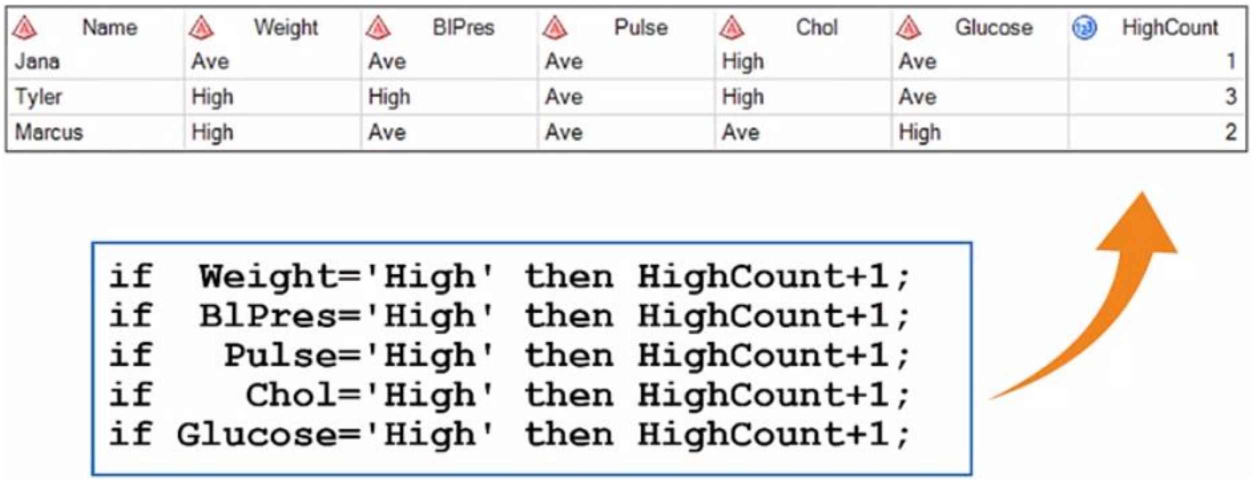
SAS Advanced Programming

SAP3 SAS Advanced Programming Techniques

SAP303 SAS Arrays: Defining and Referencing One-Dimensional Arrays, Doing More with One-Dimensional Arrays, Defining and Referencing Two-Dimensional Arrays

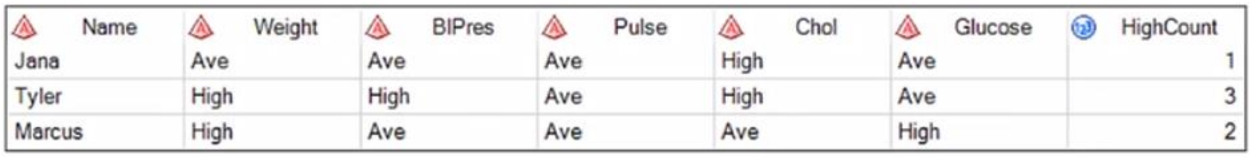
Libname.sas

```
%let path=~/EPG3M6;

%let pathout=&path/output;

libname pg3 "&path/data" filelockwait=20;

* FILELOCKWAIT=20 specifies SAS will wait up to 20 seconds
   for a locked file to become available. Use this option
   to avoid a lock error when using the FCMP procedure. */
```

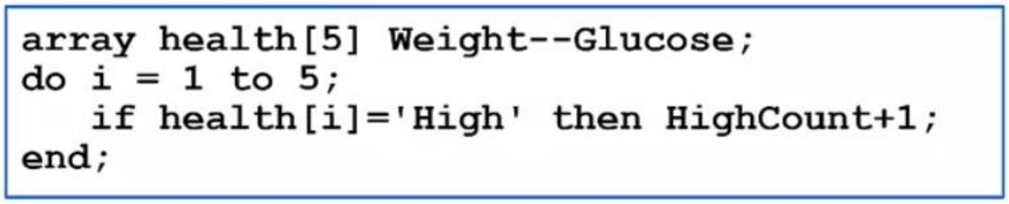


Name	Weight	BlPres	Pulse	Chol	Glucose	HighCount
Jana	Ave	Ave	Ave	High	Ave	1
Tyler	High	High	Ave	High	Ave	3
Marcus	High	Ave	Ave	Ave	High	2

```
if  Weight='High' then HighCount+1;
if  BlPres='High' then HighCount+1;
if   Pulse='High' then HighCount+1;
if    Chol='High' then HighCount+1;
if Glucose='High' then HighCount+1;
```



Name	Weight	BlPres	Pulse	Chol	Glucose	HighCount
Jana	Ave	Ave	Ave	High	Ave	1
Tyler	High	High	Ave	High	Ave	3
Marcus	High	Ave	Ave	Ave	High	2



```
array health[5] Weight--Glucose;
do i = 1 to 5;
   if health[i]='High' then HighCount+1;
end;
```

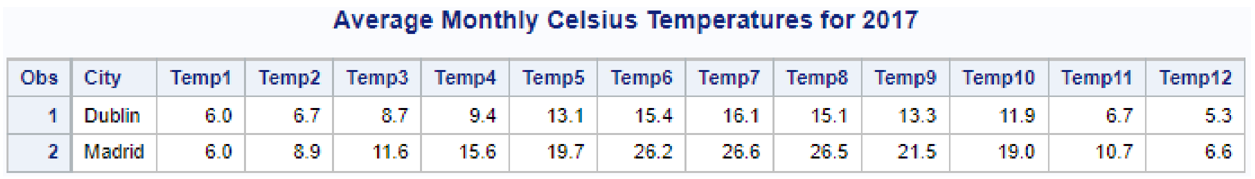
```
**********************************************************;
*  Activity 3.01                            *;
*  1) Replace the number signs (#) to reference the      *;
*     appropriate number of 2017 temperature columns.    *;
*  2) Modify the temperature conversion assignment       *;
*     statement by replacing the ??? with the name of the *;
*     column being incremented.                      *;
*  3) Run the program and confirm that you are now seeing *;
*     Celsius temperatures. What is the lowest average    *;
*     Celsius temperature for each City in 2017?         *;
**********************************************************;


 data work.DublinMadrid2017(drop=Month);
    set pg3.weather_dublinmadrid_monthly2017
      (keep=City Temp1-Temp12);
    array Temperature[12] Temp1-Temp12;
    do Month=1 to 12;
      Temperature[Month]=(Temperature[Month]-32)*5/9;
    end;
    format Temp1-Temp12 6.1;
run;
```

title 'Average Monthly Celsius Temperatures for 2017';

proc print data=work.DublinMadrid2017;

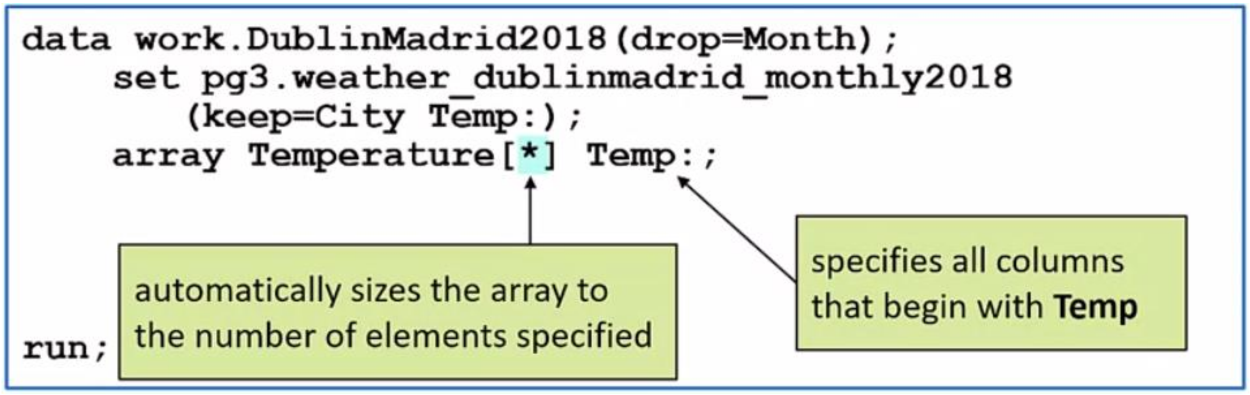
run;

title;



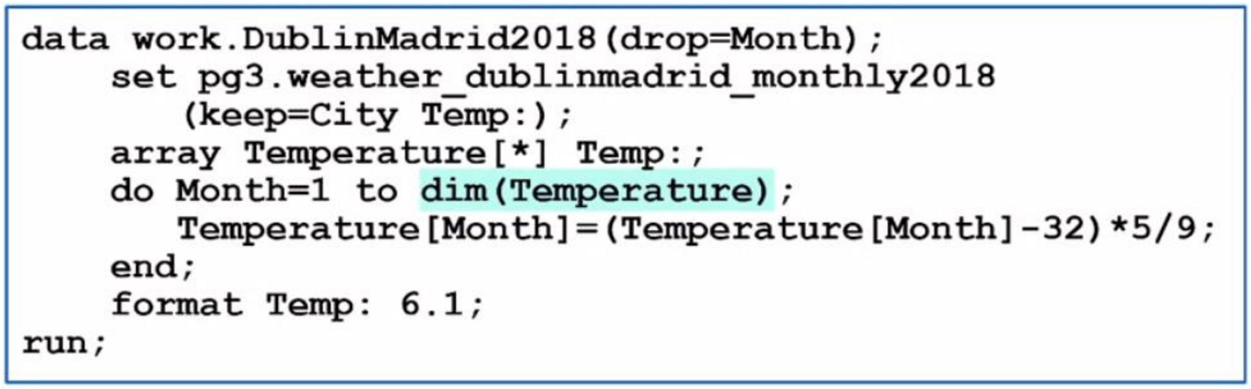
Average Monthly Celsius Temperatures for 2017

Obs	City	Temp1	Temp2	Temp3	Temp4	Temp5	Temp6	Temp7	Temp8	Temp9	Temp10	Temp11	Temp12
1	Dublin	6.0	6.7	8.7	9.4	13.1	15.4	16.1	15.1	13.3	11.9	6.7	5.3
2	Madrid	6.0	8.9	11.6	15.6	19.7	26.2	26.6	26.5	21.5	19.0	10.7	6.6

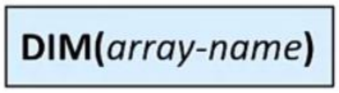


```
data work.DublinMadrid2018(drop=Month);
    set pg3.weather_dublinmadrid_monthly2018
        (keep=City Temp:);
    array Temperature[*] Temp:;
run;
```

automatically sizes the array to the number of elements specified
specifies all columns that begin with **Temp**



```
data work.DublinMadrid2018(drop=Month);
    set pg3.weather_dublinmadrid_monthly2018
        (keep=City Temp:);
    array Temperature[*] Temp:;
    do Month=1 to dim(Temperature);
        Temperature[Month]=(Temperature[Month]-32)*5/9;
    end;
    format Temp: 6.1;
run;
```



DIM(*array-name*)

***;

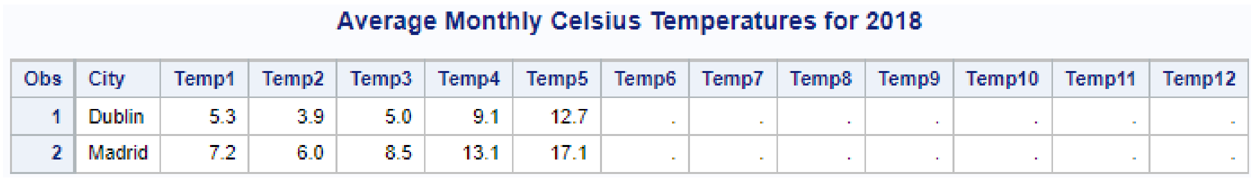
* Activity 3.02 *;

* 1) Modify the ARRAY statement to use an asterisk in *;

* place of the number of elements and to reference *;

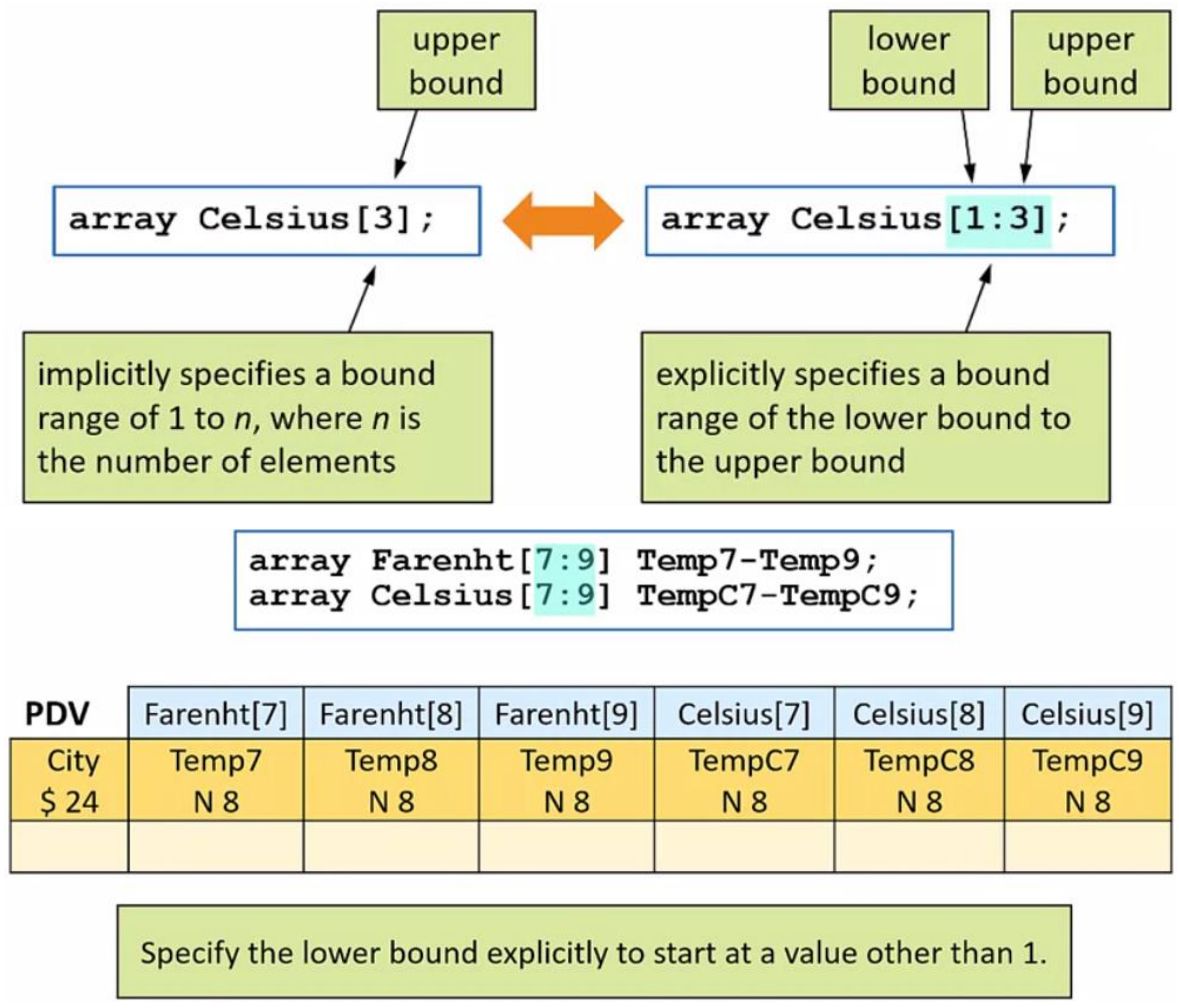
* all 2018 temperature columns that start with Temp. *;

```
*  2) In the DO statement, replace the value of 12 with   *;
*     the DIM function referencing the Temperature array. *;
*  3) Run the program. Based on the results, how many     *;
*     temperature columns are in the array for the 2018   *;
*     data?                                  *;
**********************************************************;

 data work.DublinMadrid2018(drop=Month);
    set pg3.weather_dublinmadrid_monthly2018
      (keep=City Temp:);
    array Temperature[12] Temp1-Temp12;
    do Month=1 to dim(Temperature);
      Temperature[Month]=(Temperature[Month]-32)*5/9;
    end;
    format Temp: 6.1;
run;

title 'Average Monthly Celsius Temperatures for 2018';
proc print data=work.DublinMadrid2018;
run;
title;
```



