   format Date AssessmentDate date9.;
run;
*2 months ahead;
data storm_damage2;
	set pg2.storm_damage;
	keep Event Date AssessmentDate;
```

```
            AssessmentDate=intnx('month', Date, 2);
   format Date AssessmentDate date9.;
run;
*last day of previous month;
data storm_damage2;
            set pg2.storm_damage;
            keep Event Date AssessmentDate;
            AssessmentDate=intnx('month', Date, -1, 'end');
   format Date AssessmentDate date9.;
run;
*middle of the month of previous month;
data storm_damage2;
            set pg2.storm_damage;
            keep Event Date AssessmentDate;
            AssessmentDate=intnx('month', Date, -1, 'middle');
   format Date AssessmentDate date9.;
run;
*10 year anniversary;
data storm_damage2;
            set pg2.storm_damage;
            keep Event Date AssessmentDate Anniversary;
            AssessmentDate=intnx('month', Date, -1, 'middle');
            Anniversary=intnx('year', Date, 10, 'same');
   format Date AssessmentDate Anniversary date9.;
run;
```

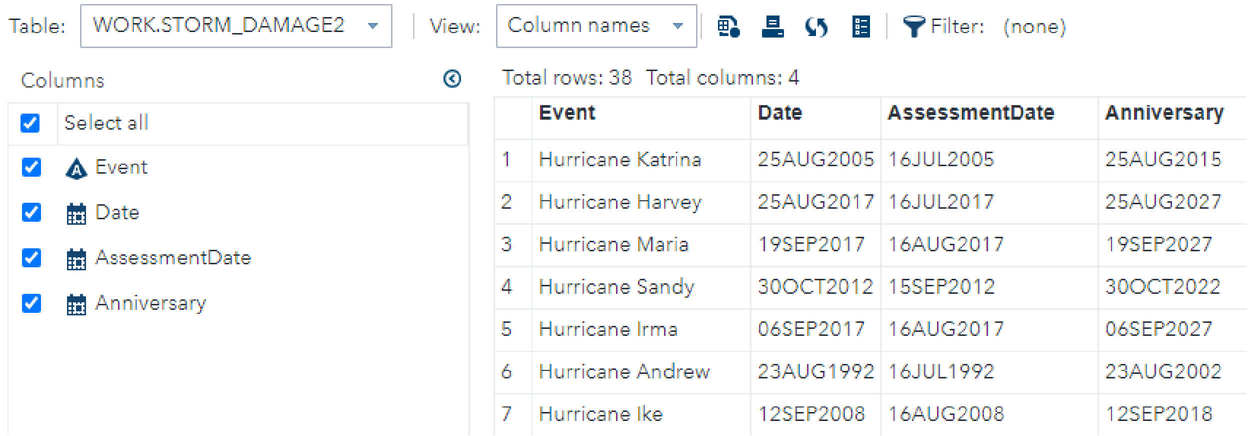


Table: WORK.STORM_DAMAGE2 View: Column names Filter: (none)

Columns

☑ Select all
☑ Event
☑ Date
☑ AssessmentDate
☑ Anniversary

Total rows: 38 Total columns: 4

	Event	Date	AssessmentDate	Anniversary
1	Hurricane Katrina	25AUG2005	16JUL2005	25AUG2015
2	Hurricane Harvey	25AUG2017	16JUL2017	25AUG2027
3	Hurricane Maria	19SEP2017	16AUG2017	19SEP2027
4	Hurricane Sandy	30OCT2012	15SEP2012	30OCT2022
5	Hurricane Irma	06SEP2017	16AUG2017	06SEP2027
6	Hurricane Andrew	23AUG1992	16JUL1992	23AUG2002
7	Hurricane Ike	12SEP2008	16AUG2008	12SEP2018

```
***********************************************************;
*  p203d03.sas Using Character Functions to Extract Words   *;
*  from a String                                           *;
***********************************************************;
*  Syntax                                                  *;
*                                                          *;
*    SCAN (string, n <, 'delimiters'>)                     *;
*    PROPCASE (string, <, 'delimiters'>)                   *;
***********************************************************;


***********************************************************;
*  Demo                                                    *;
*  1) Notice that the DATA step creates the City and       *;
*     Prefecture columns by extracting the first or        *;
*     second word from Location. Highlight the step and    *;
*     run the selected code.                               *;
*  2) Examine row 8 in the output data. Notice that the    *;
*     city name should be MIYAKE-JIMA. However, the        *;
*     hyphen is a default delimiter, so MIYAKE is          *;
*     assigned to City and JIMA is assigned to             *;
*     Prefecture.                                          *;
*  3) In both SCAN functions, add a third argument to     *;
*    specify that the only delimiter is a comma.        *;
*    Highlight the step and run the selected code.      *;
*  4) Add an additional assignment statement to create a  *;
*    column named Country that reads the last word in    *;
*    Location.                                    *;
*  5) Use the PROPCASE function in the City assignment    *;
*    statement to capitalize the first letter of each    *;
*    word and convert the remaining letters to          *;
*    lowercase. Highlight the step and run the selected  *;
*    code.                                        *;
*  6) Examine row 8 again in the output data. Because the *;
*    hyphen is a delimiter, both Miyake and Jima are     *;
*    capitalized. The proper casing for this city name   *;
*    should be Miyake-jima. Use the optional second      *;
*    argument to specify that the only delimiter should  *;
*    be a space. Highlight the step and run the selected *;
*    code.                                        *;
***********************************************************;


data weather_japan_clean;
        set pg2.weather_japan;
        Location=compbl(Location);
        City=propcase(scan(Location, 1, ','), " ");
        Prefecture=scan(Location, 2, ',');
        Country=scan(Location, -1);
run;
```

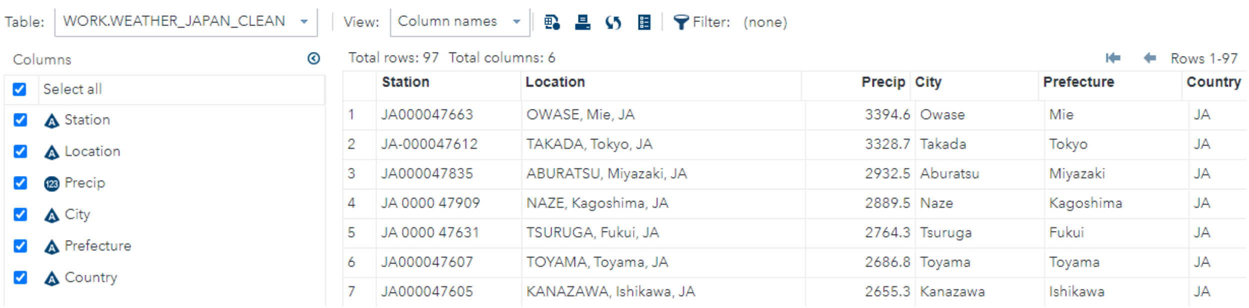


Table: WORK.WEATHER_JAPAN_CLEAN View: Column names Filter: (none)

Columns

☑ Select all
☑ Station
☑ Location
☑ Precip
☑ City
☑ Prefecture
☑ Country

Total rows: 97 Total columns: 6 Rows 1-97

	Station	Location	Precip	City	Prefecture	Country
1	JA000047663	OWASE, Mie, JA	3394.6	Owase	Mie	JA
2	JA-000047612	TAKADA, Tokyo, JA	3328.7	Takada	Tokyo	JA
3	JA000047835	ABURATSU, Miyazaki, JA	2932.5	Aburatsu	Miyazaki	JA
4	JA 0000 47909	NAZE, Kagoshima, JA	2889.5	Naze	Kagoshima	JA
5	JA 0000 47631	TSURUGA, Fukui, JA	2764.3	Tsuruga	Fukui	JA
6	JA000047607	TOYAMA, Toyama, JA	2686.8	Toyama	Toyama	JA
7	JA000047605	KANAZAWA, Ishikawa, JA	2655.3	Kanazawa	Ishikawa	JA

```
***********************************************************;
*  p203d04.sas Using the INPUT and PUT Functions to Convert        *;
*  Column Types                               *;
***********************************************************;
*  Syntax and Example                            *;
*                                            *;
*    DATA output-table;                          *;
*       SET input-table(RENAME=(current-col=new-col));   *;
*       ...                                       *;
*       column1 = INPUT (source, informat);            *;
*       column2 = PUT (source, format);                *;
*       ...                                       *;
*    RUN;                                        *;
***********************************************************;


data work.stocks2;
        set pg2.stocks2;
        Date2=input(Date,date9.);
        Volume=input(Volume,comma12.);
run;


data work.stocks2;
        set pg2.stocks2(rename=(Volume=CharVolume));
        Date2=input(Date,date9.);
        Volume=input(CharVolume,comma12.);
        drop CharVolume;
run;

data work.stocks2;
        set pg2.stocks2(rename=(Volume=CharVolume Date=CharDate));
        Volume=input(CharVolume,comma12.);
        Date=input(CharDate,date9.);
        Day=put(Date,downname3.);
        drop Char:;
run;