Average Monthly Celsius Temperatures for 2018

Obs	City	Temp1	Temp2	Temp3	Temp4	Temp5	Temp6	Temp7	Temp8	Temp9	Temp10	Temp11	Temp12
1	Dublin	5.3	3.9	5.0	9.1	12.7	.	.	.	.	.	.	.
2	Madrid	7.2	6.0	8.5	13.1	17.1	.	.	.	.	.	.	.


upper bound
lower bound
upper bound
implicitly specifies a bound range of 1 to *n*, where *n* is the number of elements
explicitly specifies a bound range of the lower bound to the upper bound
Specify the lower bound explicitly to start at a value other than 1.

```
array Celsius[3];
```

```
array Celsius[1:3];
```

```
array Farenht[7:9] Temp7-Temp9;
array Celsius[7:9] TempC7-TempC9;
```

PDV	Farenht[7]	Farenht[8]	Farenht[9]	Celsius[7]	Celsius[8]	Celsius[9]
City $ 24	Temp7 N 8	Temp8 N 8	Temp9 N 8	TempC7 N 8	TempC8 N 8	TempC9 N 8

```
************************************************************;
*   Processing One-Dimensional Arrays: Part 1            *;
************************************************************;


data HighSumamry1;
   set pg3.health_stats;
   HighCount=0;
   if  Weight='High' then HighCount+1;
   if  BlPres='High' then HighCount+1;
   if   Pulse='High' then HighCount+1;
   if    Chol='High' then HighCount+1;
   if Glucose='High' then HighCount+1;
run;

data HighSumamry2(drop=i);
   set pg3.health_stats;
   HighCount=0;
   array health[5] Weight BlPres Pulse Chol Glucose;
   *array health[5] Weight--Glucose;
   do i = 1 to 5;
      if health[i]='High' then HighCount+1;
   end;
run;

data work.DublinMadrid2017(drop=Month);
   set pg3.weather_dublinmadrid_Monthly2017
      (keep=City Temp1-Temp12);
   array Temperature[12] Temp1-Temp12;
   do Month=1 to 12;
      Temperature[Month]=(Temperature[Month]-32)*5/9;
   end;
   format Temp1-Temp12 6.1;
run;

data work.DublinMadrid2018(drop=Month);
   set pg3.weather_dublinmadrid_Monthly2018
      (keep=City Temp:);
   array Temperature[*] Temp:;
   do Month=1 to dim(Temperature);
      Temperature[Month]=(Temperature[Month]-32)*5/9;
   end;
   format Temp: 6.1;
run;

***********************************************************;
*  Demo                                          *;
*  1) In the first DATA step, notice the two ARRAY statements. *;
*    The array Farenht references existing columns            *;
*    of Fahrenheit temperatures read from the input table.    *;
*    The array Celsius references new numeric columns that    *;
*    are being created.                                *;
*  2) Add the following DO loop to the first DATA step after   *;
*    the ARRAY statements. The assignment statement           *;
*    calculates the Celsius temperatures.                 *;
*      do Month=1 to 3;                               *;
*        Celsius[Month]=(Farenht[Month]-32)*5/9;           *;
*      end;                                        *;
*  3) Highlight and run the DATA step. Verify that your output *;
*    table tempQ1 contains the three existing Fahrenheit      *;
*    temperatures and the three new Celsius temperatures for  *;
*    months 1 through 3.                              *;
*  4) Modify the first DATA step to create the table tempQ3    *;
*    that contains the Celsius temperatures for months 7      *;
*    through 9.                                    *;
*      data work.tempQ3(drop=Month);                      *;
*         set pg3.weather_dublinmadrid_Monthly2017          *;
*            (keep=City Temp7-Temp9);                   *;
*         array Farenht[7:9] Temp7-Temp9;                 *;
*         array Celsius[7:9] TempC7-TempC9;               *;
*         do Month=7 to 9;                              *;
*            Celsius[Month]=(Farenht[Month]-32)*5/9;        *;
*         end;                                      *;
*         format TempC7-TempC9 6.1;                       *;
*      run;                                         *;
*  5) Highlight and run the DATA step. Verify that your output *;
*    table tempQ3 contains the three existing Fahrenheit      *;
*    temperatures and the three new Celsius temperatures for  *;
*    months 7 through 9.                              *;
*  6) Open the pg3.weather_dublinmadrid_monthly2017 table.     *;
*    Notice that the table contains four quarterly           *;
*    precipitation columns in inches (PrecipQ1-PrecipQ4) in   *;
*    addition to the Fahrenheit temperature columns.         *;
*  7) In the second DATA step, notice the two array           *;
*    statements. The array P references existing columns of   *;
*    quarterly precipitation read from the input table. The   *;
*    array Pct references new numeric columns that are being  *;
*    created.                                         *;
*  8) After the first ARRAY statement, add an assignment       *;
*    statement to calculate the total yearly precipitation by *;
*    summing the four quarterly precipitation columns.       *;
*      PrecipTotal=sum(of PrecipQ1-PrecipQ4);              *;
*    An alternative for specifying the columns in the SUM     *;
*    function is to reference all elements of the P array by  *;
*    using an asterisk in an array reference.                *;
*      PrecipTotal=sum(of P[*]);                          *;
*  9) In the DO loop, add an assignment statement to calculate *;
*    the quarterly percent of precipitation based on the      *;
*    quarterly precipitation divided by the total yearly      *;
*    precipitation.                          *;
*     Pct[i]=P[i]/PrecipTotal;                    *;
* 10) Highlight and run the DATA step. Verify that your output *;
*    table precip contains the four existing quarterly       *;
*    precipitation columns and the four new percentages of    *;
*    quarterly precipitation along with the total yearly      *;
*    precipitation.                          *;
**********************************************************;
```

```
*First DATA Step;
data work.tempQ1(drop=Month);
   set pg3.weather_dublinmadrid_monthly2017
     (keep=City Temp1-Temp3);
   array Farenht[3] Temp1-Temp3;
   array Celsius[3] TempC1-TempC3;
   do Month=1 to 3;
     Celsius[Month]=(Farenht[Month]-32)*5/9;
   end;
   format TempC1-TempC3 6.1;
run;
```

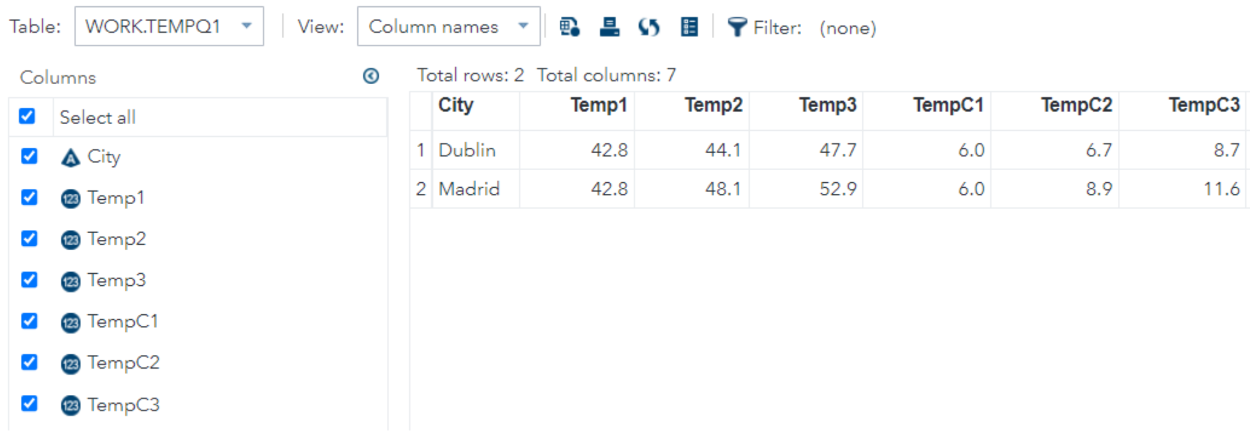


Table: WORK.TEMPQ1 View: Column names Filter: (none)

Columns: Select all, City, Temp1, Temp2, Temp3, TempC1, TempC2, TempC3

Total rows: 2 Total columns: 7

	City	Temp1	Temp2	Temp3	TempC1	TempC2	TempC3
1	Dublin	42.8	44.1	47.7	6.0	6.7	8.7
2	Madrid	42.8	48.1	52.9	6.0	8.9	11.6

```
data work.tempQ3(drop=Month);
```

```
  set pg3.weather_dublinmadrid_monthly2017
    (keep=City Temp7-Temp9);
  array Farenht[7:9] Temp7-Temp9;
  array Celsius[7:9] TempC7-TempC9;
  do Month=7 to 9;
    Celsius[Month]=(Farenht[Month]-32)*5/9;
  end;
  format TempC7-TempC9 6.1;
run;
```

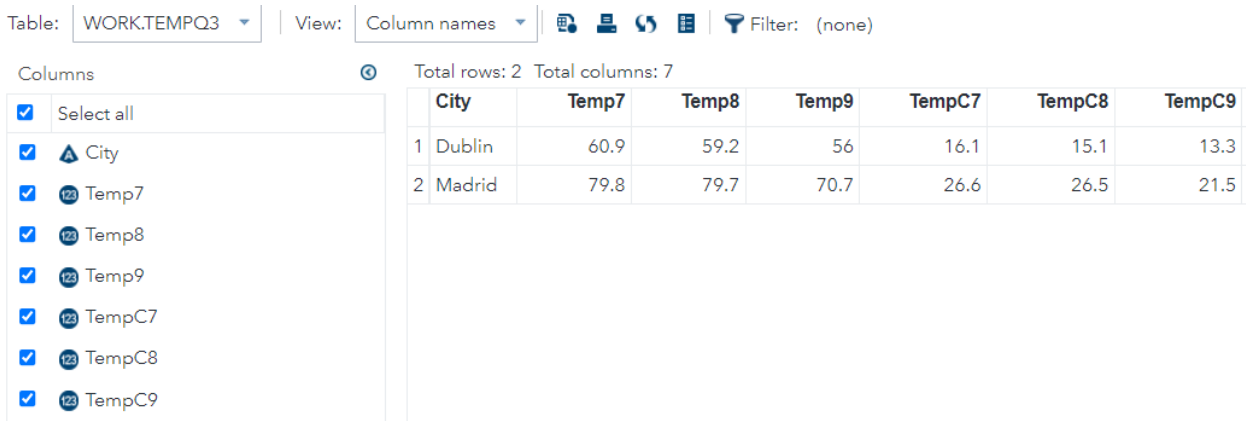


Table: WORK.TEMPQ3 View: Column names Filter: (none)

Columns: Select all, City, Temp7, Temp8, Temp9, TempC7, TempC8, TempC9

Total rows: 2 Total columns: 7

	City	Temp7	Temp8	Temp9	TempC7	TempC8	TempC9
1	Dublin	60.9	59.2	56	16.1	15.1	13.3
2	Madrid	79.8	79.7	70.7	26.6	26.5	21.5

```
*Second DATA Step;
data work.precip(drop=i);
  set pg3.weather_dublinmadrid_monthly2017
    (keep=City PrecipQ1-PrecipQ4);
  array P[4] PrecipQ1-PrecipQ4;
        PrecipTotal=sum(of PrecipQ1-PrecipQ4);
        *PrecipTotal=sum(of P[*]);
  array Pct[4] PrecipPctQ1-PrecipPctQ4;
  do i=1 to 4;
   Pct[i]=P[i]/PrecipTotal;
  end;
  format PrecipPctQ1-PrecipPctQ4 percent8.1;
```

run;

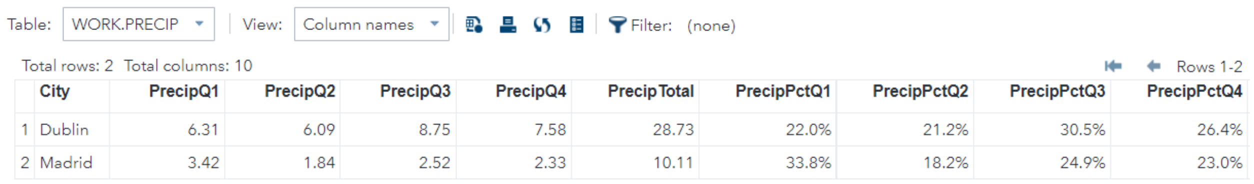
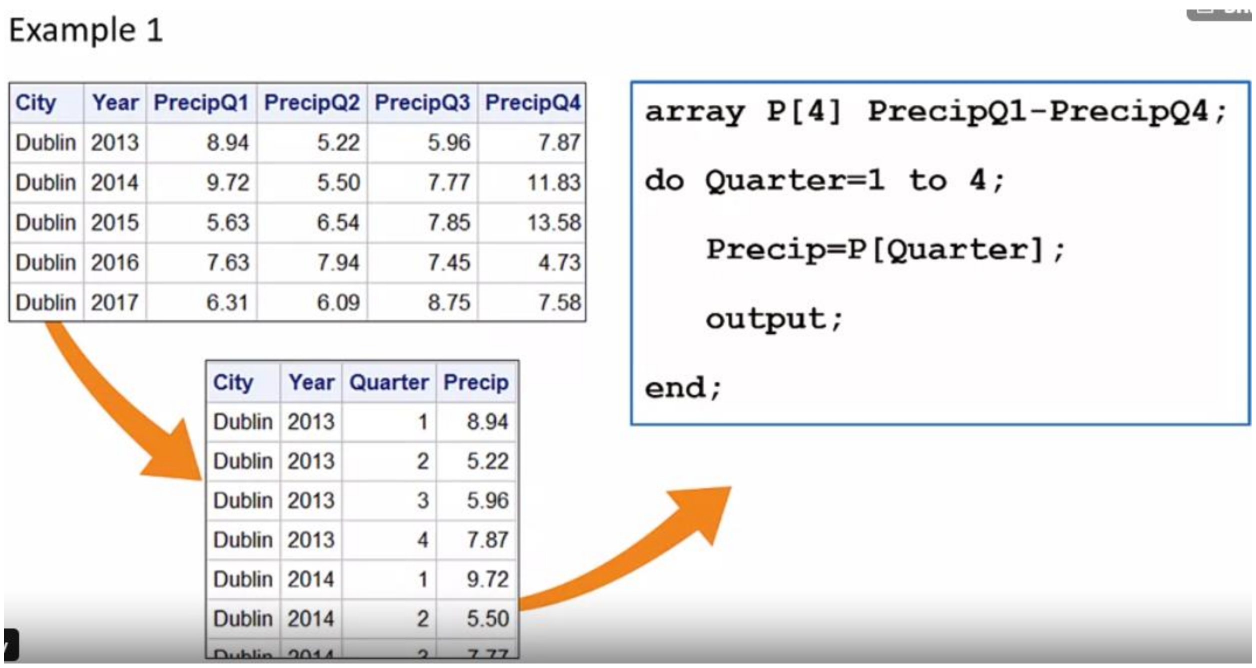


Table: WORK.PRECIP View: Column names Filter: (none)

Total rows: 2 Total columns: 10 Rows 1-2

	City	PrecipQ1	PrecipQ2	PrecipQ3	PrecipQ4	PrecipTotal	PrecipPctQ1	PrecipPctQ2	PrecipPctQ3	PrecipPctQ4
1	Dublin	6.31	6.09	8.75	7.58	28.73	22.0%	21.2%	30.5%	26.4%
2	Madrid	3.42	1.84	2.52	2.33	10.11	33.8%	18.2%	24.9%	23.0%



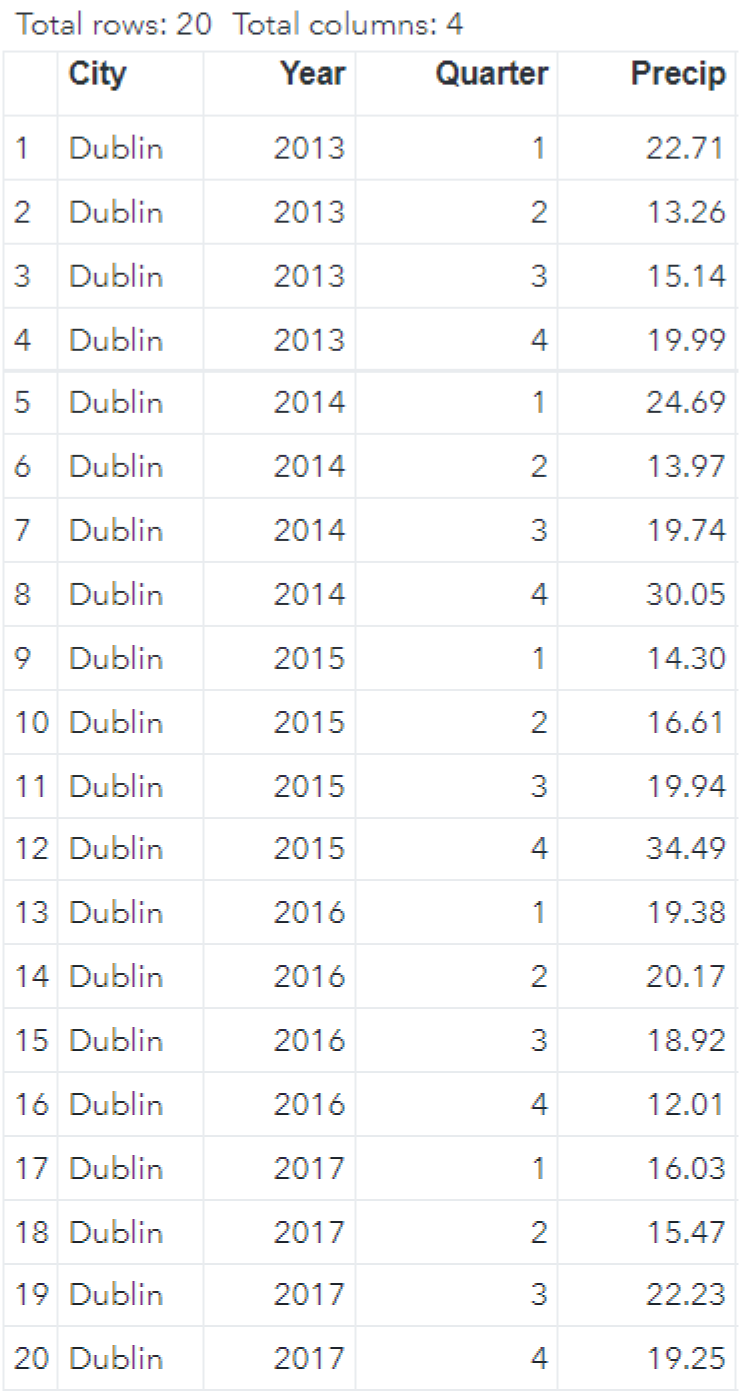
Example 1

City	Year	PrecipQ1	PrecipQ2	PrecipQ3	PrecipQ4
Dublin	2013	8.94	5.22	5.96	7.87
Dublin	2014	9.72	5.50	7.77	11.83
Dublin	2015	5.63	6.54	7.85	13.58
Dublin	2016	7.63	7.94	7.45	4.73
Dublin	2017	6.31	6.09	8.75	7.58

City	Year	Quarter	Precip
Dublin	2013	1	8.94
Dublin	2013	2	5.22
Dublin	2013	3	5.96
Dublin	2013	4	7.87
Dublin	2014	1	9.72
Dublin	2014	2	5.50

```
array P[4] PrecipQ1-PrecipQ4;
do Quarter=1 to 4;
   Precip=P[Quarter];
   output;
end;
```

```
*******************************************************;
*  Activity 3.03                           *;
*  1) Run the DATA step, which does not include ARRAY    *;
*     syntax, and verify that the new table contains 20   *;
*     rows of rotated data.                        *;
*  2) Modify the DATA step to use ARRAY syntax to rotate  *;
*     the data. Delete 12 assignment statements.         *;
*     Add 5 statements.                           *;
*      array P[4] PrecipQ1-PrecipQ4;                *;
*      do Quarter=1 to 4;                         *;
*         Precip=P[Quarter]*2.54;                  *;
*         output;                              *;
*      end;                                   *;
*  3) Run the DATA step, which now includes ARRAY syntax, *;
*    and verify that the new table contains 20 rows of   *;
*    rotated data.                                       *;
*  4) Run the PROC SGPLOT step to create the desired bar  *;
*    chart.                                              *;
*  5) What is the highest average quarterly precipitation *;
*    in centimeters for Dublin?                          *;
**********************************************************;

data work.DublinPrecipRotate;
   set pg3.weather_dublinmadrid_monthly5yr
      (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   Quarter=1; Precip=PrecipQ1*2.54; output;
   Quarter=2; Precip=PrecipQ2*2.54; output;
   Quarter=3; Precip=PrecipQ3*2.54; output;
   Quarter=4; Precip=PrecipQ4*2.54; output;
   format Precip 6.2;
   drop PrecipQ1-PrecipQ4;
run;
```

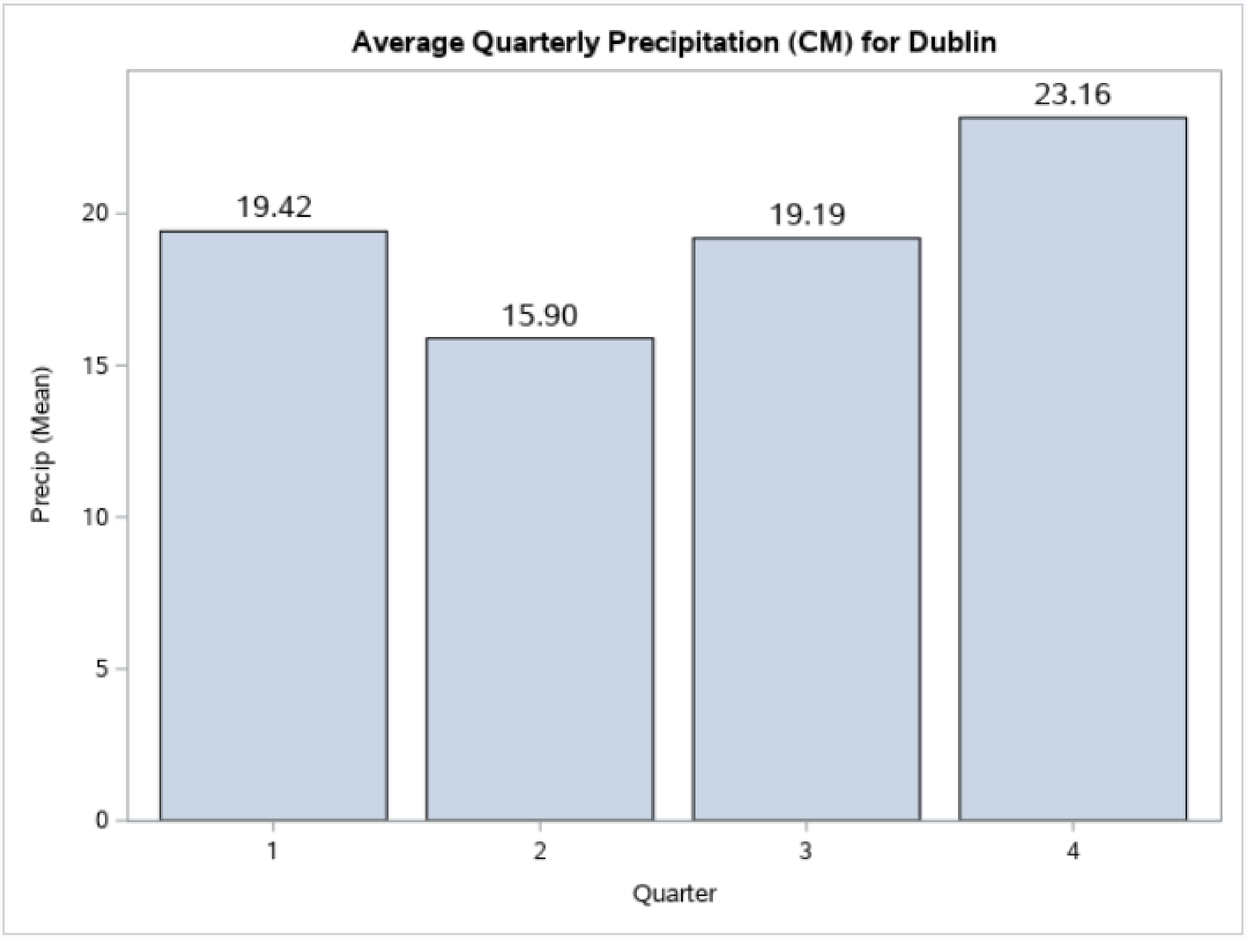


Total rows: 20 Total columns: 4

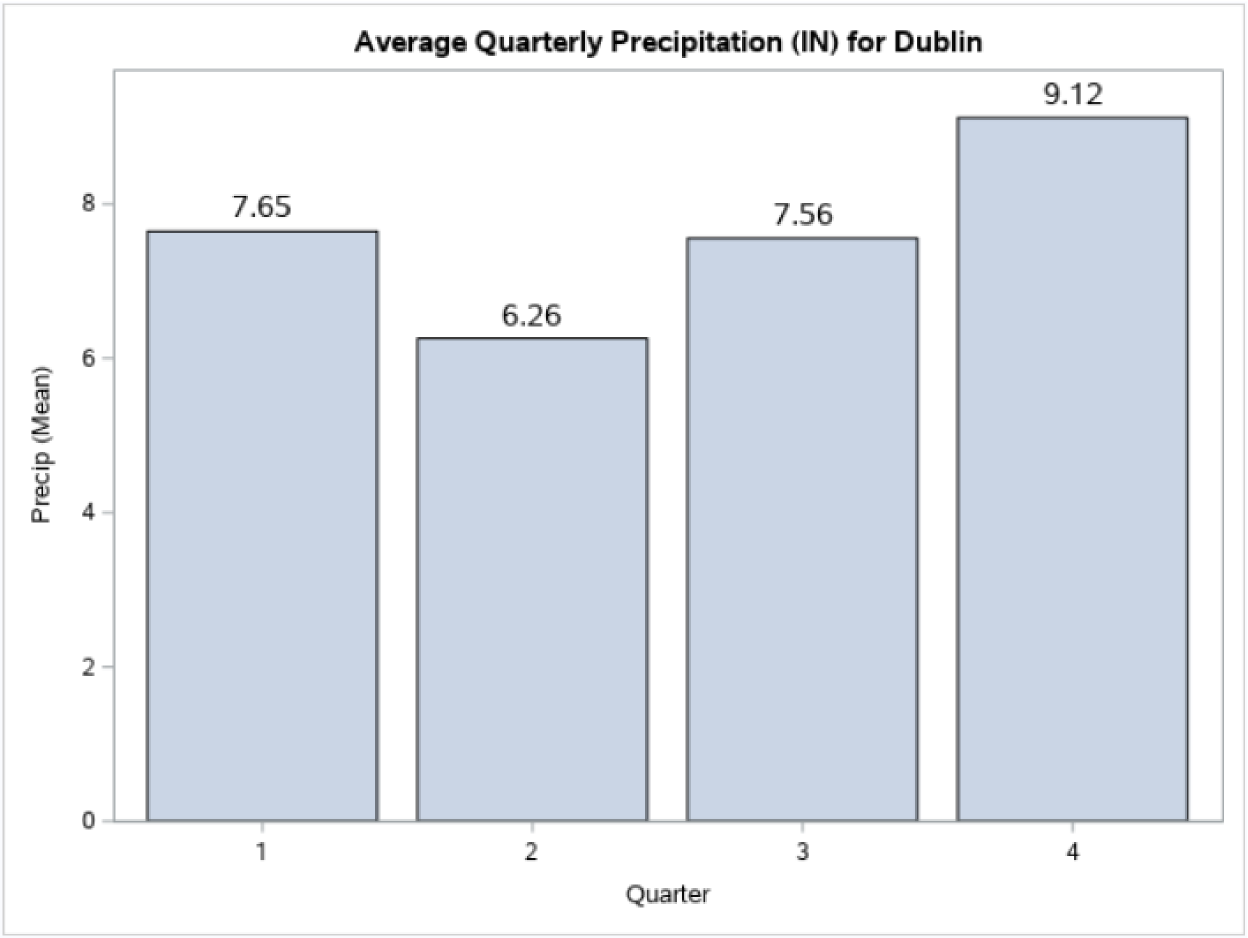
	City	Year	Quarter	Precip
1	Dublin	2013	1	22.71
2	Dublin	2013	2	13.26
3	Dublin	2013	3	15.14
4	Dublin	2013	4	19.99
5	Dublin	2014	1	24.69
6	Dublin	2014	2	13.97
7	Dublin	2014	3	19.74
8	Dublin	2014	4	30.05
9	Dublin	2015	1	14.30
10	Dublin	2015	2	16.61
11	Dublin	2015	3	19.94
12	Dublin	2015	4	34.49
13	Dublin	2016	1	19.38
14	Dublin	2016	2	20.17
15	Dublin	2016	3	18.92
16	Dublin	2016	4	12.01
17	Dublin	2017	1	16.03
18	Dublin	2017	2	15.47
19	Dublin	2017	3	22.23
20	Dublin	2017	4	19.25

```
data work.DublinPrecipRotate;
   set pg3.weather_dublinmadrid_monthly5yr
      (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   array P[4] PrecipQ1-PrecipQ4;
```