***********************************************************;
*  Demo                                        *;
*  1) Open the PG2.WEATHER_ATLANTA table and notice the   *;
*    following:                                *;
*    * ZipCode is a numeric column.                 *;
*    * Date and Precip are character columns. A Precip   *;
*    value of T means that a trace value was recorded,   *;
*    which means a very small amount of precipitation    *;
*    that results in no measurable accumulation.        *;
*  2) Run the first DATA step.                    *;
*  3) View the SAS log. SAS attempts to convert the       *;
*    character Precip value to a numeric value using the *;
*    w. informat. SAS is successful when the character   *;
*    value is a legitimate numeric value such as .27.    *;
*    SAS is unsuccessful when the value is equal to a    *;
*    non-numeric value such as T. A value of T is       *;
*    converted to a missing numeric value.            *;
*  4) View the output table. Notice that TotalPrecip was  *;
*    accurately created for each row. The sum statement  *;
*    ignores the missing values for the Precip values of *;
*    T.                                       *;
*  5) Add to the DATA step to create a new column named   *;
*    PrecipNum. Use PrecipNum in the assignment         *;
*    statement instead of Precip. Drop the Precip       *;
*    column.                                   *;
*  6) Run the DATA step. Notice that the SAS log no      *;
*    longer contains a note about character values being *;
*    converted to numeric values and no longer contains  *;
*    notes about invalid numeric data for Precip='T'.    *;
*  7) Add to the DATA step to create a numeric column     *;
*    Date from the character column Date. Also, format   *;
*    the numeric Date and drop the character Date.      *;
*  8) Run the DATA step. Confirm that you have a numeric  *;
*    precipitation column and a numeric date column.     *;
**********************************************************;

/* INPUT Function */
data atl_precip;
        set pg2.weather_atlanta;
        where AirportCode='ATL';
        drop AirportCode City Temp: ZipCode Precip;
        *TotalPrecip+Precip;
        If Precip ne 'T' then PrecipNum=input(Precip, 6.);
        else PrecipNum=0;
        TotalPrecip+PrecipNum;
```

```
run;

/* INPUT Function */
data atl_precip;
        set pg2.weather_atlanta(rename=(date=CharDate));
        where AirportCode='ATL';
        drop AirportCode City Temp: ZipCode Precip CharDate;
        *TotalPrecip+Precip;
        If Precip ne 'T' then PrecipNum=input(Precip, 6.);
        else PrecipNum=0;
        TotalPrecip+PrecipNum;
        Date=input(CharDate, mmddyy10.);
        Format Date date9.;
run;

/* Original INPUT Function */
data atl_precip;
        set pg2.weather_atlanta;
        where AirportCode='ATL';
        drop AirportCode City Temp: ZipCode;
        TotalPrecip+Precip;
run;

***********************************************************;
*  9) Run the second DATA step and notice that           *;
*    CityStateZip was accurately created for each row.   *;
*    The CAT functions automatically convert numeric     *;
*    values to character values and remove leading       *;
*    blanks in the converted value. SAS does not write a *;
*    note to the log when values are converted with the  *;
*    CAT functions.                              *;
* 10) Add to the DATA step to create a character column   *;
*    ZipCodeLast2 that contains the last two digits of   *;
*    the numeric column ZipCode.                    *;
* 11) View the SAS log. SAS converts the numeric ZipCode  *;
*    value to a character value.                    *;
* 12) View the output table. Notice that ZipCodeLast2 is  *;
*    not displaying the last two digits of the ZIP code. *;
*    When SAS automatically converts a numeric value to  *;
*    a character value, the BEST12. format is used, and  *;
*    the resulting character value is right-aligned. The *;
*    numeric value of 30320 becomes the character value  *;
*    of seven leading spaces followed by 30320.        *;
* 13) Modify the first argument of the SUBSTR function to *;
*    explicitly convert the numeric ZipCode value to a   *;
*    character value.                               *;
* 14) View the output table. Notice that ZipCodeLast2 now *;
*    displays the last two digits of the ZIP code.      *;
*********************************************************;

/* PUT Function */
data atl_precip;
        set pg2.weather_atlanta;
        CityStateZip=catx(' ',City,'GA',ZipCode);
        ZipCodeLast2=substr(put(Zipcode, z5.), 4, 2);
run;
```

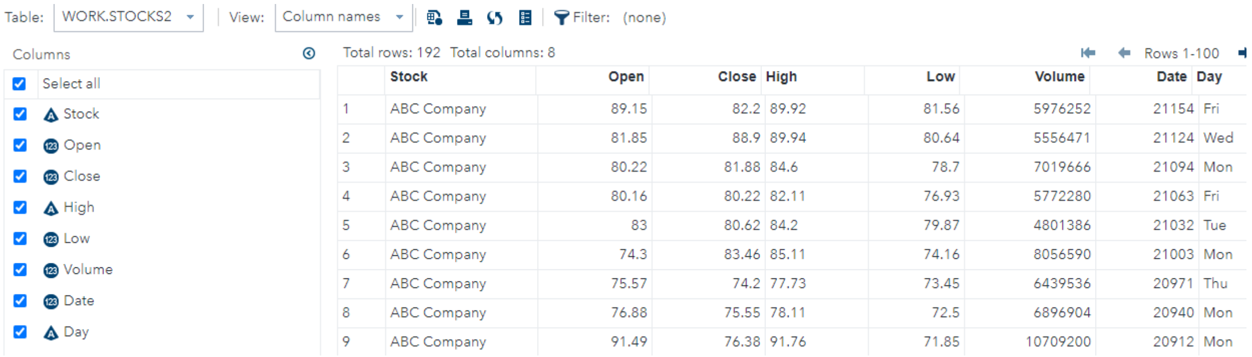


Table: WORK.STOCKS2 View: Column names Filter: (none)

Columns

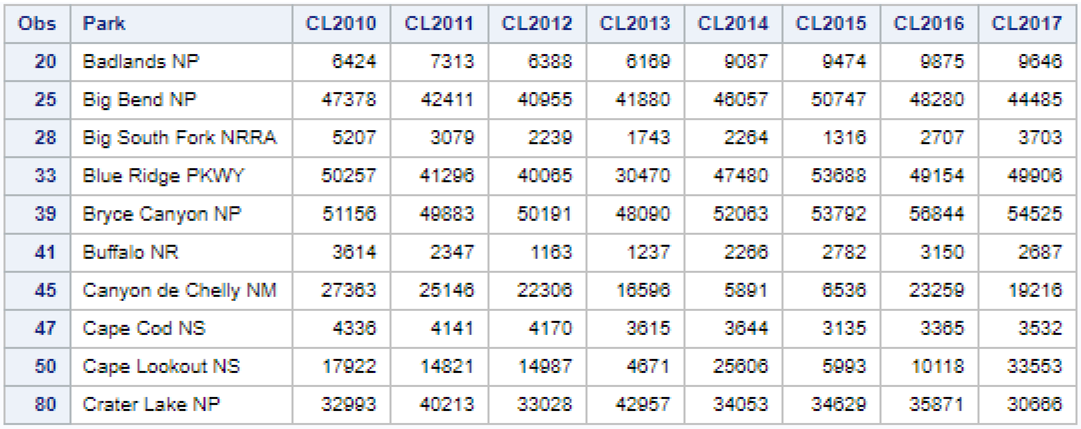
☑ Select all
☑ Stock
☑ Open
☑ Close
☑ High
☑ Low
☑ Volume
☑ Date
☑ Day

Total rows: 192 Total columns: 8 Rows 1-100

	Stock	Open	Close	High	Low	Volume	Date	Day
1	ABC Company	89.15	82.2	89.92	81.56	5976252	21154	Fri
2	ABC Company	81.85	88.9	89.94	80.64	5556471	21124	Wed
3	ABC Company	80.22	81.88	84.6	78.7	7019666	21094	Mon
4	ABC Company	80.16	80.22	82.11	76.93	5772280	21063	Fri
5	ABC Company	83	80.62	84.2	79.87	4801386	21032	Tue
6	ABC Company	74.3	83.46	85.11	74.16	8056590	21003	Mon
7	ABC Company	75.57	74.2	77.73	73.45	6439536	20971	Thu
8	ABC Company	76.88	75.55	78.11	72.5	6896904	20940	Mon
9	ABC Company	91.49	76.38	91.76	71.85	10709200	20912	Mon

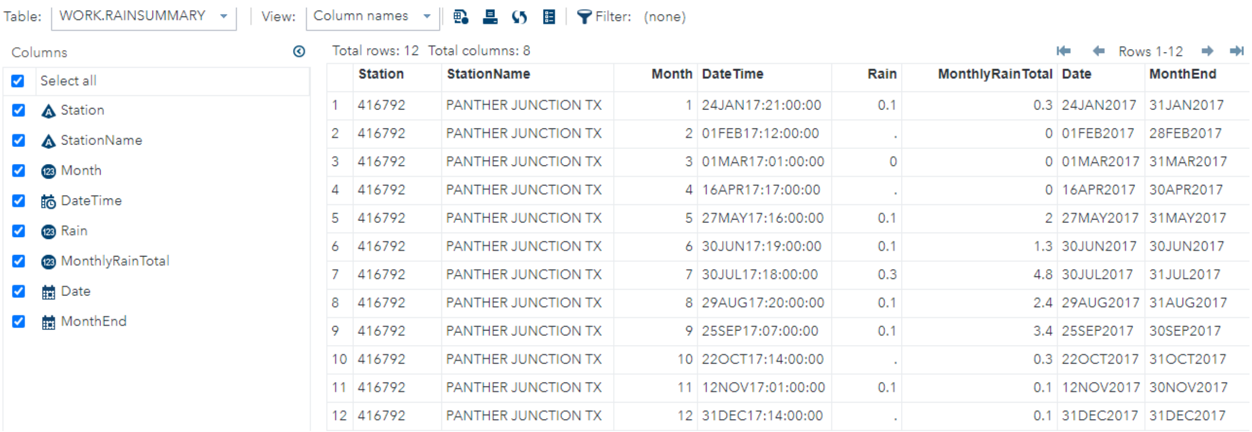
```
*******************************************************;
*  p203p01.sas LESSON 3, PRACTICE 1                    *;
*  a) Highlight the PROC PRINT step and run the selected  *;
*     code. Examine the column names and the 10 rows     *;
*     printed from the np_lodging table.               *;
*  b) Use the LARGEST function to create three new       *;
*     columns (Stay1, Stay2, and Stay3) whose values are  *;
*     the first, second, and third highest number of      *;
*     nights stayed from 2010 through 2017.             *;
*  c) Use the MEAN function to create a column named      *;
*     StayAvg that is the average number of nights stayed *;
*     for the years 2010 through 2017. Use the ROUND      *;
*     function to round values to the nearest integer.    *;
*  d) Add a subsetting IF statement to output only rows   *;
*     with StayAvg greater than zero. Highlight the DATA  *;
*     step and run the selected code.                  *;
*******************************************************;


proc print data=pg2.np_lodging(obs=10);
          where CL2010>0;
run;
```

```
data stays;
        set pg2.np_lodging;
        *Add assignment statements;
        Stay1=largest(1, of CL2010-CL2017);
        Stay2=largest(2, of CL2010-CL2017);
        Stay3=largest(3, of CL2010-CL2017);
        StayAvg=round(mean(of CL2010-CL2017),1);
        If StayAvg>0 then output;
        format Stay: comma11.;
        keep Park Stay:;
run;
```



Obs	Park	CL2010	CL2011	CL2012	CL2013	CL2014	CL2015	CL2016	CL2017
20	Badlands NP	6424	7313	6388	6169	9087	9474	9875	9646
25	Big Bend NP	47378	42411	40955	41880	46057	50747	48280	44485
28	Big South Fork NRRA	5207	3079	2239	1743	2264	1316	2707	3703
33	Blue Ridge PKWY	50257	41296	40065	30470	47480	53688	49154	49906
39	Bryce Canyon NP	51156	49883	50191	48090	52063	53792	56844	54525
41	Buffalo NR	3614	2347	1163	1237	2266	2782	3150	2687
45	Canyon de Chelly NM	27363	25146	22306	16596	5891	6536	23259	19216
47	Cape Cod NS	4336	4141	4170	3615	3644	3135	3365	3532
50	Cape Lookout NS	17922	14821	14987	4671	25606	5993	10118	33553
80	Crater Lake NP	32993	40213	33028	42957	34053	34629	35871	30666

```
**********************************************************;
*  p203p02.sas LESSON 3, PRACTICE 2                        *;
*  a) Run the program and notice that each row includes a *;
*     datetime value and rain amount. The                 *;
*     MonthlyRainTotal column represents a cumulative     *;
*     total of Rain for each value of Month.              *;
*  b) Uncomment the subsetting IF statement to continue   *;
*     processing a row only if it is the last row within  *;
*     each month. After the subsetting IF statement,      *;
*     create the following new columns:                  *;
*     1) Date - the date portion of the DateTime column   *;
*     2) MonthEnd - the last day of the month           *;
*  c) Format Date and MonthEnd as a date value and keep   *;
*     only the StationName, MonthlyRainTotal, Date, and   *;
*     MonthEnd columns.                                  *;
**********************************************************;


data rainsummary;
          set pg2.np_hourlyrain;
          by Month;
          if first.Month=1 then MonthlyRainTotal=0;
          MonthlyRainTotal+Rain;
          if last.Month=1;
          Date=datepart(DateTime);
          MonthEnd=intnx('month', Date, 0, 'end');
          format Date MonthEnd date9.;
run;
```



Total rows: 12 Total columns: 8

	Station	StationName	Month	DateTime	Rain	MonthlyRainTotal	Date	MonthEnd
1	416792	PANTHER JUNCTION TX	1	24JAN17:21:00:00	0.1	0.3	24JAN2017	31JAN2017
2	416792	PANTHER JUNCTION TX	2	01FEB17:12:00:00	.	0	01FEB2017	28FEB2017
3	416792	PANTHER JUNCTION TX	3	01MAR17:01:00:00	0	0	01MAR2017	31MAR2017
4	416792	PANTHER JUNCTION TX	4	16APR17:17:00:00	.	0	16APR2017	30APR2017
5	416792	PANTHER JUNCTION TX	5	27MAY17:16:00:00	0.1	2	27MAY2017	31MAY2017
6	416792	PANTHER JUNCTION TX	6	30JUN17:19:00:00	0.1	1.3	30JUN2017	30JUN2017
7	416792	PANTHER JUNCTION TX	7	30JUL17:18:00:00	0.3	4.8	30JUL2017	31JUL2017
8	416792	PANTHER JUNCTION TX	8	29AUG17:20:00:00	0.1	2.4	29AUG2017	31AUG2017
9	416792	PANTHER JUNCTION TX	9	25SEP17:07:00:00	0.1	3.4	25SEP2017	30SEP2017
10	416792	PANTHER JUNCTION TX	10	22OCT17:14:00:00	.	0.3	22OCT2017	31OCT2017
11	416792	PANTHER JUNCTION TX	11	12NOV17:01:00:00	0.1	0.1	12NOV2017	30NOV2017
12	416792	PANTHER JUNCTION TX	12	31DEC17:14:00:00	.	0.1	31DEC2017	31DEC2017

```
*******************************************************;
*  p203p04.sas LESSON 3, PRACTICE 4                    *;
*  a) Run the program and examine the data. Notice that   *;
*     ParkName includes a code at the end of each value   *;
*     that represents the park type. Also notice that     *;
*     some of the values for Location are in uppercase.   *;
*  b) Add a LENGTH statement to create a new            *;
*     five-character column named Type.                 *;
*  c) Add an assignment statement that uses the SCAN      *;
*     function to extract the last word from the ParkName *;
*     column and assigns the resulting value to Type.     *;
*  d) Add an assignment statement to use the UPCASE and   *;
*     COMPRESS functions to change the case of Region and *;
*     remove any blanks.                                *;
*  e) Add an assignment statement to use the PROPCASE     *;
*     function to change the case of Location.          *;
*******************************************************;

data clean_traffic;
	set pg2.np_monthlytraffic;
	drop Year;
	length Type $ 5;
	Type=scan(ParkName, -1);
	Region=compress(upcase(Region));
	Location=propcase(Location);
run;
```

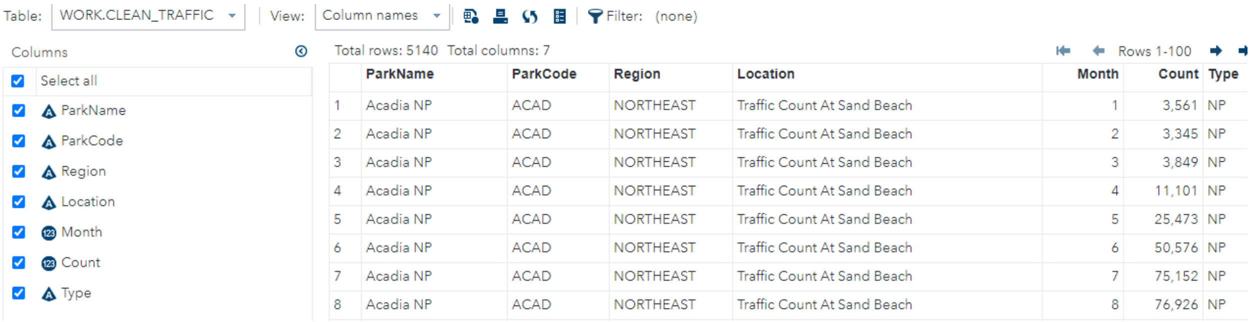


Table: WORK.CLEAN_TRAFFIC View: Column names Filter: (none)

Columns

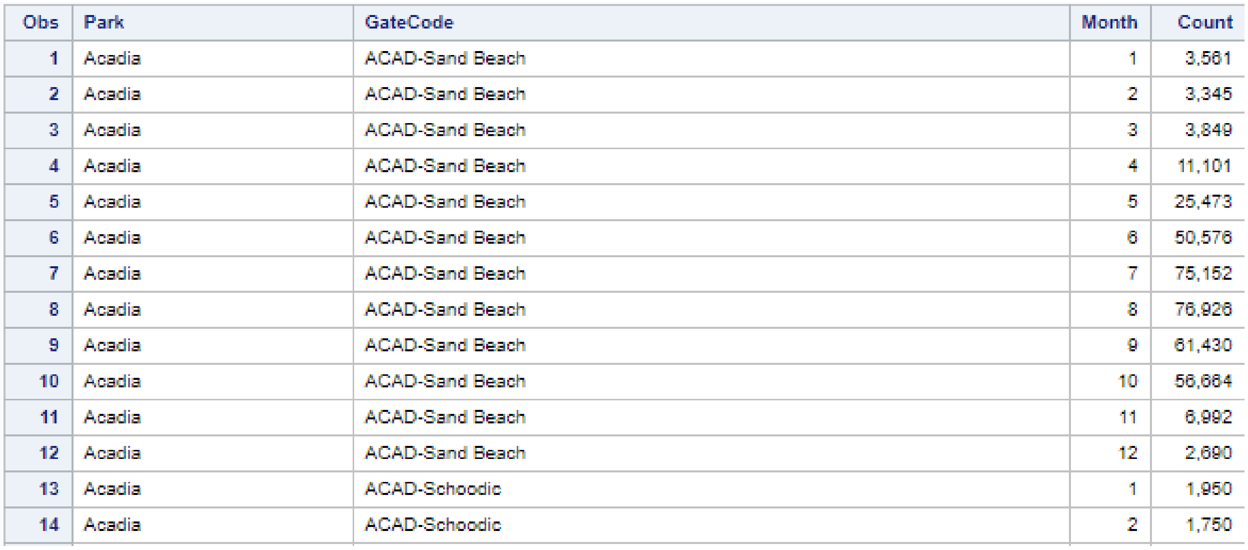
☑ Select all
☑ ParkName
☑ ParkCode
☑ Region
☑ Location
☑ Month
☑ Count
☑ Type

Total rows: 5140 Total columns: 7 Rows 1-100

	ParkName	ParkCode	Region	Location	Month	Count	Type
1	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	1	3,561	NP
2	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	2	3,345	NP
3	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	3	3,849	NP
4	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	4	11,101	NP
5	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	5	25,473	NP
6	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	6	50,576	NP
7	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	7	75,152	NP
8	Acadia NP	ACAD	NORTHEAST	Traffic Count At Sand Beach	8	76,926	NP