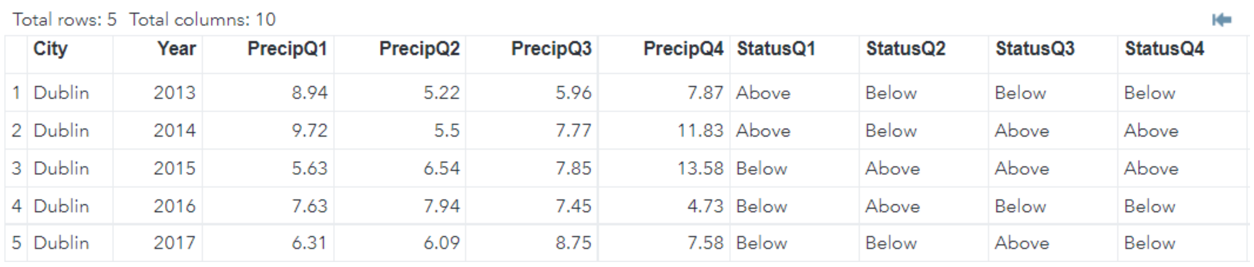
```
      do Quarter=1 to 4;
         Precip=P[Quarter]*2.54;
         output;
      end;
   format Precip 6.2;
   drop PrecipQ1-PrecipQ4;
run;

title 'Average Quarterly Precipitation (CM) for Dublin';
proc sgplot data=work.DublinPrecipRotate;
   vbar Quarter / response=Precip stat=mean datalabel
                datalabelattrs=(size=12pt);
   format Precip 6.2;
run;
title;
```



```
*************************************************************;
*  Processing One-Dimensional Arrays: Part 2              *;
*************************************************************;


data work.DublinPrecipRotate;
   set pg3.weather_dublinmadrid_monthly5yr
     (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   array P[4] PrecipQ1-PrecipQ4;
   do Quarter=1 to 4;
     Precip=P[Quarter];
     output;
   end;
   format Precip 6.2;
   drop PrecipQ1-PrecipQ4;
run;

title 'Average Quarterly Precipitation (IN) for Dublin';
proc sgplot data=work.DublinPrecipRotate;
   vbar Quarter / response=Precip stat=mean datalabel
              datalabelattrs=(size=12pt);
   format Precip 6.2;
run;
title;
```



```
*****************************************************************;
*  Demo                                   *;
*  1) In the DATA step, notice the three ARRAY statements. The *;
*    array P references existing columns of quarterly        *;
*    precipitation read from the input table. The array PAvg  *;
*    creates and references new numeric columns with initial  *;
*    values representing quarterly averages more than five    *;
*    years. The array Status references new character columns *;
*    that are being created with a byte size of 5.          *;
*  2) In the DO loop, add three conditional statements to      *;
*    create the values for the status columns based on the    *;
*    comparison of the precipitation columns with the average *;
*    precipitation columns.                              *;
*      if P[i] > PAvg[i] then Status[i]='Above';            *;
*      else if P[i] < PAvg[i] then Status[i]='Below';        *;
*      else if P[i] = PAvg[i] then Status[i]='Same';         *;
*  3) Run the DATA step and view the output table. Notice the  *;
*    redundant rows for the PAvg1 through PAvg4 columns.      *;
*  4) Add a DROP statement to eliminate the average           *;
*    precipitation columns.                              *;
*      drop PAvgQ1-PAvgQ4;                               *;
*  5) Run the DATA step. Verify that the output table contains *;
*    the four precipitation columns and the four status       *;
*    columns but not the four average precipitation columns.  *;
*  6) Alternatively, delete the DROP statement and replace the *;
*    syntax of PAvgQ1-PAvgQ4 with _TEMPORARY_ in the ARRAY    *;
*    statement for the PAvg array.                        *;
*      array PAvg[4] _temporary_ (7.65 , 6.26 , 7.56 , 9.12); *;
*  7) Run the DATA step and verify that the output table       *;
*    contains the same data as before the alternative        *;
*    changes.                                       *;
*  8) Self-study: The section at the end of the demo program   *;
*     is an example of storing the initial values in a macro   *;
*     variable and then referencing the macro variable in the  *;
*     ARRAY statement.                                *;
**************************************************************;


data work.DublinPrecipStatus(drop=i);
   set pg3.weather_dublinmadrid_monthly5yr
     (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   array P[4] PrecipQ1-PrecipQ4;
   array PAvg[4] PAvgQ1-PAvgQ4 (7.65 , 6.26 , 7.56 , 9.12);
   array Status[4] $ 5 StatusQ1-StatusQ4;
   do i=1 to 4;
                  if P[i] > PAvg[i] then Status[i]='Above';
                  else if P[i] < PAvg[i] then Status[i]='Below';
                  else if P[i] = PAvg[i] then Status[i]='Same';
   end;
   drop PAvgQ1-PAvgQ4;
run;
```


Total rows: 5 Total columns: 10

	City	Year	PrecipQ1	PrecipQ2	PrecipQ3	PrecipQ4	StatusQ1	StatusQ2	StatusQ3	StatusQ4
1	Dublin	2013	8.94	5.22	5.96	7.87	Above	Below	Below	Below
2	Dublin	2014	9.72	5.5	7.77	11.83	Above	Below	Above	Above
3	Dublin	2015	5.63	6.54	7.85	13.58	Below	Above	Above	Above
4	Dublin	2016	7.63	7.94	7.45	4.73	Below	Above	Below	Below
5	Dublin	2017	6.31	6.09	8.75	7.58	Below	Below	Above	Below

```
data work.DublinPrecipStatus(drop=i);
   set pg3.weather_dublinmadrid_monthly5yr
     (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   array P[4] PrecipQ1-PrecipQ4;
   array PAvg[4] _temporary_ (7.65 , 6.26 , 7.56 , 9.12);
   array Status[4] $ 5 StatusQ1-StatusQ4;
   do i=1 to 4;
                    if P[i] > PAvg[i] then Status[i]='Above';
                    else if P[i] < PAvg[i] then Status[i]='Below';
                    else if P[i] = PAvg[i] then Status[i]='Same';
   end;
run;

**********************************************************;
*  Self-study: The following example uses SQL to create a      *;
*           macro variable containing the four desired      *;
*           initial values and then references the macro    *;
*           variable in the ARRAY statement               *;
**********************************************************;

proc sql noprint;
   select round(mean(Precip),.01)
            into :averages separated by ', '
           from work.DublinPrecipRotate
           group by Quarter;
quit;

options symbolgen;

data work.DublinPrecipStatus(drop=i);
   set pg3.weather_dublinmadrid_monthly5yr
     (keep=City Year PrecipQ1-PrecipQ4);
   where City='Dublin';
   array P[4] PrecipQ1-PrecipQ4;
   array PAvg[4] _temporary_ (&averages);
   array Status[4] $ 5 StatusQ1-StatusQ4;
   do i=1 to 4;
      if P[i] > PAvg[i] then Status[i]='Above';
      else if P[i] < PAvg[i] then Status[i]='Below';
      else if P[i] = PAvg[i] then Status[i]='Same';
   end;
run;

***********************************************************;
*  LESSON 3, PRACTICE 1                                   *;
***********************************************************;
```

/*Practice Level 1: Using One-Dimensional Arrays on Numeric Data

If necessary, start SAS Studio before you begin. If you restarted your SAS session,

submit your libname.sas program to access the practice data.

The pg3.eu_occ table contains monthly occupancy rates broken down by type of property (Hotel, ShortStay, and Camp)

for European countries from January 2004 through September 2017. Calculate the percentage that each type of property

represents of the total occupancy for each month and year by country.

Open the p303p01.sas program in the practices folder. Run the program to view the European occupancy data.

Add an ARRAY statement to create an array named OccType that references the three existing columns of property type:

Hotel, ShortStay, and Camp.

Add another ARRAY statement to create an array named OccPct that creates the numeric columns

HotelPct, ShortStayPct, and CampPct.

Add a DO loop with the index column Num. Use a start value of 1 and an end value of 3.

Within the DO loop, add an assignment statement using array references to calculate the percentage of occupancy rate.

The percentage (HotelPct, ShortStayPct, and CampPct) is equal to the property type (Hotel, ShortStay, and Camp) divided by

the total occupancy (OccTotal).

Add a DROP statement to eliminate the Num column from the output table.

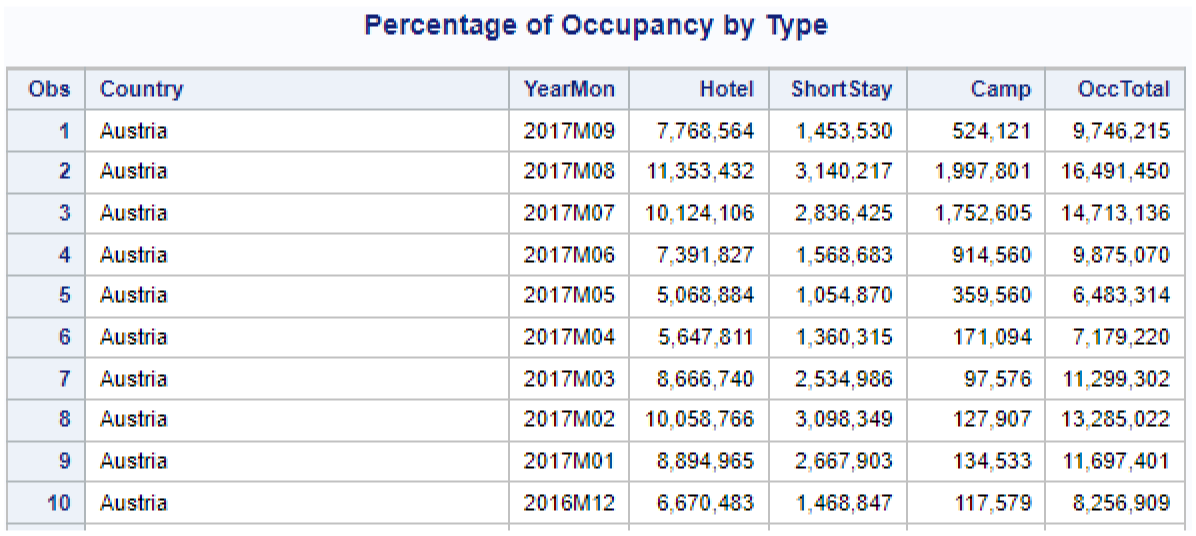
Run the program. What is the value of HotelPct for row 1?

Add to the FORMAT statement to format HotelPct, ShortStayPct, and CampPct using the PERCENT8.1 format.

Run the program.

What is the value of HotelPct for row 1?

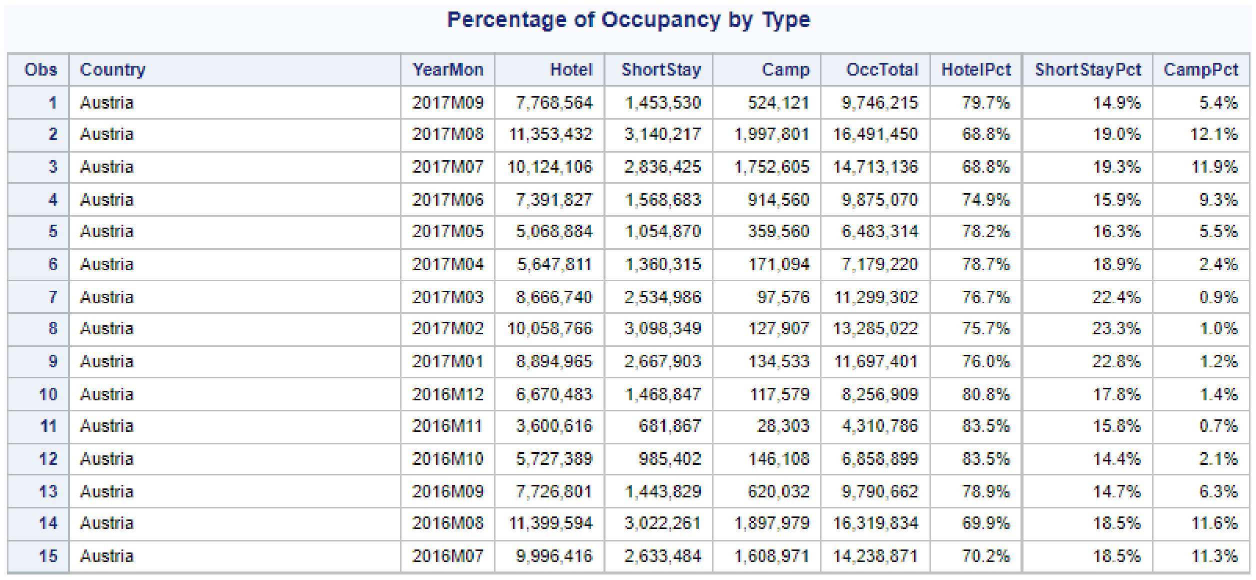
```
*/

data work.MonthlyOcc;

   set pg3.eu_occ(drop=Geo);

   OccTotal=sum(Hotel,ShortStay,Camp);


   format Hotel ShortStay Camp OccTotal comma16.;

run;


title 'Percentage of Occupancy by Type';

proc print data=work.MonthlyOcc;

run;

title;
```



Percentage of Occupancy by Type

Obs	Country	YearMon	Hotel	ShortStay	Camp	OccTotal
1	Austria	2017M09	7,768,564	1,453,530	524,121	9,746,215
2	Austria	2017M08	11,353,432	3,140,217	1,997,801	16,491,450
3	Austria	2017M07	10,124,106	2,836,425	1,752,605	14,713,136
4	Austria	2017M06	7,391,827	1,568,683	914,560	9,875,070
5	Austria	2017M05	5,068,884	1,054,870	359,560	6,483,314
6	Austria	2017M04	5,647,811	1,360,315	171,094	7,179,220
7	Austria	2017M03	8,666,740	2,534,986	97,576	11,299,302
8	Austria	2017M02	10,058,766	3,098,349	127,907	13,285,022
9	Austria	2017M01	8,894,965	2,667,903	134,533	11,697,401
10	Austria	2016M12	6,670,483	1,468,847	117,579	8,256,909

```
data work.MonthlyOcc;
   set pg3.eu_occ(drop=Geo);
   OccTotal=sum(Hotel,ShortStay,Camp);
          Array OccType[3] Hotel ShortStay Camp;
          Array OccPct[3] HotelPct ShortStayPct CampPct;
          do Num=1 to 3;
                    OccPct[Num]=OCCType[Num]/OccTotal;
          end;
   format Hotel ShortStay Camp OccTotal comma16. HotelPct ShortStayPct CampPct percent8.1;
   drop Num;
run;

title 'Percentage of Occupancy by Type';
proc print data=work.MonthlyOcc;
run;
title;
```



Percentage of Occupancy by Type

Obs	Country	YearMon	Hotel	ShortStay	Camp	OccTotal	HotelPct	ShortStayPct	CampPct
1	Austria	2017M09	7,768,564	1,453,530	524,121	9,746,215	79.7%	14.9%	5.4%
2	Austria	2017M08	11,353,432	3,140,217	1,997,801	16,491,450	68.8%	19.0%	12.1%
3	Austria	2017M07	10,124,106	2,836,425	1,752,605	14,713,136	68.8%	19.3%	11.9%
4	Austria	2017M06	7,391,827	1,568,683	914,560	9,875,070	74.9%	15.9%	9.3%
5	Austria	2017M05	5,068,884	1,054,870	359,560	6,483,314	78.2%	16.3%	5.5%
6	Austria	2017M04	5,647,811	1,360,315	171,094	7,179,220	78.7%	18.9%	2.4%
7	Austria	2017M03	8,666,740	2,534,986	97,576	11,299,302	76.7%	22.4%	0.9%
8	Austria	2017M02	10,058,766	3,098,349	127,907	13,285,022	75.7%	23.3%	1.0%
9	Austria	2017M01	8,894,965	2,667,903	134,533	11,697,401	76.0%	22.8%	1.2%
10	Austria	2016M12	6,670,483	1,468,847	117,579	8,256,909	80.8%	17.8%	1.4%
11	Austria	2016M11	3,600,616	681,867	28,303	4,310,786	83.5%	15.8%	0.7%
12	Austria	2016M10	5,727,389	985,402	146,108	6,858,899	83.5%	14.4%	2.1%
13	Austria	2016M09	7,726,801	1,443,829	620,032	9,790,662	78.9%	14.7%	6.3%
14	Austria	2016M08	11,399,594	3,022,261	1,897,979	16,319,834	69.9%	18.5%	11.6%
15	Austria	2016M07	9,996,416	2,633,484	1,608,971	14,238,871	70.2%	18.5%	11.3%

```
***********************************************************;
*  LESSON 3, PRACTICE 2                                   *;
***********************************************************;
```

/*Practice Level 2: Using One-Dimensional Arrays on Character Data

If necessary, start SAS Studio before you begin. If you restarted your SAS session,

submit your libname.sas program to access the practice data.

The pg3.test_answers table contains employee answers (A through E) to 10 test questions.

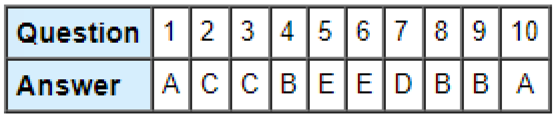
Calculate the test score for each employee by comparing their answers to the correct answers.

Open the p303p02.sas program in the practices folder.

Run the program to view the employee's answers.

Create an array named EmpAnswer that references the employee's answers to the 10 test questions.

Create a temporary array named CorAnswer that references the correct answers to the 10 questions as shown below:

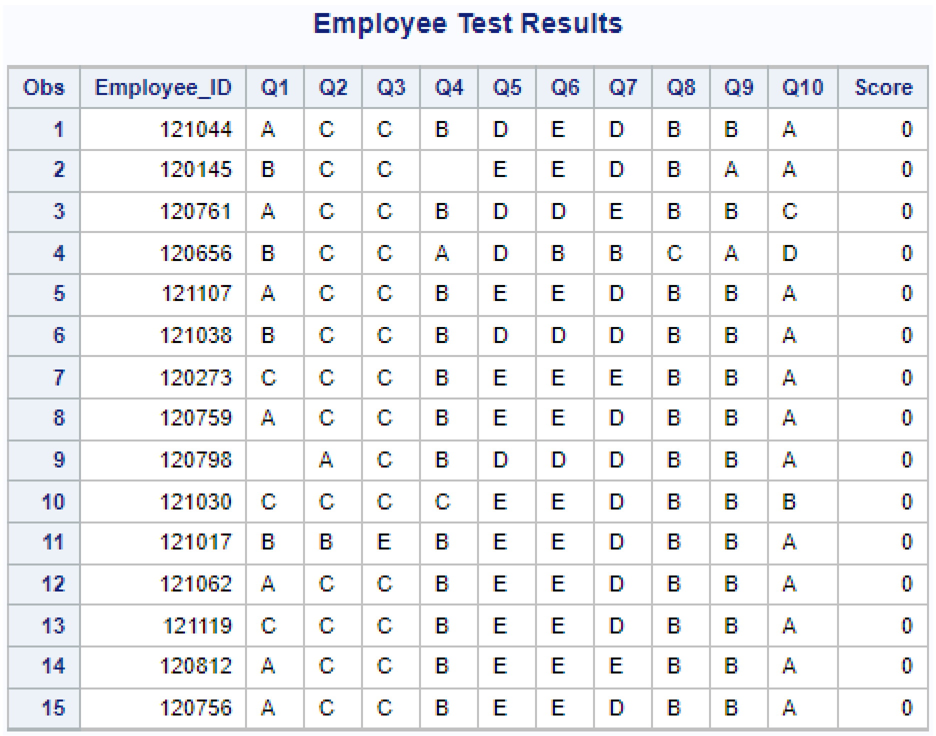


Question	1	2	3	4	5	6	7	8	9	10
Answer	A	C	C	B	E	E	D	B	B	A

Within a DO loop, use a conditional IF/THEN statement to compare the employee answer to the correct answer for the 10 questions.

If the values compare, add 1 to the column Score.

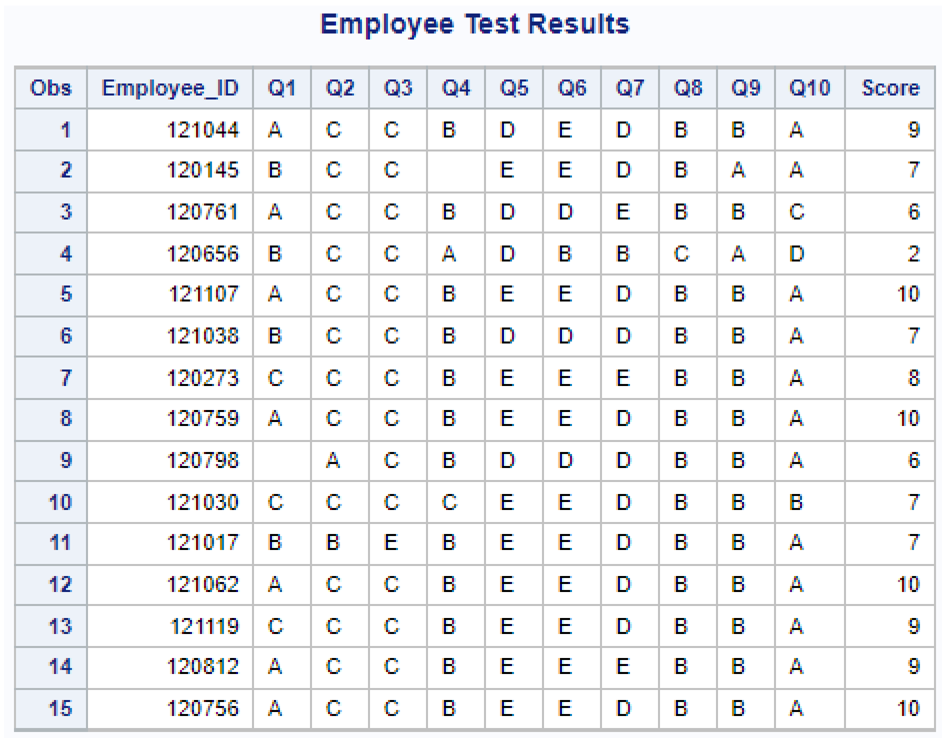
```
Eliminate the index column from the output table.

Run the program and view the results.

How many employees have a perfect score?  Note: Type a numeric value.

*/

data work.TestScores;

   set pg3.test_answers;

   Score=0;


run;


title 'Employee Test Results';

proc print data=work.TestScores;

run;

title;
```



Employee Test Results

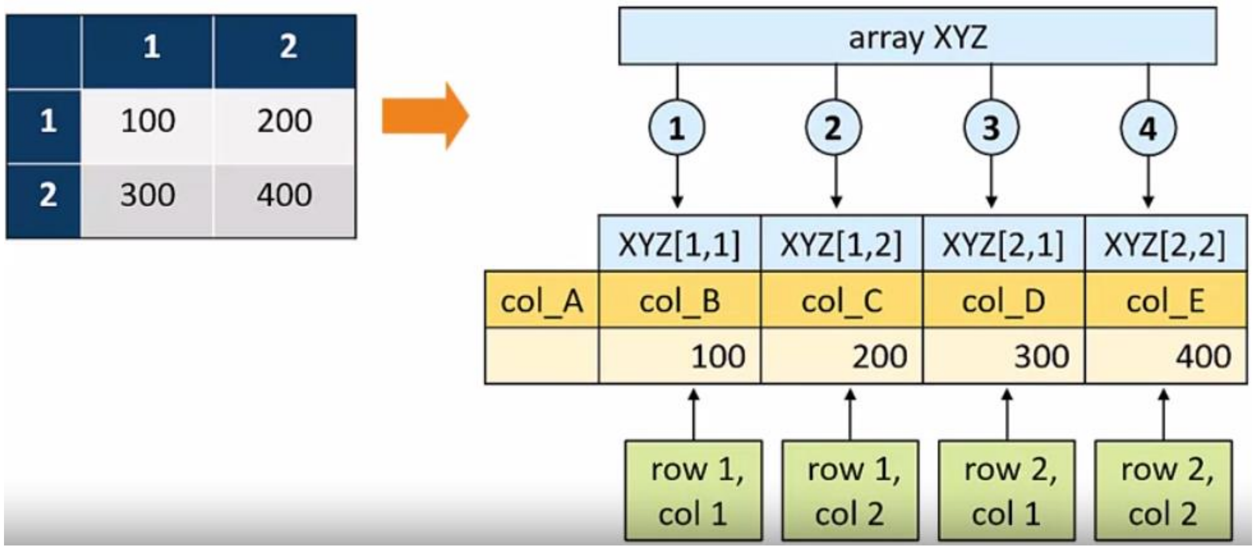
Obs	Employee_ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
1	121044	A	C	C	B	D	E	D	B	B	A	0
2	120145	B	C	C		E	E	D	B	A	A	0
3	120761	A	C	C	B	D	D	E	B	B	C	0
4	120656	B	C	C	A	D	B	B	C	A	D	0
5	121107	A	C	C	B	E	E	D	B	B	A	0
6	121038	B	C	C	B	D	D	D	B	B	A	0
7	120273	C	C	C	B	E	E	E	B	B	A	0
8	120759	A	C	C	B	E	E	D	B	B	A	0
9	120798		A	C	B	D	D	D	B	B	A	0
10	121030	C	C	C	C	E	E	D	B	B	B	0
11	121017	B	B	E	B	E	E	D	B	B	A	0
12	121062	A	C	C	B	E	E	D	B	B	A	0
13	121119	C	C	C	B	E	E	D	B	B	A	0
14	120812	A	C	C	B	E	E	E	B	B	A	0
15	120756	A	C	C	B	E	E	D	B	B	A	0

```
data work.TestScores;

   set pg3.test_answers;
   Score=0;
            array EmpAnswer[10] Q1-Q10;
            array CorAnswer[10] $ 1 _temporary_ ('A', 'C', 'C', 'B', 'E', 'E', 'D', 'B', 'B', 'A');
            do i=1 to 10;
                        if EmpAnswer[i]=CorAnswer[i] then Score+1;
            end;
            drop i;
run;

title 'Employee Test Results';
proc print data=work.TestScores;
run;
title;
```



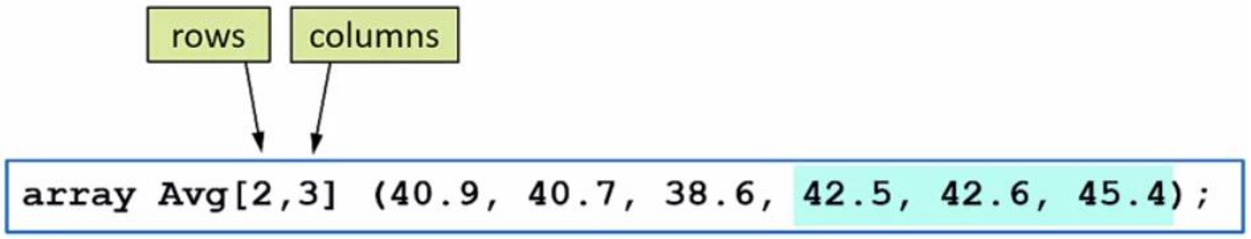
Employee Test Results

Obs	Employee_ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
1	121044	A	C	C	B	D	E	D	B	B	A	9
2	120145	B	C	C		E	E	D	B	A	A	7
3	120761	A	C	C	B	D	D	E	B	B	C	6
4	120656	B	C	C	A	D	B	B	C	A	D	2
5	121107	A	C	C	B	E	E	D	B	B	A	10
6	121038	B	C	C	B	D	D	D	B	B	A	7
7	120273	C	C	C	B	E	E	E	B	B	A	8
8	120759	A	C	C	B	E	E	D	B	B	A	10
9	120798		A	C	B	D	D	D	B	B	A	6
10	121030	C	C	C	C	E	E	D	B	B	B	7
11	121017	B	B	E	B	E	E	D	B	B	A	7
12	121062	A	C	C	B	E	E	D	B	B	A	10
13	121119	C	C	C	B	E	E	D	B	B	A	9
14	120812	A	C	C	B	E	E	E	B	B	A	9
15	120756	A	C	C	B	E	E	D	B	B	A	10

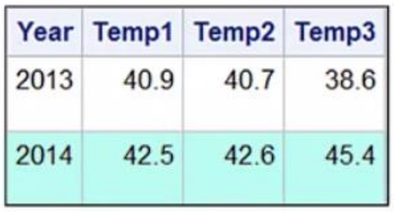


	1	2
1	100	200
2	300	400

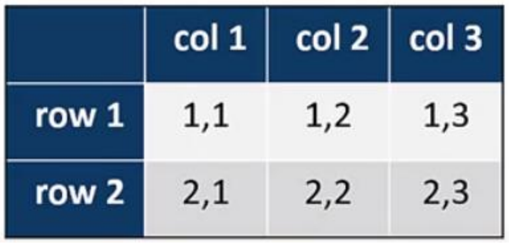
col_A	col_B	col_C	col_D	col_E
	100	200	300	400



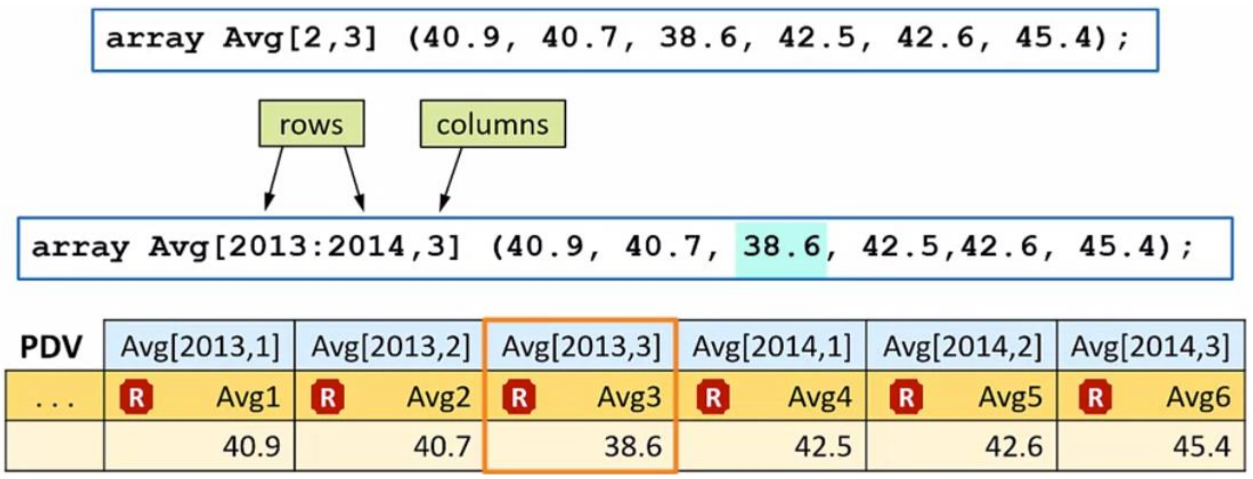
```
array Avg[2,3] (40.9, 40.7, 38.6, 42.5, 42.6, 45.4);
```



Year	Temp1	Temp2	Temp3
2013	40.9	40.7	38.6
2014	42.5	42.6	45.4



	col 1	col 2	col 3
row 1	1,1	1,2	1,3
row 2	2,1	2,2	2,3



```
array Avg[2,3] (40.9, 40.7, 38.6, 42.5, 42.6, 45.4);
```

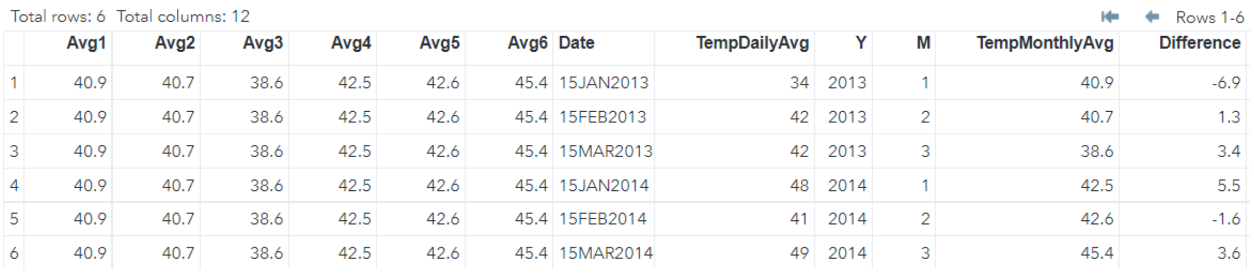
```
array Avg[2013:2014,3] (40.9, 40.7, 38.6, 42.5,42.6, 45.4);
```