```
************************************************************;
*  p203p05.sas LESSON 3, PRACTICE 5                        *;
*  a) Notice that the DATA step creates a table named      *;
*     PARKS and reads only those rows where ParkName ends *;
*     with NP.                                             *;
*  b) Modify the DATA step to create or modify the         *;
*     following columns:                                   *;
*     1) Use the SUBSTR function to create a new column    *;
*        named Park that reads each ParkName value and     *;
*        excludes the NP code at the end of the string.    *;
*        Note: Use the FIND function to identify the       *;
*        position number of the NP string. That value can *;
*        be used as the third argument of the SUBSTR       *;
*        function to specify how many characters to read. *;
*     2) Convert the Location column to proper case. Use   *;
*        the COMPBL function to remove any extra blanks    *;
*        between words.                                    *;
*     3) Use the TRANWRD function to create a new column   *;
*        named Gate that reads Location and converts the   *;
*        string Traffic Count At to a blank.               *;
*     4) Create a new column names GateCode that           *;
*        concatenates ParkCode and Gate together with a    *;
*        single hyphen between the strings.                *;
```

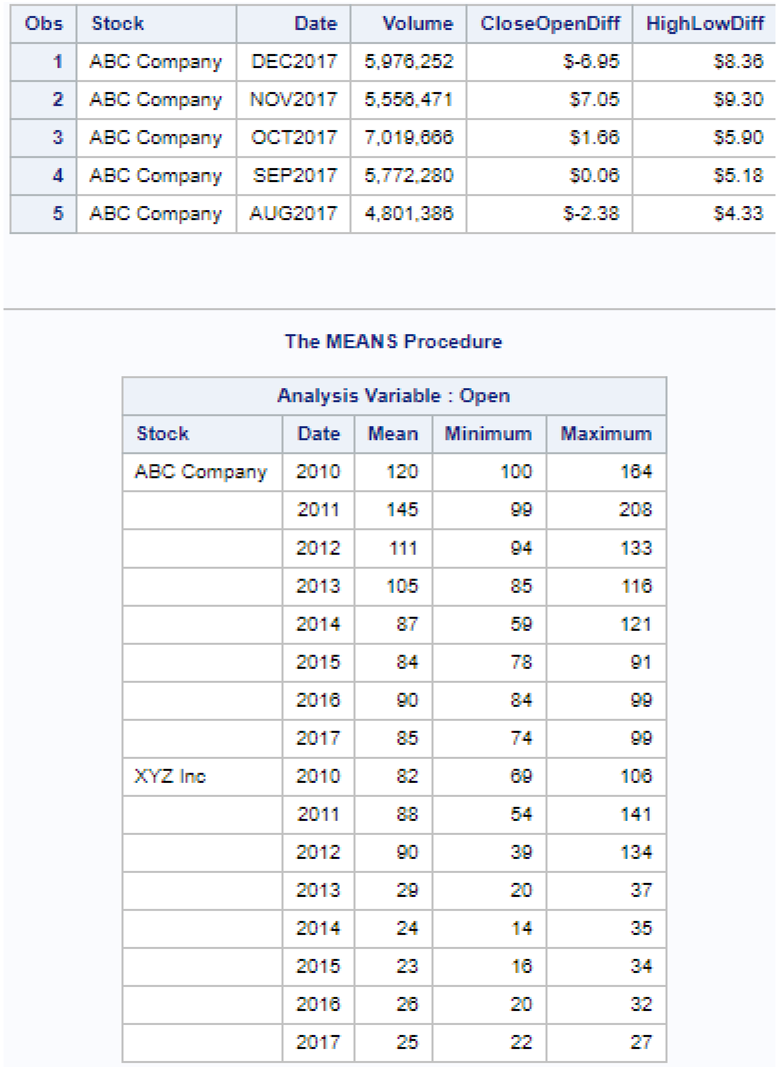
```
***************************************************************;

data parks;
	set pg2.np_monthlytraffic;
	where ParkName like '%NP';
	Park=substr(ParkName, 1, find(ParkName, 'NP')-1);
	Location=compbl(propcase(Location));
	Gate=tranwrd(Location, 'Traffic Count At ',' ');
	GateCode=cats(ParkCode,'-',Gate);
run;

proc print data=parks;
	var Park GateCode Month Count;
run;
```



Obs	Park	GateCode	Month	Count
1	Acadia	ACAD-Sand Beach	1	3,561
2	Acadia	ACAD-Sand Beach	2	3,345
3	Acadia	ACAD-Sand Beach	3	3,849
4	Acadia	ACAD-Sand Beach	4	11,101
5	Acadia	ACAD-Sand Beach	5	25,473
6	Acadia	ACAD-Sand Beach	6	50,576
7	Acadia	ACAD-Sand Beach	7	75,152
8	Acadia	ACAD-Sand Beach	8	76,926
9	Acadia	ACAD-Sand Beach	9	61,430
10	Acadia	ACAD-Sand Beach	10	56,664
11	Acadia	ACAD-Sand Beach	11	6,992
12	Acadia	ACAD-Sand Beach	12	2,690
13	Acadia	ACAD-Schoodic	1	1,950
14	Acadia	ACAD-Schoodic	2	1,750

```
***********************************************************;
*  p204a01.sas Activity 4.01                              *;
*  1) Add a FORMAT statement in the DATA step to format   *;
*     the following values:                               *;
*       Date => 3-letter month and 4-digit year (MONYY7.) *;
*       Volume => Add commas (COMMA12.)                   *;
*       CloseOpenDiff, HighLowDiff =>                     *;
*           Add dollar signs and include 2 decimal        *;
*           places (DOLLAR8.2)                            *;
*  2) Run the program and verify the formatted values in  *;
*     the PROC PRINT output.                              *;
*  3) Add a FORMAT statement in the PROC MEANS step to    *;
*     format the values of Date to show only a four-digit *;
*     year. Run the PROC MEANS step again.                *;
*  4) What is the advantage of adding a FORMAT statement  *;
*     to the DATA step versus the PROC step?              *;
***********************************************************;

data work.stocks;
   set pg2.stocks;
   CloseOpenDiff=Close-Open;
   HighLowDiff=High-Low;
   *add a FORMAT statement;
   format Date monyy7. Volume comma12. CloseOpenDiff HighLowDiff dollar8.2;
run;

proc print data=stocks (obs=5);
   var Stock Date Volume CloseOpenDiff HighLowDiff;
run;
```

```
proc means data=stocks maxdec=0 nonobs mean min max;

   class Stock Date;

   var Open;

   *add a FORMAT statement;

   format Date year4.;

run;
```



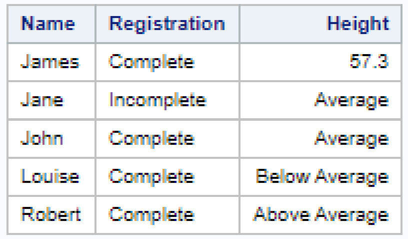
Obs	Stock	Date	Volume	CloseOpenDiff	HighLowDiff
1	ABC Company	DEC2017	5,976,252	$-6.95	$8.36
2	ABC Company	NOV2017	5,556,471	$7.05	$9.30
3	ABC Company	OCT2017	7,019,666	$1.66	$5.90
4	ABC Company	SEP2017	5,772,280	$0.06	$5.18
5	ABC Company	AUG2017	4,801,386	$-2.38	$4.33

The MEANS Procedure

Analysis Variable : Open				
Stock	Date	Mean	Minimum	Maximum
ABC Company	2010	120	100	164
	2011	145	99	208
	2012	111	94	133
	2013	105	85	116
	2014	87	59	121
	2015	84	78	91
	2016	90	84	99
	2017	85	74	99
XYZ Inc	2010	82	69	106
	2011	88	54	141
	2012	90	39	134
	2013	29	20	37
	2014	24	14	35
	2015	23	16	34
	2016	26	20	32
	2017	25	22	27