PDV	Avg[2013,1]	Avg[2013,2]	Avg[2013,3]	Avg[2014,1]	Avg[2014,2]	Avg[2014,3]
. . .	Avg1	Avg2	Avg3	Avg4	Avg5	Avg6
	40.9	40.7	38.6	42.5	42.6	45.4

**;

```
*   Processing Two-Dimensional Arrays: Part 1              *;
***********************************************************;

***********************************************************;
*  Demo                                   *;
*    Note: The debugger portion of this demo must be         *;
*    performed in Enterprise Guide.                     *;
*  1) Notice the ARRAY statement that creates the            *;
*    two-dimensional array Avg. This array has two rows       *;
*    defined with year values of 2013 and 2014 and three      *;
*    columns defined with month values of 1, 2, and 3. The    *;
*    array creates six new columns, Avg1-Avg6. For simplicity *;
*    purposes, the SET statement contains a WHERE= data set   *;
*    option limiting the daily average temperatures to the    *;
*    15th day of January, February, or March for the years    *;
*    2013 and 2014.                                *;
*  2) In Enterprise Guide, click the Toggle DATA Step Debugger *;
*    toolbar button to enable debugging in the program. Click *;
*    the Debugger icon next to the DATA statement. The DATA   *;
*    Step Debugger window appears. Notice that at the         *;
*    beginning of execution the six average columns are       *;
*    populated with the initial values from the ARRAY         *;
*    statement.                                  *;
*  3) Click the Step execution to the next line toolbar button *;
*    to execute through the statements. As the SET statement  *;
*    executes, Date and TempDailyAvg are populated. The Y and *;
*    M columns are populated from the first two assignment    *;
*    statements. TempMonthlyAvg is populated by the third     *;
*    assignment statement, which locates the desired monthly  *;
*    average temperature in the array using Y and M for the   *;
*    lookup. Continue clicking through the six iterations of  *;
*    the DATA step and notice that each iteration uses the    *;
*    appropriate value from the array.                        *;
*  4) Close the DATA Step Debugger window.                    *;
*  5) Run the DATA step. Notice that the output table contains *;
*    the six average columns.                                 *;
*  6) Add _temporary_ to the ARRAY statement prior to the      *;
*    initial values.                                          *;
*      array Avg[2013:2014,3] _temporary_                     *;
*                         (40.9, 40.7, 38.6,                  *;
*                          42.5, 42.6, 45.4);                 *;
*  7) Click the Debugger icon next to the DATA statement to    *;
*    open the DATA Step Debugger window. Notice that the six  *;
*    average columns do not appear in the debugger because    *;
*    the columns are defined as temporary. Close the DATA     *;
*    Step Debugger window.                                    *;
*  8) Run the DATA step. Verify that the output table contains *;
*    the desired output.                                      *;
*  9) As an alternative, eliminate the assignment statements   *;
*    for the Y and M columns. In the assignment statement for *;
*    the TempMonthlyAvg column, use expressions for the rows  *;
*    and columns within the reference to the Avg array.       *;
*      TempMonthlyAvg=Avg[year(Date),month(Date)];            *;
* 10) Run the DATA step. Verify that the output table contains *;
*    the desired output.                                      *;
**************************************************************;

data work.DublinDaily;
```

```
   array Avg[2013:2014,3] (40.9, 40.7, 38.6,
                  42.5, 42.6, 45.4);
   set pg3.weather_dublin_daily5yr
       (where=(day(Date)=15 and
             month(Date) le 3 and
             year(Date) in (2013,2014))
        keep=Date TempDailyAvg);
   Y=year(Date);
   M=month(Date);
   TempMonthlyAvg=Avg[Y,M];
   Difference=TempDailyAvg-TempMonthlyAvg;
run;
```

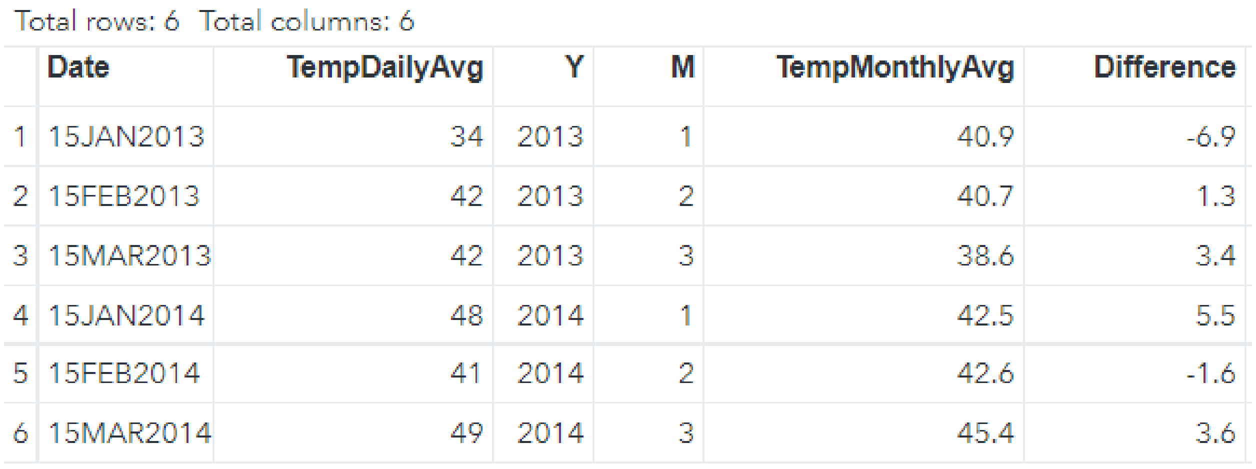


Total rows: 6 Total columns: 12 Rows 1-6

	Avg1	Avg2	Avg3	Avg4	Avg5	Avg6	Date	TempDailyAvg	Y	M	TempMonthlyAvg	Difference
1	40.9	40.7	38.6	42.5	42.6	45.4	15JAN2013	34	2013	1	40.9	-6.9
2	40.9	40.7	38.6	42.5	42.6	45.4	15FEB2013	42	2013	2	40.7	1.3
3	40.9	40.7	38.6	42.5	42.6	45.4	15MAR2013	42	2013	3	38.6	3.4
4	40.9	40.7	38.6	42.5	42.6	45.4	15JAN2014	48	2014	1	42.5	5.5
5	40.9	40.7	38.6	42.5	42.6	45.4	15FEB2014	41	2014	2	42.6	-1.6
6	40.9	40.7	38.6	42.5	42.6	45.4	15MAR2014	49	2014	3	45.4	3.6

```
data work.DublinDaily;
   array Avg[2013:2014,3] _temporary_ (40.9, 40.7, 38.6,
                  42.5, 42.6, 45.4);
   set pg3.weather_dublin_daily5yr
       (where=(day(Date)=15 and
             month(Date) le 3 and
             year(Date) in (2013,2014))
        keep=Date TempDailyAvg);
   Y=year(Date);
   M=month(Date);
   TempMonthlyAvg=Avg[Y,M];
   Difference=TempDailyAvg-TempMonthlyAvg;
```

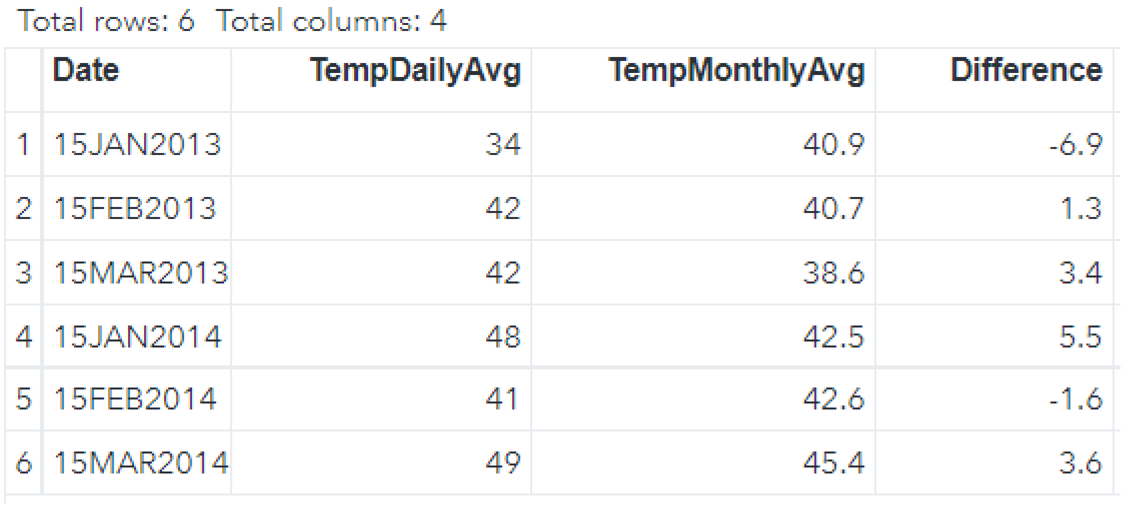
```
run;
```



Total rows: 6 Total columns: 6

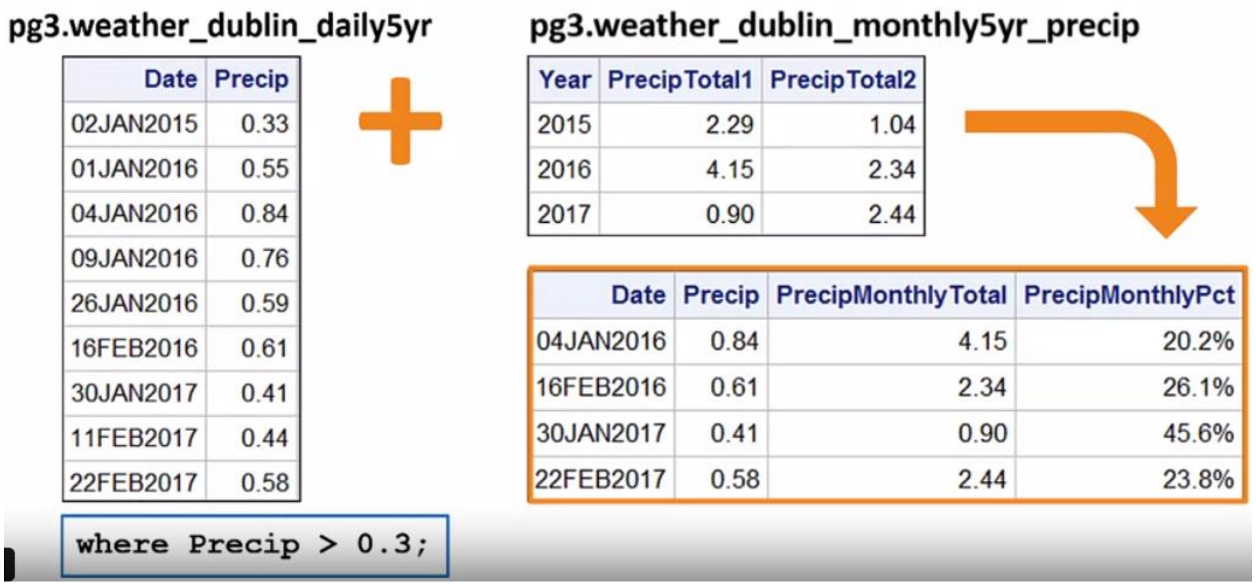
	Date	TempDailyAvg	Y	M	TempMonthlyAvg	Difference
1	15JAN2013	34	2013	1	40.9	-6.9
2	15FEB2013	42	2013	2	40.7	1.3
3	15MAR2013	42	2013	3	38.6	3.4
4	15JAN2014	48	2014	1	42.5	5.5
5	15FEB2014	41	2014	2	42.6	-1.6
6	15MAR2014	49	2014	3	45.4	3.6

```
data work.DublinDaily;
  array Avg[2013:2014,3] _temporary_ (40.9, 40.7, 38.6,
                    42.5, 42.6, 45.4);
  set pg3.weather_dublin_daily5yr
      (where=(day(Date)=15 and
          month(Date) le 3 and
          year(Date) in (2013,2014))
       keep=Date TempDailyAvg);
  TempMonthlyAvg=Avg[Year(Date),Month(Date)];
  Difference=TempDailyAvg-TempMonthlyAvg;
run;
```



Total rows: 6 Total columns: 4

	Date	TempDailyAvg	TempMonthlyAvg	Difference
1	15JAN2013	34	40.9	-6.9
2	15FEB2013	42	40.7	1.3
3	15MAR2013	42	38.6	3.4
4	15JAN2014	48	42.5	5.5
5	15FEB2014	41	42.6	-1.6
6	15MAR2014	49	45.4	3.6



pg3.weather_dublin_daily5yr

Date	Precip
02JAN2015	0.33
01JAN2016	0.55
04JAN2016	0.84
09JAN2016	0.76
26JAN2016	0.59
16FEB2016	0.61
30JAN2017	0.41
11FEB2017	0.44
22FEB2017	0.58

```
where Precip > 0.3;
```