```
*******************************************************;
*  p204a02.sas Activity 4.02                          *;
*  1) In the PROC FORMAT step, modify the second VALUE    *;
*    statement to create a format named HRANGE that has  *;
*    the following criteria:                          *;
*    * A range of 50 - 57 has a formatted value of      *;
*      Below Average.                                 *;
*    * A range of 58 - 60 has a formatted value of      *;
*      Average.                                       *;
*    * A range of 61 - 70 has a formatted value of      *;
*      Above Average.                                 *;
*  2) In the PROC PRINT step, modify the FORMAT statement *;
*    to format Height with the HRANGE format.           *;
*  3) Run the program and verify the formatted values in  *;
*    the PRINT output.                                *;
*  4) Why is the Height value for the first row not       *;
*    formatted?                                       *;
*******************************************************;

proc format;
   value $regfmt 'C'='Complete'
                 'I'='Incomplete';
   *modify the following VALUE statement;
   value hrange 50-57 = 'Below Average'
                58-60 = 'Average'
                61-70 = 'Above Average' ;
run;

proc print data=pg2.class_birthdate noobs;
```

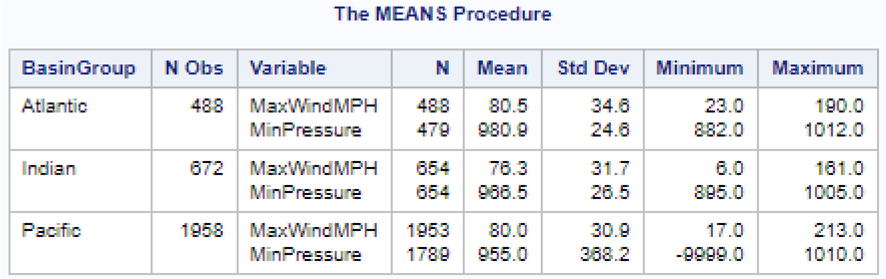
```
   where Age=12;
   var Name Registration Height;
   *add to the following FORMAT statement;
   format Registration $regfmt. Height hrange.;
run;
```



Name	Registration	Height
James	Complete	57.3
Jane	Incomplete	Average
John	Complete	Average
Louise	Complete	Below Average
Robert	Complete	Above Average

```
***************************************************;
*  p204a03.sas Activity 4.03                        *;
*  1) Review the PROC FORMAT step that creates the     *;
*     $REGION format that assigns basin codes into     *;
*     groups. Highlight the step and run the selected    *;
*     code.                                     *;
*  2) Notice the DATA step includes IF-THEN/ELSE        *;
*     statements to create a new column named BasinGroup. *;
*  3) Delete the IF-THEN/ELSE statements and replace it   *;
*     with an assignment statement to create the         *;
*     BasinGroup column. Use the PUT function with Basin  *;
*     as the first argument and $REGION. as the second    *;
*     argument.                                 *;
*  4) Highlight the DATA and PROC MEANS steps and run the *;
*     selected code. How many BasinGroup values are in    *;
*     the summary report?                        *;
***************************************************;

proc format;
```

```
  value $region 'NA'='Atlantic'
         'WP','EP','SP'='Pacific'
         'NI','SI'='Indian'
         ' '='Missing'
         other='Unknown';
run;

data storm_summary;
  set pg2.storm_summary;
  Basin=upcase(Basin);
  *Delete the IF-THEN/ELSE statements and replace them with an assignment statement;
/*   if Basin='NA' then BasinGroup='Atlantic';
  else if Basin in ('WP','EP','SP') then BasinGroup='Pacific';
  else if Basin in ('NI','SI') then BasinGroup='Indian';
  else if Basin=' ' then BasinGroup='Missing';
  else BasinGroup='Unknown';
*/
        BasinGroup=put(Basin, $region.);
run;

proc means data=storm_summary maxdec=1;
        class BasinGroup;
        var MaxWindMPH MinPressure;
run;
```

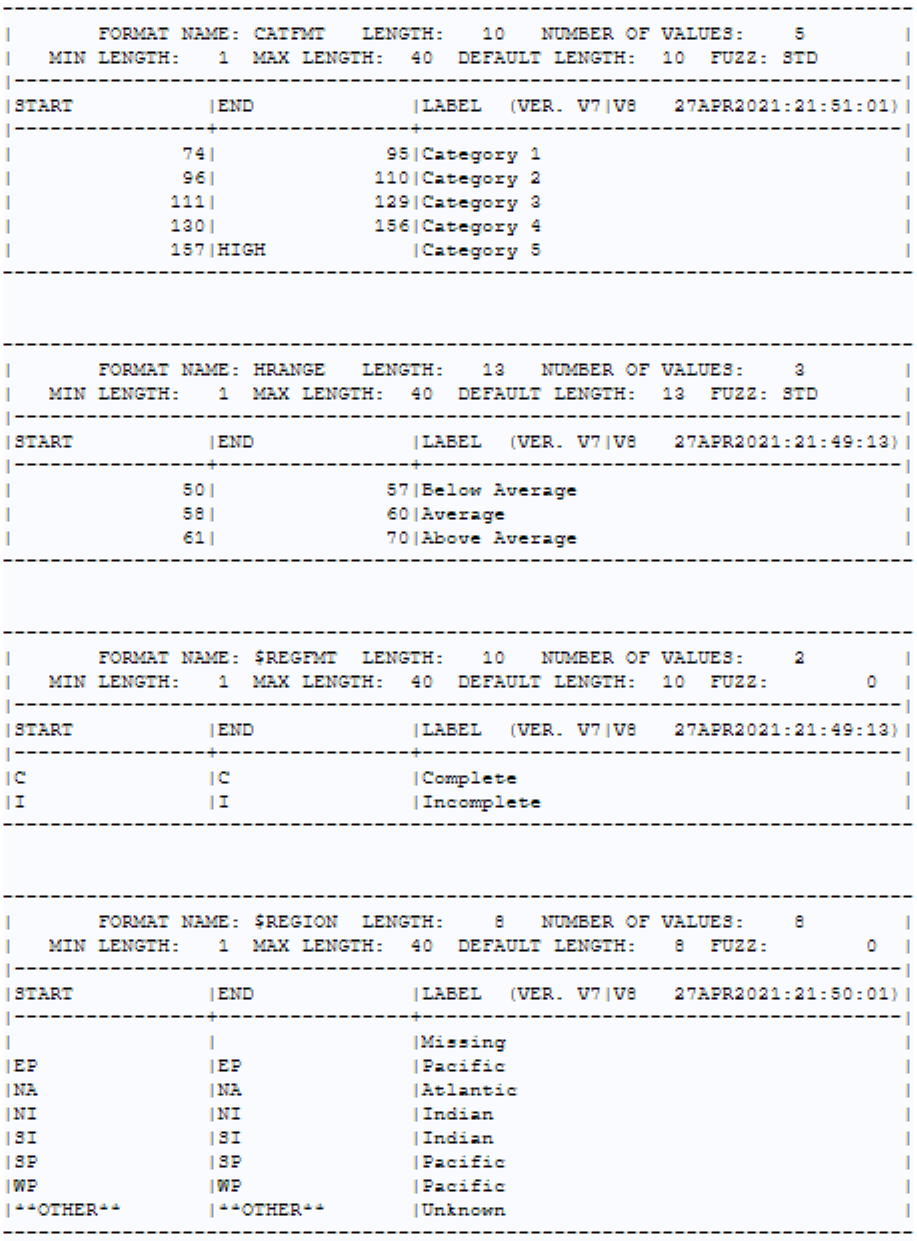


The MEANS Procedure

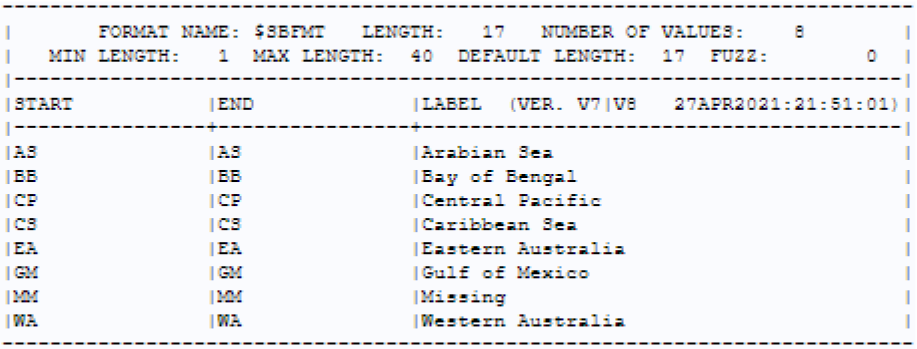
BasinGroup	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
Atlantic	488	MaxWindMPH MinPressure	488 479	80.5 980.9	34.6 24.6	23.0 882.0	190.0 1012.0
Indian	672	MaxWindMPH MinPressure	654 654	76.3 966.5	31.7 26.5	6.0 895.0	161.0 1005.0
Pacific	1958	MaxWindMPH MinPressure	1953 1789	80.0 955.0	30.9 368.2	17.0 -9999.0	213.0 1010.0

```
*********************************************************;
*  p204a04.sas Activity 4.04                              *;
*  1) Run the program to create the $SBFMT and CATFMT     *;
*     formats. View the log to confirm both were output.  *;
*  2) Uncomment the PROC FORMAT step at the end of the    *;
*     program. Highlight the step and run the selected    *;
*     code. A report for all formats in the WORK library  *;
*     is generated.                                       *;
*  3) Add the following statement in the last PROC FORMAT *;
*     step to limit the report to selected formats. Run   *;
*     the step.                                           *;
*         select $sbfmt catfmt;                           *;
*  4) What are the default lengths for the $SBFMT and     *;
*     CATFMT formats?                                     *;
*********************************************************;

/*Create the $SBFMT format for subbasin codes*/
data sbdata;
   retain FmtName '$sbfmt';
   set pg2.storm_subbasincodes(rename=(Sub_Basin=Start
                               SubBasin_Name=Label));
   keep Start Label FmtName;
run;

proc format cntlin=sbdata;
run;

/*Create the CATFMT format for storm categories*/
data catdata;
```

```
   retain FmtName "catfmt";
   set pg2.storm_categories(rename=(Low=Start
                              High=End
                              Category=Label));
   keep FmtName Start End Label;
run;

proc format cntlin=catdata;
run;

proc format fmtlib library=work;
run;
```

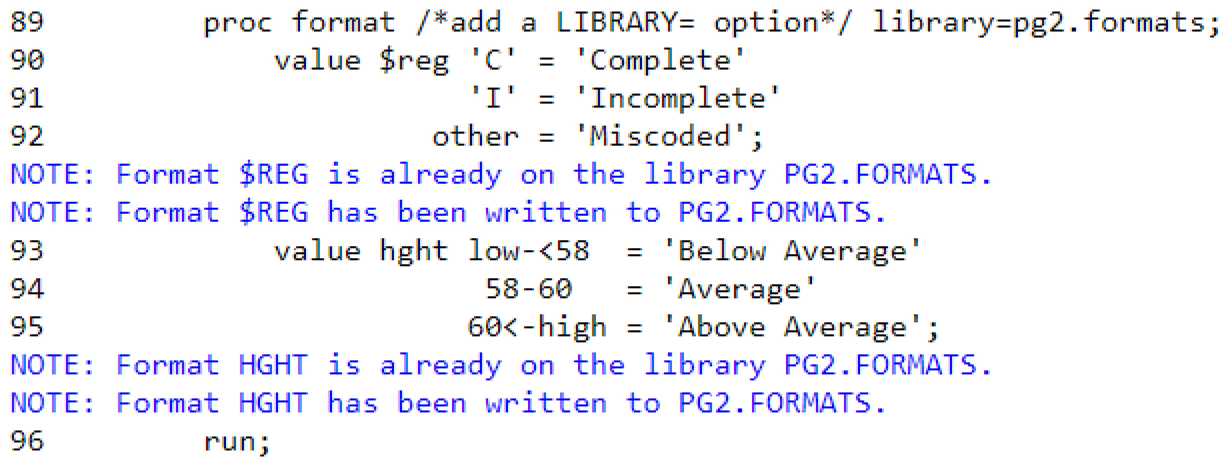


```
----------------------------------------------------------------------------
|       FORMAT NAME: CATFMT   LENGTH:   10   NUMBER OF VALUES:    5         |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  10  FUZZ: STD        |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   27APR2021:21:51:01)|
|----------------+----------------+----------------------------------------|
|              74|              95|Category 1                              |
|              96|             110|Category 2                              |
|             111|             129|Category 3                              |
|             130|             156|Category 4                              |
|             157|HIGH            |Category 5                              |
----------------------------------------------------------------------------


----------------------------------------------------------------------------
|       FORMAT NAME: HRANGE   LENGTH:   13   NUMBER OF VALUES:    3         |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  13  FUZZ: STD        |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   27APR2021:21:49:13)|
|----------------+----------------+----------------------------------------|
|              50|              57|Below Average                           |
|              58|              60|Average                                 |
|              61|              70|Above Average                           |
----------------------------------------------------------------------------


----------------------------------------------------------------------------
|       FORMAT NAME: $REGFMT  LENGTH:   10   NUMBER OF VALUES:    2         |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  10  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   27APR2021:21:49:13)|
|----------------+----------------+----------------------------------------|
|C               |C               |Complete                                |
|I               |I               |Incomplete                              |
----------------------------------------------------------------------------


----------------------------------------------------------------------------
|       FORMAT NAME: $REGION  LENGTH:    8   NUMBER OF VALUES:    8         |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:   8  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   27APR2021:21:50:01)|
|----------------+----------------+----------------------------------------|
|                |                |Missing                                 |
|EP              |EP              |Pacific                                 |
|NA              |NA              |Atlantic                                |
|NI              |NI              |Indian                                  |
|SI              |SI              |Indian                                  |
|SP              |SP              |Pacific                                 |
|WP              |WP              |Pacific                                 |
|**OTHER**       |**OTHER**       |Unknown                                 |
----------------------------------------------------------------------------
```



```
----------------------------------------------------------------------------
|       FORMAT NAME: $SBFMT   LENGTH:   17   NUMBER OF VALUES:    8         |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  17  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. V7|V8   27APR2021:21:51:01)|
|----------------+----------------+----------------------------------------|
|AS              |AS              |Arabian Sea                             |
|BB              |BB              |Bay of Bengal                           |
|CP              |CP              |Central Pacific                         |
|CS              |CS              |Caribbean Sea                           |
|EA              |EA              |Eastern Australia                       |
|GM              |GM              |Gulf of Mexico                          |
|MM              |MM              |Missing                                 |
|WA              |WA              |Western Australia                       |
----------------------------------------------------------------------------
```

```
***********************************************************;
*  p204a05.sas Activity 4.05                              *;
*  1) In the PROC FORMAT statement, add the LIBRARY=      *;
*     option to save the formats to the PG2.FORMATS       *;
*     catalog.                                            *;
*  2) Run the PROC FORMAT step and verify in the log that *;
*     the two formats were created in a permanent         *;
*     location.                                           *;
*  3) Before the PROC PRINT step, add an OPTIONS          *;
*     statement so that SAS can find the two permanent    *;
*     formats.                                            *;
*        options fmtsearch=(pg2.formats);                 *;
*  4) Run the OPTIONS statement and the PROC PRINT step.  *;
*     Are the Registration and Height values formatted?   *;
***********************************************************;

proc format /*add a LIBRARY= option*/ library=pg2.formats;
   value $reg 'C' = 'Complete'
              'I' = 'Incomplete'
            other = 'Miscoded';
   value hght low-<58  = 'Below Average'
               58-60   = 'Average'
              60<-high = 'Above Average';
run;
```

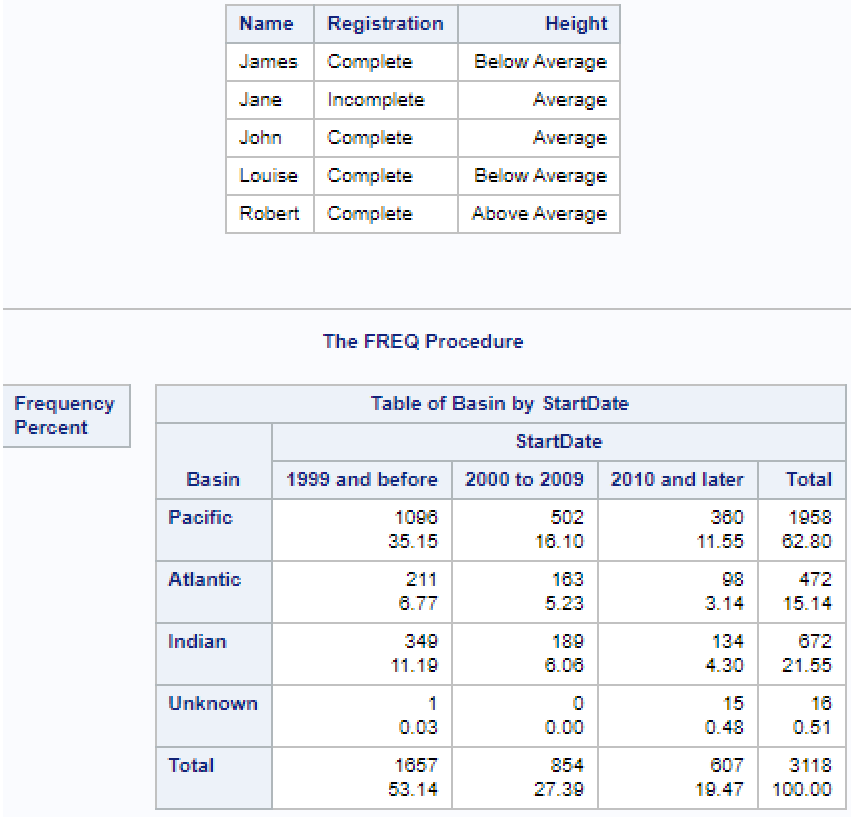


```
89         proc format /*add a LIBRARY= option*/ library=pg2.formats;
90             value $reg 'C' = 'Complete'
91                        'I' = 'Incomplete'
92                      other = 'Miscoded';
NOTE: Format $REG is already on the library PG2.FORMATS.
NOTE: Format $REG has been written to PG2.FORMATS.
93             value hght low-<58  = 'Below Average'
94                         58-60    = 'Average'
95                       60<-high = 'Above Average';
NOTE: Format HGHT is already on the library PG2.FORMATS.
NOTE: Format HGHT has been written to PG2.FORMATS.
96         run;
```

```
***************************************************;
*  p204d01.sas Creating and Using Custom Formats          *;
***************************************************;
*  Syntax                          *;
*                                *;
*    PROC FORMAT;                        *;
*     VALUE format-name                    *;
*          value-or-range-1='formatted-value'   *;
*          value-or-range-2='formatted-value'   *;
*          ... ;                      *;
*    RUN;                           *;
***************************************************;


proc format;
   value $regfmt 'C' = 'Complete'
              'I' = 'Incomplete'
             other = 'Miscoded';
   value hrange low-<58  = 'Below Average'
              58-60    = 'Average'
              60<-high = 'Above Average';
run;
```

```
proc print data=pg2.class_birthdate noobs;
   where Age=12;
   var Name Registration Height;
   format Registration $regfmt. Height hrange.;
run;