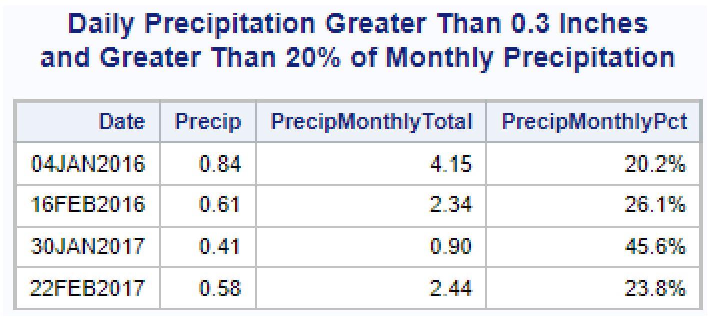
pg3.weather_dublin_monthly5yr_precip

Year	PrecipTotal1	PrecipTotal2
2015	2.29	1.04
2016	4.15	2.34
2017	0.90	2.44

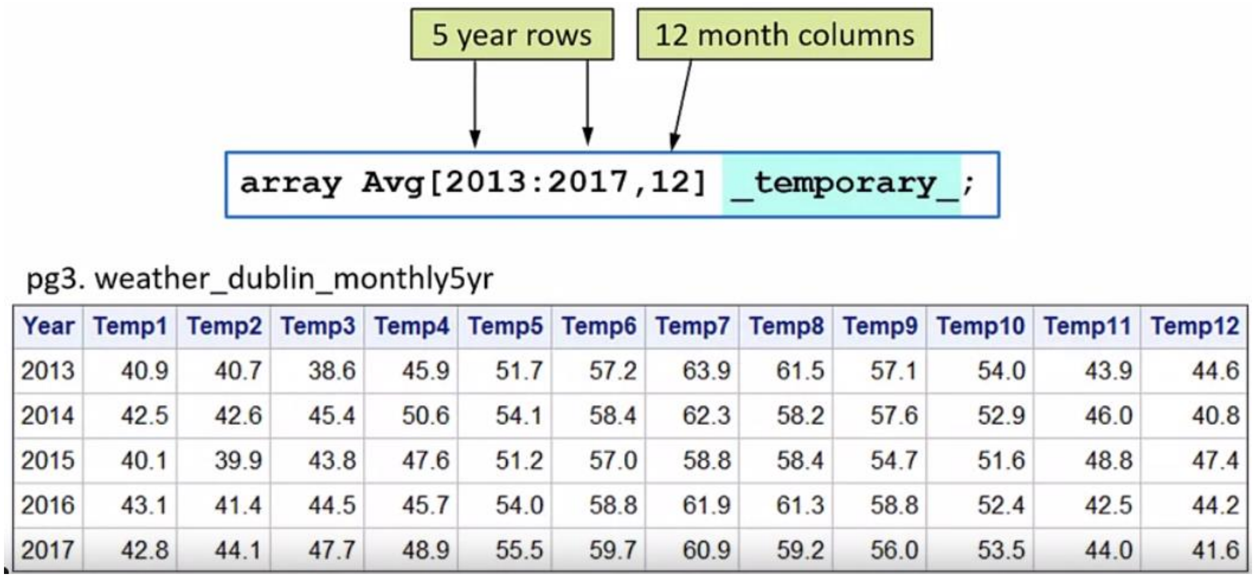
Date	Precip	PrecipMonthlyTotal	PrecipMonthlyPct
04JAN2016	0.84	4.15	20.2%
16FEB2016	0.61	2.34	26.1%
30JAN2017	0.41	0.90	45.6%
22FEB2017	0.58	2.44	23.8%

```
***********************************************************;
*  Activity 3.05                                          *;
*  1) Add an ARRAY statement to create a two-dimensional  *;
*     array.                                              *;
*     - Name the array PMT.                               *;
*     - The row dimension should reference the values     *;
*       2015, 2016, and 2017.                             *;
*     - The column dimension should reference the values  *;
*       1 to 2.                                           *;
*     - The array elements should be temporary.           *;
*     - Use the following as the six initial values:      *;
*       2.29, 1.04, 4.15, 2.34, 0.90, and 2.44.           *;
*  2) Run the program and view the results.               *;
*  3) How many dates have daily precipitation greater     *;
*     than 0.3 inches and greater than 20% of the monthly *;
*     precipitation?                                      *;
***********************************************************;

data work.DublinPrecipPct(drop=Y M);
```

```
   /* add an ARRAY statement */
         array PMT[2015:2017,2] _temporary_ (2.29, 1.04, 4.15, 2.34, 0.90, 2.44);
   set pg3.weather_dublin_daily5yr(keep=Date Precip);
   where month(Date) le 2 and year(Date) ge 2015
      and Precip > 0.3;
   Y=year(Date);
   M=month(Date);
   PrecipMonthlyTotal=PMT[Y,M];
   PrecipMonthlyPct=Precip/PrecipMonthlyTotal;
   format PrecipMonthlyPct percent8.1;
run;

title1 'Daily Precipitation Greater Than 0.3 Inches';
title2 'and Greater Than 20% of Monthly Precipitation';
proc print data=work.DublinPrecipPct noobs;
   where PrecipMonthlyPct>0.2;
run;
title;
```


Daily Precipitation Greater Than 0.3 Inches
and Greater Than 20% of Monthly Precipitation

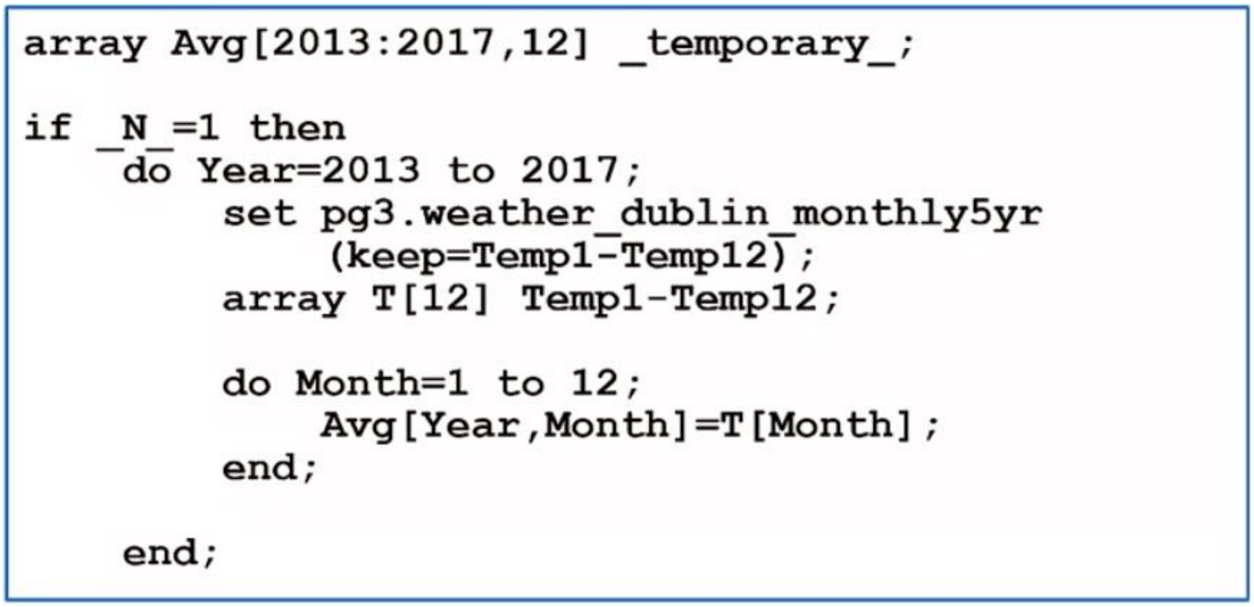
Date	Precip	PrecipMonthlyTotal	PrecipMonthlyPct
04JAN2016	0.84	4.15	20.2%
16FEB2016	0.61	2.34	26.1%
30JAN2017	0.41	0.90	45.6%
22FEB2017	0.58	2.44	23.8%



```
array Avg[2013:2017,12] _temporary_;
```

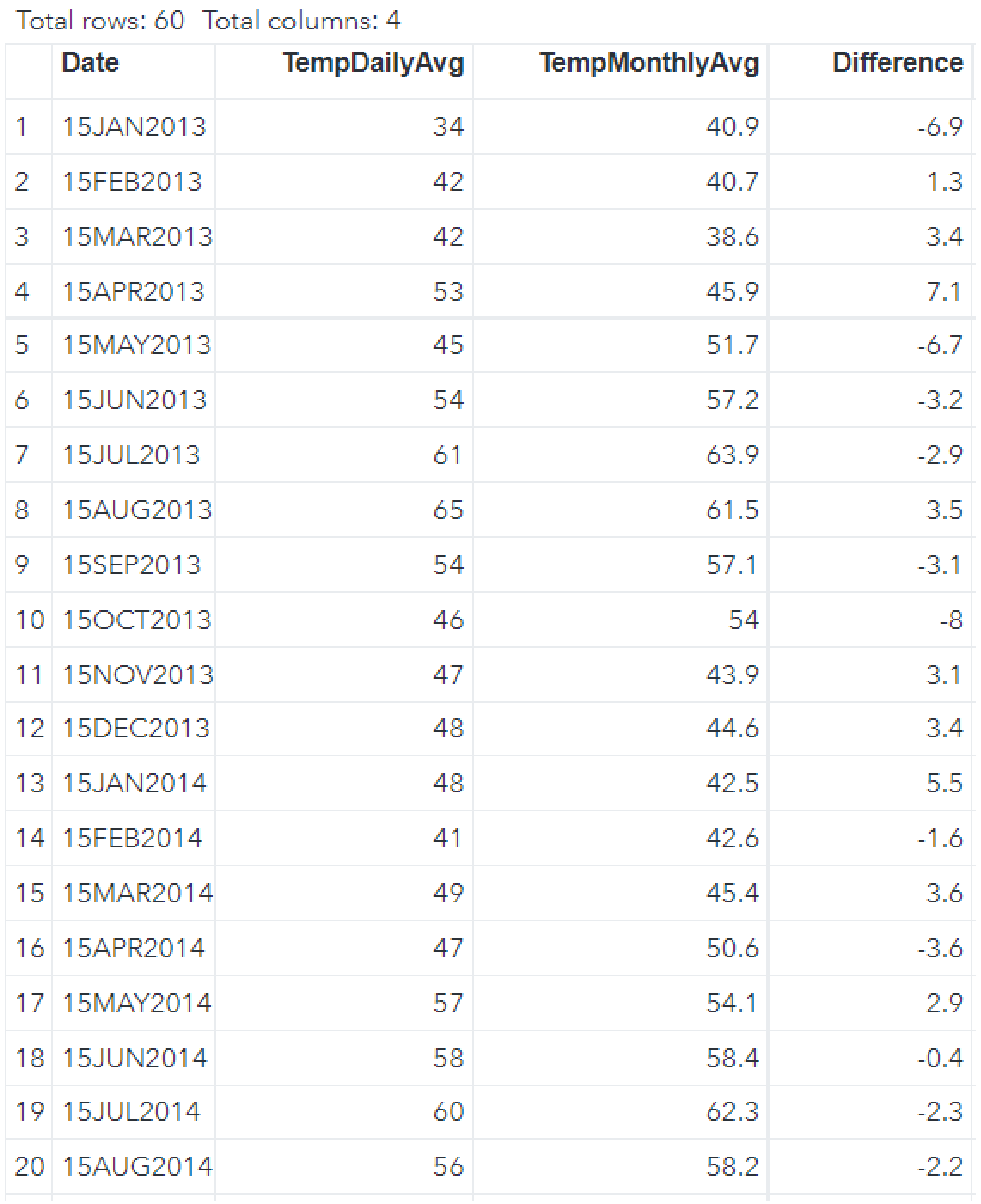
pg3. weather_dublin_monthly5yr

Year	Temp1	Temp2	Temp3	Temp4	Temp5	Temp6	Temp7	Temp8	Temp9	Temp10	Temp11	Temp12
2013	40.9	40.7	38.6	45.9	51.7	57.2	63.9	61.5	57.1	54.0	43.9	44.6
2014	42.5	42.6	45.4	50.6	54.1	58.4	62.3	58.2	57.6	52.9	46.0	40.8
2015	40.1	39.9	43.8	47.6	51.2	57.0	58.8	58.4	54.7	51.6	48.8	47.4
2016	43.1	41.4	44.5	45.7	54.0	58.8	61.9	61.3	58.8	52.4	42.5	44.2
2017	42.8	44.1	47.7	48.9	55.5	59.7	60.9	59.2	56.0	53.5	44.0	41.6



```
array Avg[2013:2017,12] _temporary_;

if _N_=1 then
   do Year=2013 to 2017;
       set pg3.weather_dublin_monthly5yr
           (keep=Temp1-Temp12);
       array T[12] Temp1-Temp12;

       do Month=1 to 12;
           Avg[Year,Month]=T[Month];
       end;

   end;
```

```
**************************************************************;
*  Processing Two-Dimensional Arrays: Part 2                *;
**************************************************************;


**************************************************************;
*  Demo                                      *;
*     Note: The debugger portion of this demo must be       *;
*     performed in Enterprise Guide.                  *;
*  1) Notice the ARRAY statement that creates the            *;
*    two-dimensional array Avg. This array is defined using 5 *;
*    rows of year values and 12 columns of month values. The  *;
*    array creates 60 new columns, Avg1 through Avg60. For    *;
*    simplicity, the second SET statement contains a WHERE=   *;
*    data set option that limits the daily average            *;
*    temperatures to the 15th day of the month.               *;
*  2) In Enterprise Guide, use the DATA Step Debugger.        *;
*    a) Click the Toggle DATA Step Debugger toolbar button to *;
*      enable debugging in the program. Click the Debugger    *;
*      icon next to the DATA statement. The DATA Step         *;
*      Debugger window appears. Notice that at the beginning *;
*      of execution there are 60 average columns with         *;
*      missing values.                                        *;
*    b) Click the Step execution to the next line toolbar     *;
*      button to execute through the statements that are      *;
*      loading the two-dimensional array. Observe that the    *;
*      values of Yr, Temp1 through Temp12, Month, and Avg1    *;
*      through Avg60 change.                                  *;
*    c) To speed through the loading of the array, put a      *;
*      watch on the Avg60 column. Click the Start/continue    *;
*      debugger execution toolbar button. This continues      *;
*      the execution until the Avg60 column has been          *;
*      changed.                                               *;
*    d) Click the Step execution to the next line toolbar     *;
*      button. Observe that the values of Date,               *;
*      TempDailyAvg, Y, M, TempMonthlyAvg, and Difference     *;
*      change.                                                *;
*    e) Continue clicking the Step execution to the next line *;
*      toolbar button until _N_ changes to 2 (the second     *;
*      iteration of the DATA step).                        *;
*    f) Notice the values of Avg1 through Avg60 are reset to  *;
*      missing. Because these values are assigned by the 12  *;
*      assignment statements, they are reset to missing at   *;
*      the beginning of each iteration.                    *;
*    g) Close the DATA Step Debugger window.                *;
*  3) Add _temporary_ to the two-dimensional ARRAY statement,  *;
*    which retains the loaded values and eliminates the      *;
*    columns from the output table.                        *;
*     array Avg[2013:2017,12] _temporary_;                 *;
*  4) Modify the IF-THEN block to eliminate the repetitive     *;
*    assignment statements.                                *;
*     if _N_=1 then do Yr=2013 to 2017;                    *;
*       set pg3.weather_dublin_monthly5yr                  *;
*          (keep=Temp1-Temp12);                            *;
*       array T[12] Temp1-Temp12;                          *;
*       do Month=1 to 12;                                  *;
*          Avg[Yr,Month]=T[Month];                         *;
*       end;                                               *;
*     end;                                                 *;
*  5) Run the DATA step. Verify that the output table contains *;
*    the desired output.                                   *;
**************************************************************;


data work.DublinDaily;
   array Avg[2013:2017,12] _temporary_;
   if _N_=1 then do Yr=2013 to 2017;
      set pg3.weather_dublin_monthly5yr(keep=Temp1-Temp12);
      array T[12] Temp1-Temp12;
      do Month=1 to 12;
                    Avg[Yr,Month]=T[Month];
      end;
   end;
   set pg3.weather_dublin_daily5yr(where=(day(Date)=15)
                          keep=Date TempDailyAvg);
   Y=year(Date);
   M=month(Date);
   TempMonthlyAvg=avg[Y,M];
   Difference=TempDailyAvg-TempMonthlyAvg;
   keep Date TempDailyAvg TempMonthlyAvg Difference;
run;

*Original;
data work.DublinDaily;
   array Avg[2013:2017,12] _TEMPORARY_;
   if _N_=1 then do Yr=2013 to 2017;
      set pg3.weather_dublin_monthly5yr(keep=Temp1-Temp12);
      Avg[Yr,1]=Temp1;
      Avg[Yr,2]=Temp2;
      Avg[Yr,3]=Temp3;
      Avg[Yr,4]=Temp4;
      Avg[Yr,5]=Temp5;
      Avg[Yr,6]=Temp6;
      Avg[Yr,7]=Temp7;
      Avg[Yr,8]=Temp8;
      Avg[Yr,9]=Temp9;
      Avg[Yr,10]=Temp10;
      Avg[Yr,11]=Temp11;
      Avg[Yr,12]=Temp12;
   end;
   set pg3.weather_dublin_daily5yr(where=(day(Date)=15)
                              keep=Date TempDailyAvg);
   Y=year(Date);
   M=month(Date);
   TempMonthlyAvg=avg[Y,M];
   Difference=TempDailyAvg-TempMonthlyAvg;
   keep Date TempDailyAvg TempMonthlyAvg Difference;
run;
```



Total rows: 60 Total columns: 4

	Date	TempDailyAvg	TempMonthlyAvg	Difference
1	15JAN2013	34	40.9	-6.9
2	15FEB2013	42	40.7	1.3
3	15MAR2013	42	38.6	3.4
4	15APR2013	53	45.9	7.1
5	15MAY2013	45	51.7	-6.7
6	15JUN2013	54	57.2	-3.2
7	15JUL2013	61	63.9	-2.9
8	15AUG2013	65	61.5	3.5
9	15SEP2013	54	57.1	-3.1
10	15OCT2013	46	54	-8
11	15NOV2013	47	43.9	3.1
12	15DEC2013	48	44.6	3.4
13	15JAN2014	48	42.5	5.5
14	15FEB2014	41	42.6	-1.6
15	15MAR2014	49	45.4	3.6
16	15APR2014	47	50.6	-3.6
17	15MAY2014	57	54.1	2.9
18	15JUN2014	58	58.4	-0.4
19	15JUL2014	60	62.3	-2.3
20	15AUG2014	56	58.2	-2.2

Array Advantages and Disadvantages

There are advantages and disadvantages when using arrays.