********************************************************;
*  Demo                                          *;
*  1) Notice the syntax for creating the STDATE format in *;
*     the PROC FORMAT step.                            *;
*  2) Add a VALUE statement to the PROC FORMAT step to    *;
*     create the $REGION format with the following        *;
*     labels:                                       *;
*        NA => Atlantic                             *;
*        WP, EP, SP => Pacific                       *;
*        NI, SI => Indian                           *;
*        blank => Missing                           *;
*        other => Unknown                            *;
*  3) Highlight the PROC FORMAT step and run the selected *;
*     code. Verify in the SAS log that the formats have   *;
*     been output.                                  *;
*  4) Add a FORMAT statement in the PROC FREQ step to     *;
*     format Basin with the $REGION format and StartDate  *;
*     with the STDATE format. Highlight PROC FREQ step    *;
*     and run the selected code.                      *;
********************************************************;

proc format;
```

```
  value stdate low - '31DEC1999'd = '1999 and before'
          '01JAN2000'd - '31DEC2009'd = '2000 to 2009'
          '01JAN2010'd - high = '2010 and later'
          . = 'Not Supplied';
  *Add a VALUE statement;
  value $region 'NA'='Atlantic'
                          'WP','EP','SP'='Pacific'
                          'NI','SI'='Indian'
                          ' '='Missing'
                          other='Unknown';
run;

proc freq data=pg2.storm_summary;
  tables Basin*StartDate / norow nocol;
  *Add a FORMAT statement;
  format StartDate stdate. Basin $region.;
run;
```



Name	Registration	Height
James	Complete	Below Average
Jane	Incomplete	Average
John	Complete	Average
Louise	Complete	Below Average
Robert	Complete	Above Average

The FREQ Procedure

Frequency
Percent

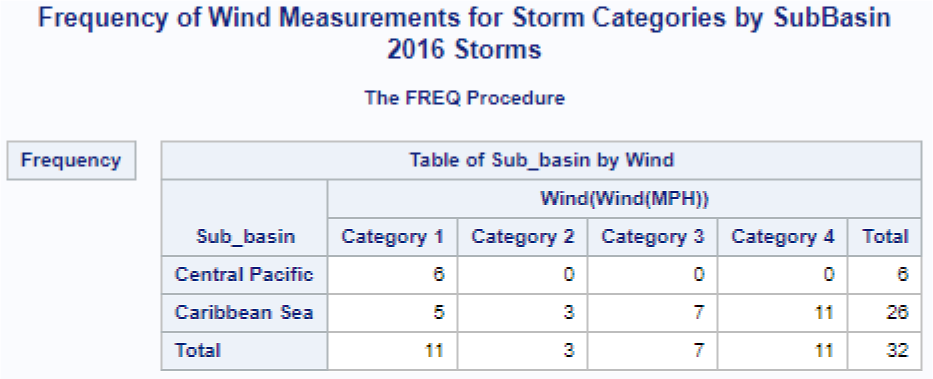
Table of Basin by StartDate

	StartDate			
Basin	1999 and before	2000 to 2009	2010 and later	Total
Pacific	1096 35.15	502 16.10	360 11.55	1958 62.80
Atlantic	211 6.77	163 5.23	98 3.14	472 15.14
Indian	349 11.19	189 6.06	134 4.30	672 21.55
Unknown	1 0.03	0 0.00	15 0.48	16 0.51
Total	1657 53.14	854 27.39	607 19.47	3118 100.00

```
**************************************************;
*  p204d02.sas Creating Custom Formats from Tables        *;
**************************************************;
*  Syntax                              *;
*                                      *;
*   CNTLIN table must include:               *;
*      FmtName: name of format               *;
*      Start: values to format             *;
*      Label: labels to apply              *;
*                                      *;
*   PROC FORMAT CNTLIN=input-table FMTLIB;       *;
*     SELECT format-names;                   *;
*   RUN;                               *;
**************************************************;
```

```
*********************************************************************;
*  Demo                                   *;
*    1) Examine the DATA step that creates the SBDATA table from    *;
*       the PG2.STORM_SUBBASINCODES table and the PROC FORMAT step  *;
*       that imports the SBDATA table. Highlight the demo program   *;
*       and run the selected code. Verify that the new table        *;
*       contains three required columns to build a format. View the *;
*       log and confirm the $SBFMT format was created.              *;
*    2) Open the PG2.STORM_CATEGORIES table. This table defines a   *;
*       range of maximum wind speeds (Low and High) and assigns a   *;
*       storm Category.                               *;
*    3) Modify the second DATA and PROC FORMAT steps to create a    *;
*       table named CATDATA that will include the following         *;
*       columns. Highlight the DATA and PROC FORMAT steps and run   *;
*       the selected code. View the log and confirm the CATFMT      *;
*       format was created.                           *;
*          Column in PG2.STORM_CATEGORIES => Column in CATDATA      *;
*             <none> => FmtName (assign CATFMT for each row       *;
*             Low => Start                           *;
*             High => End                            *;
*             Category => Label                        *;
*    4) Add a FORMAT statement in the PROC FREQ step to format      *;
*       Sub_basin with the $SBFMT. format and Wind with the CATFMT. *;
*       format. Highlight the TITLE statements and PROC FREQ step   *;
*       and run the selected code.                    *;
*********************************************************************;


/*Create the $SBFMT format for subbasin codes*/
data sbdata;
```

```
   retain FmtName '$sbfmt';
   set pg2.storm_subbasincodes(rename=(Sub_Basin=Start
                                  SubBasin_Name=Label));
   keep Start Label FmtName;
run;

proc format cntlin=sbdata;
run;

/*Complete the steps to create the CATFMT format from the storm_categories table*/
data catdata;
   retain Fmtname 'catfmt';
   set pg2.storm_categories(rename=(Low=Start High=End Category=Label)) ;
   keep FmtName Start End Label;
run;

proc format cntlin=catdata;
run;

title "Frequency of Wind Measurements for Storm Categories by SubBasin";
title2 "2016 Storms";
proc freq data=pg2.storm_detail;
   /*include only Category 1-5 2016 storms with known subbasin*/
         where Wind>=74 and Season=2016 and Sub_basin not in('MM', 'NA');
         tables Sub_basin*Wind / nocol norow nopercent;
         *Add a FORMAT statement;
         format Sub_basin $sbfmt. Wind catfmt.;
run;
title;
```



Frequency of Wind Measurements for Storm Categories by SubBasin
2016 Storms

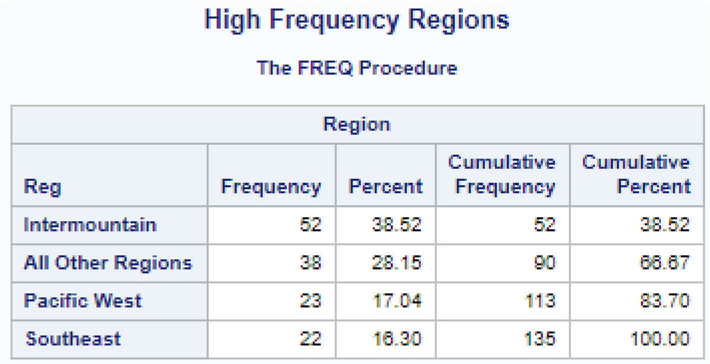
The FREQ Procedure

Frequency

Table of Sub_basin by Wind					
	Wind(Wind(MPH))				
Sub_basin	Category 1	Category 2	Category 3	Category 4	Total
Central Pacific	6	0	0	0	6
Caribbean Sea	5	3	7	11	26
Total	11	3	7	11	32

```
***********************************************************;
*  p204p01.sas LESSON 4, PRACTICE 1                        *;
*  a) Highlight the PROC FREQ step and run the selected    *;
*     code. Review the output. Notice that regional codes *;
*     are used, not descriptive values.                    *;
*  b) Add a VALUE statement to the PROC FORMAT step to     *;
*     create a format named $HIGHREG that defines the      *;
*     descriptive values shown below.                      *;
*     IM => Intermountain                                  *;
*     PW => Pacific West                                   *;
*     SE => Southeast                                      *;
*     other codes => All Other Regions                     *;
*  c) Add a FORMAT statement to the PROC FREQ step so      *;
*     that the $HIGHREG format is applied to the Reg       *;
*     column.                                              *;
*  d) Run the program and review the output. Verify that   *;
*     the descriptive values for the Reg column are        *;
*     displayed.                                           *;
***********************************************************;


proc format;
	value $highreg 'IM'='Intermountain'
```

```
                                    'PW'='Pacific West'

                                    'SE'='Southeast'

                                    other='All Other Regions';

run;


title 'High Frequency Regions';

proc freq data=pg2.np_summary order=freq;

  tables Reg;

  label Reg='Region';

        format Reg $highreg.;

run;

title;
```

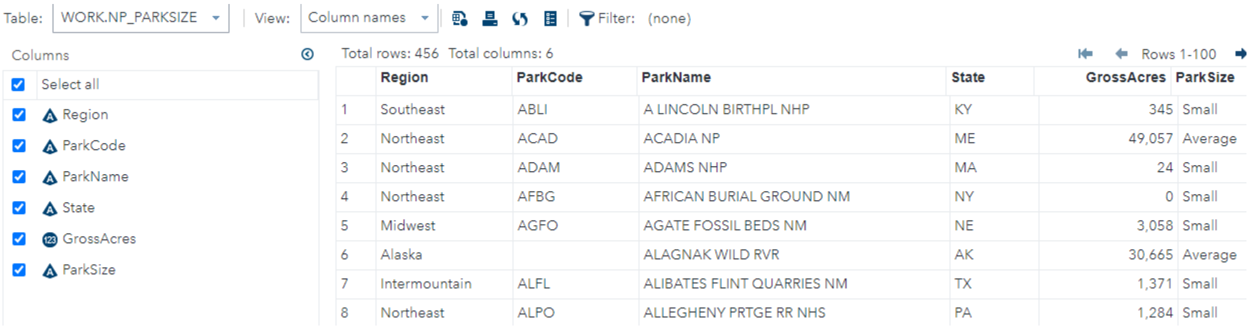


High Frequency Regions

The FREQ Procedure

Region				
Reg	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Intermountain	52	38.52	52	38.52
All Other Regions	38	28.15	90	66.67
Pacific West	23	17.04	113	83.70
Southeast	22	16.30	135	100.00