Advantages of Arrays	**Disadvantages of Arrays**
simplifies programs for processing repetitive code, rotating data, and performing table lookup	size of array defined at compile time
fast processing time because the array is stored in memory	cannot combine character and numeric elements in an array
potentially eliminates the need for multiple steps	must use a numeric value to reference an element
ability to use non-sorted data	returns only a single value when referencing an element

```
*************************************************************;
*  LESSON 3, PRACTICE 4                                    *;
*************************************************************;
```

/*Practice Level 1: Using a Two-Dimensional Array Based on Initial Values

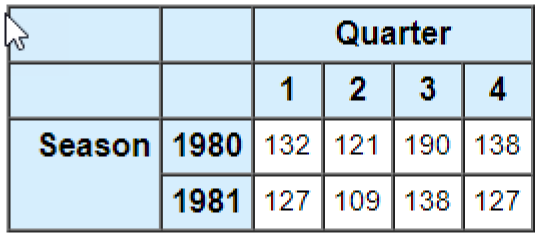
If necessary, start SAS Studio before you begin. If you restarted your SAS session,

submit your libname.sas program to access the practice data.

The pg3.storm_stats table contains statistics such as MaxWindMPH for storms from seasons 1980 to 2016.

For the storms in season 1980 and 1981, calculate the difference in a storm's MaxWindMPH compared to the quarterly maximum wind

speed per season as shown below.



		Quarter			
		1	2	3	4
Season	1980	132	121	190	138
	1981	127	109	138	127

Open the p303p04.sas program in the practices folder.

Run the program to view the MaxWindMPH for storms from seasons 1980 and 1981.

Add an ARRAY statement to create a temporary two-dimensional array named MWtable.

The row dimension is based on two rows with a lower boundary of 1980 and an upper boundary of 1981.

The column dimension is based on four columns corresponding to the four quarters.

The array elements should include the initial values as shown in the table above.

Add an assignment statement to create a column named MaxWindSQ.

The value of this column will be retrieved from the MWtable array using Season for the row dimension and Qtr

for the column dimension.

Add MaxWindSQ to the VAR statement in the PROC PRINT step.

Run the program and view the results.

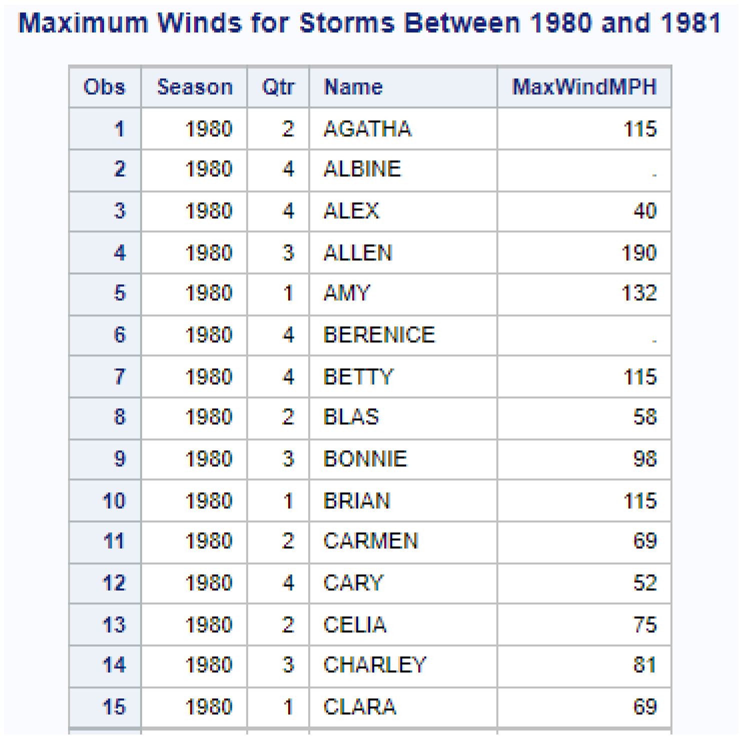
Add an assignment statement to create a column named Difference that is equal to MaxWindMPH minus MaxWindSQ.

Add Difference to the VAR statement in the PROC PRINT step.

Run the program and view the results.

What is the value of Difference for the 1980 storm named Lester (row 55)? Note: Type your answer as shown in the results.

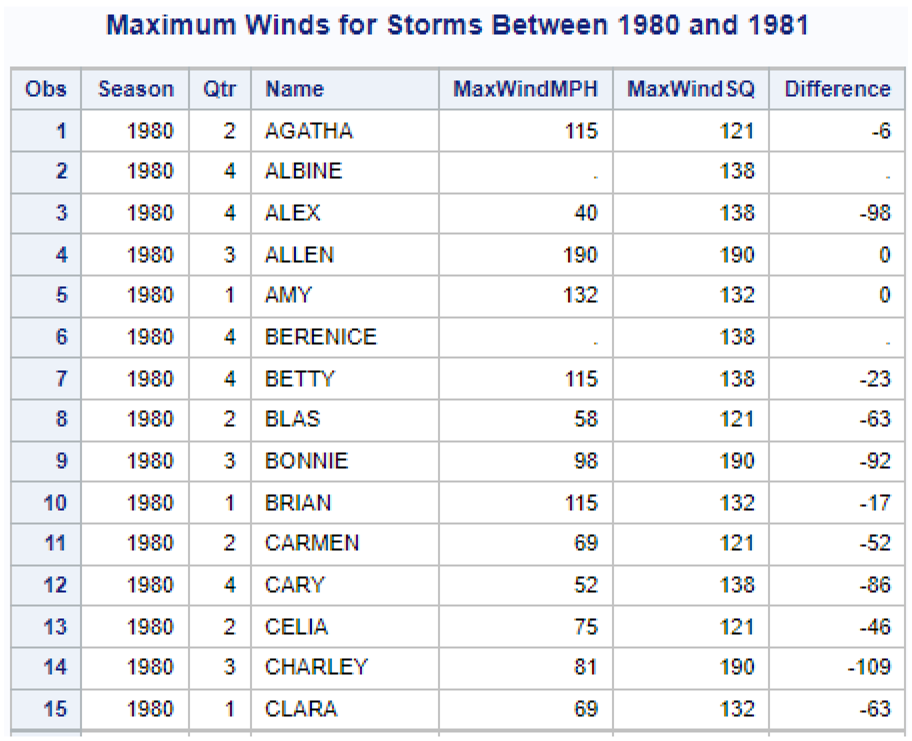
```
*/
data work.MaxWind;
   set pg3.storm_stats;
   where Season between 1980 and 1981;
   Qtr=qtr(StartDate);
run;

title 'Maximum Winds for Storms Between 1980 and 1981';
proc print data=work.MaxWind;
   var Season Qtr Name MaxWindMPH;
run;
title;
```



Maximum Winds for Storms Between 1980 and 1981

Obs	Season	Qtr	Name	MaxWindMPH
1	1980	2	AGATHA	115
2	1980	4	ALBINE	.
3	1980	4	ALEX	40
4	1980	3	ALLEN	190
5	1980	1	AMY	132
6	1980	4	BERENICE	.
7	1980	4	BETTY	115
8	1980	2	BLAS	58
9	1980	3	BONNIE	98
10	1980	1	BRIAN	115
11	1980	2	CARMEN	69
12	1980	4	CARY	52
13	1980	2	CELIA	75
14	1980	3	CHARLEY	81
15	1980	1	CLARA	69

```
data work.MaxWind;
   set pg3.storm_stats;
   where Season between 1980 and 1981;
   Qtr=qtr(StartDate);
   array MWTable[1980:1981,4] _temporary_ (132,121,190,138,127,109,138,127);
   MaxWindSQ=MWTable[Season,Qtr];
   Difference=MaxWindMPH-MaxWindSQ;
run;

title 'Maximum Winds for Storms Between 1980 and 1981';
proc print data=work.MaxWind;
   var Season Qtr Name MaxWindMPH MaxWindSQ Difference;
run;
title;
```



Maximum Winds for Storms Between 1980 and 1981

Obs	Season	Qtr	Name	MaxWindMPH	MaxWindSQ	Difference
1	1980	2	AGATHA	115	121	-6
2	1980	4	ALBINE	.	138	.
3	1980	4	ALEX	40	138	-98
4	1980	3	ALLEN	190	190	0
5	1980	1	AMY	132	132	0
6	1980	4	BERENICE	.	138	.
7	1980	4	BETTY	115	138	-23
8	1980	2	BLAS	58	121	-63
9	1980	3	BONNIE	98	190	-92
10	1980	1	BRIAN	115	132	-17
11	1980	2	CARMEN	69	121	-52
12	1980	4	CARY	52	138	-86
13	1980	2	CELIA	75	121	-46
14	1980	3	CHARLEY	81	190	-109
15	1980	1	CLARA	69	132	-63

```
***********************************************************;
*  LESSON 3, PRACTICE 5                                   *;
***********************************************************;
```

/*Practice Level 2: Using a Two-Dimensional Array Based on Loading a Table

If necessary, start SAS Studio before you begin.

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

The pg3.storm_stats table contains statistics such as MaxWindMPH for storms from seasons 1980 through 2016.

The pg3.storm_maxwindseasqtr table contains quarterly maximum wind speeds (MaxWindQ1 to MaxWindQ4) for the same seasons.

Calculate the difference in a storm's MaxWindMPH value compared to the quarterly maximum wind speed per season.

1) Open the p303p05.sas program in the practices folder.

Run the program to view MaxWindMPH for storms from season 1980 to 2016.

2) Add an ARRAY statement to create a two-dimensional temporary array named MWtable.

The row dimension is based on 37 rows with a lower boundary of 1980 and an upper boundary of 2016.

The column dimension is based on four columns corresponding to the four quarters.

3) Load the two-dimensional array with the pg3.storm_maxwindseasqtr table.

Load the array only during the first iteration of the DATA step.

Use an outer DO loop to iterate through each season.

Create a one-dimensional array referencing MaxWindQ1-MaxWindQ4.

Use an inner DO loop to iterate through the quarters.

Load the two-dimensional array by referencing the one-dimensional array.

4) Add the following assignment statements:

an assignment to create a column named MaxWindSQ.

The value of this column will be retrieved from the MWtable array using Season for the row dimension and Qtr

for the column dimension.

an assignment statement to create a column named Difference that is equal to MaxWindMPH minus MaxWindSQ.

Add MaxWindSQ and Difference to the VAR statement in the PROC PRINT step.

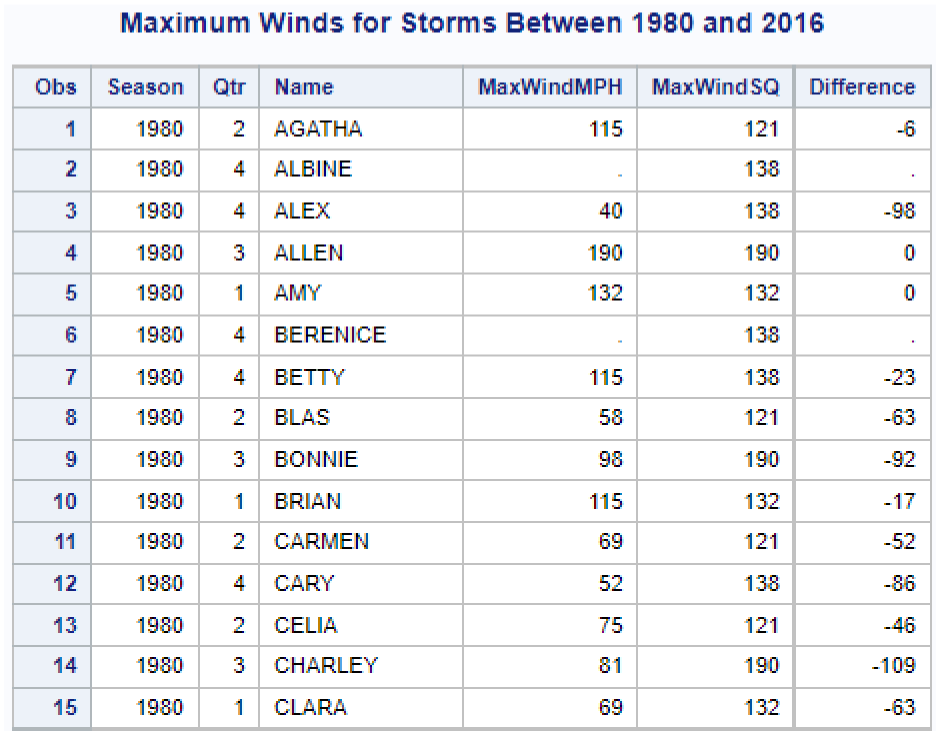
5) Remove the index columns and MaxWindQ1-MaxWindQ4.

6) Run the program and verify the results.

What is the value of Difference for the 2016 storm named Zena (row 3038)? Note: Type the answer as shown in the results.

```
*/

data work.MaxWind;

  /* add code to create and load two-dimensional array */

        array MWtable[1980:2016,4] _temporary_;

  if _N_=1 then do S=1980 to 2016;

    set pg3.storm_maxwindseasqtr(keep=MaxWindQ1-MaxWindQ4);

    array MWQ[4] MaxWindQ1-MaxWindQ4;

    do Q=1 to 4;

                MWtable[S,Q]=MWQ[Q];

    end;

  end;

  set pg3.storm_stats;
```

```
   Qtr=qtr(StartDate);
   /* add code to use two-dimensional array */
           MaxWindSQ=MWtable[Season,Qtr];
   Difference=MaxWindMPH-MaxWindSQ;
           drop S Q MaxWindQ1-MaxWindQ4;
run;

title 'Maximum Winds for Storms Between 1980 and 2016';
proc print data=work.MaxWind;
   var Season Qtr Name MaxWindMPH MaxWindSQ Difference;
run;
title;
```



Maximum Winds for Storms Between 1980 and 2016

Obs	Season	Qtr	Name	MaxWindMPH	MaxWindSQ	Difference
1	1980	2	AGATHA	115	121	-6
2	1980	4	ALBINE	.	138	.
3	1980	4	ALEX	40	138	-98
4	1980	3	ALLEN	190	190	0
5	1980	1	AMY	132	132	0
6	1980	4	BERENICE	.	138	.
7	1980	4	BETTY	115	138	-23
8	1980	2	BLAS	58	121	-63
9	1980	3	BONNIE	98	190	-92
10	1980	1	BRIAN	115	132	-17
11	1980	2	CARMEN	69	121	-52
12	1980	4	CARY	52	138	-86
13	1980	2	CELIA	75	121	-46
14	1980	3	CHARLEY	81	190	-109
15	1980	1	CLARA	69	132	-63