```
***************************************************************;

*  p204p02.sas LESSON 4, PRACTICE 2                   *;

*  a) Before the DATA step, add a PROC FORMAT step to    *;

*     create a format named PSIZE that categorizes parks  *;

*     based on the gross acres. Use the ranges and values *;

*     as identified below.                      *;

*     Less than 10,000 acres => Small                 *;

*     10,000 through less than 500,000 acres => Average   *;

*     500,000 and more acres => Large                  *;

*  b) In the DATA step, add an assignment statement to    *;
*     create a new column named ParkSize. Use the PUT     *;
*     function to create the new column based on the      *;
*     formatted values of GrossAcres.                     *;
*  c) Run the program and view the output table. Verify   *;
*     the values of the ParkSize column.                  *;
****************************************************;


/* Add a PROC FORMAT Step */
proc format;
          value psize low-<10000='Small'
                              10000-<500000='Average'
                              500000-high='Large';
run;


data np_parksize;
   set pg2.np_acres;
          ParkSize=put(GrossAcres, psize.);
   format GrossAcres comma16.;
run;
```

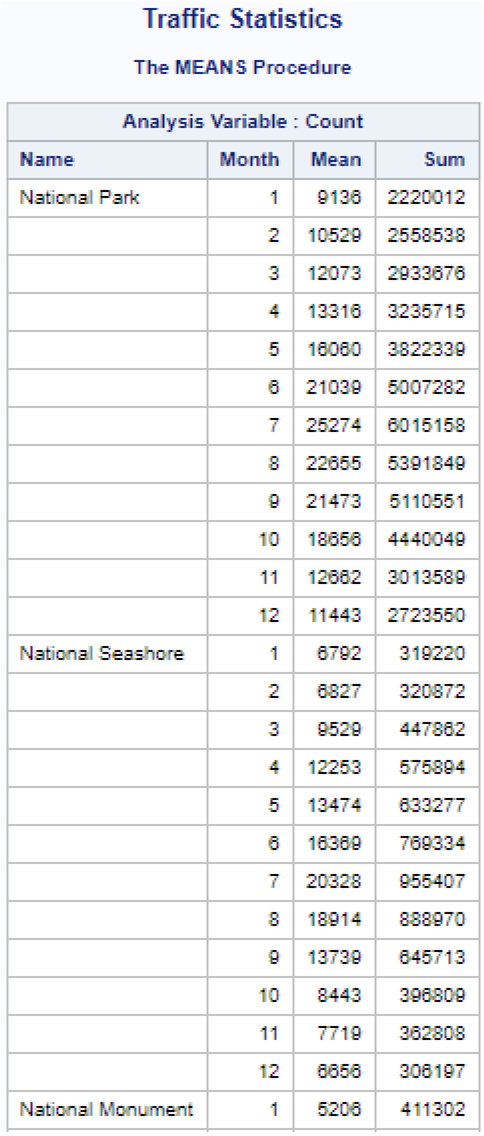


Total rows: 456 Total columns: 6

	Region	ParkCode	ParkName	State	GrossAcres	ParkSize
1	Southeast	ABLI	A LINCOLN BIRTHPL NHP	KY	345	Small
2	Northeast	ACAD	ACADIA NP	ME	49,057	Average
3	Northeast	ADAM	ADAMS NHP	MA	24	Small
4	Northeast	AFBG	AFRICAN BURIAL GROUND NM	NY	0	Small
5	Midwest	AGFO	AGATE FOSSIL BEDS NM	NE	3,058	Small
6	Alaska		ALAGNAK WILD RVR	AK	30,665	Average
7	Intermountain	ALFL	ALIBATES FLINT QUARRIES NM	TX	1,371	Small
8	Northeast	ALPO	ALLEGHENY PRTGE RR NHS	PA	1,284	Small

```
*********************************************************;
*  p204p04.sas LESSON 4, PRACTICE 4                     *;
*  a) Highlight the PROC MEANS step and run the selected  *;
*    code. Review the output. Notice that the traffic    *;
*    statistics are listed by a four-letter park code.   *;
*  b) Open the PG2.NP_CODELOOKUP table. Notice that       *;
*    ParkCode contains the four-letter park code and     *;
*    Type contains the type of park.                     *;
*  c) Modify the DATA step.                              *;
*    1) Add a RENAME= data set option to the SET         *;
*      statement to rename the ParkCode column to Start *;
*      and the Type column to Label.                     *;
*    2) Add a RETAIN statement before the SET statement  *;
*      to create the FmtName column with a value of     *;
*      $TypeFmt (without a period at the end).          *;
*  d) In the PROC FORMAT statement, add a CNTLIN= option  *;
*    to build a format from the type_lookup table.       *;
*  e) In the PROC MEANS step, add a FORMAT statement so   *;
*    that the $TypeFmt format is applied to the ParkCode *;
*    column.                                             *;
*  f) Run the program and review the results. Verify that *;
*    the data is grouped by park types.                  *;
*********************************************************;


data type_lookup;
        retain FmtName '$TypeFmt';
   set pg2.np_codeLookup(rename=(ParkCode=Start Type=Label));
   keep Start Label FmtName;
run;
```

```
proc format cntlin=type_lookup;
run;

title 'Traffic Statistics';
proc means data=pg2.np_monthlyTraffic maxdec=0 mean sum nonobs;
   var Count;
   class ParkCode Month;
   label ParkCode='Name';
   format ParkCode $Typefmt.;
run;
title;
```



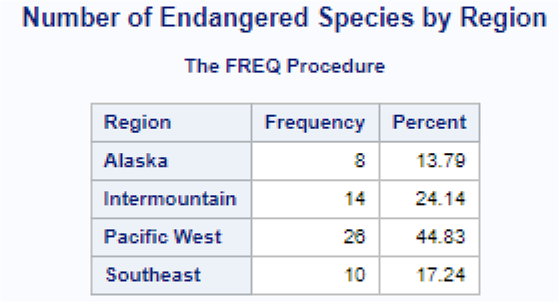
Traffic Statistics

The MEANS Procedure

Analysis Variable : Count			
Name	Month	Mean	Sum
National Park	1	9136	2220012
	2	10529	2558538
	3	12073	2933676
	4	13316	3235715
	5	16060	3822339
	6	21039	5007282
	7	25274	6015158
	8	22655	5391849
	9	21473	5110551
	10	18656	4440049
	11	12662	3013589
	12	11443	2723550
National Seashore	1	6792	319220
	2	6827	320872
	3	9529	447862
	4	12253	575894
	5	13474	633277
	6	16369	769334
	7	20328	955407
	8	18914	888970
	9	13739	645713
	10	8443	396809
	11	7719	362808
	12	6656	306197
National Monument	1	5206	411302

```
***********************************************************;
*  LESSON 4, PRACTICE 5                                   *;
*  a) Modify the first DATA step to create the NP_LOOKUP  *;
*     table that will be used to build a custom format.   *;
*     1) Add a RETAIN statement to create the FmtName     *;
*        column with a value of $RegLbl.                  *;
*     2) Add a RENAME= data set option to the SET         *;
*        statement to rename the ParkCode column to       *;
*        Start.                                           *;
*    3) Add conditional statements to create the Label   *;
*       column. The Label column is equal to the Region  *;
*       column unless the region is missing. In that     *;
*       case, the Label column is equal to a value of    *;
*       Unknown.                                 *;
*    4) Add a KEEP statement to include the Start,       *;
*       Label, and FmtName columns.                  *;
*  b) Highlight the first DATA step and run the selected  *;
*    code. Verify the output table.                   *;
*  c) Modify the PROC FORMAT step to read in the         *;
*    NP_LOOKUP table.                              *;
*  d) In the second DATA step, create a new column named  *;
*    Region. Use the PUT function to create the new      *;
*    column based on using the $RegLbl format on the     *;
*    ParkCode column. Run the program and confirm the    *;
*    results in the PROC FREQ output.                 *;
**********************************************************;


data np_lookup;
	retain FmtName '$RegLbl';
   set pg2.np_codeLookup(rename=(ParkCode=Start));
   if Region ne ' ' then Label=Region;
   else Label='Unknown';
	keep Start Label FmtName;
run;


proc format cntlin=np_lookup;
run;
```

```
data np_endanger;
   set pg2.np_species;
   where Conservation_Status='Endangered';
   Region=put(ParkCode, $RegLbl.);
run;


title 'Number of Endangered Species by Region';
proc freq data=np_endanger;
   tables Region / nocum;
run;
title;
```



Number of Endangered Species by Region

The FREQ Procedure

Region	Frequency	Percent
Alaska	8	13.79
Intermountain	14	24.14
Pacific West	26	44.83
Southeast	10	17.24