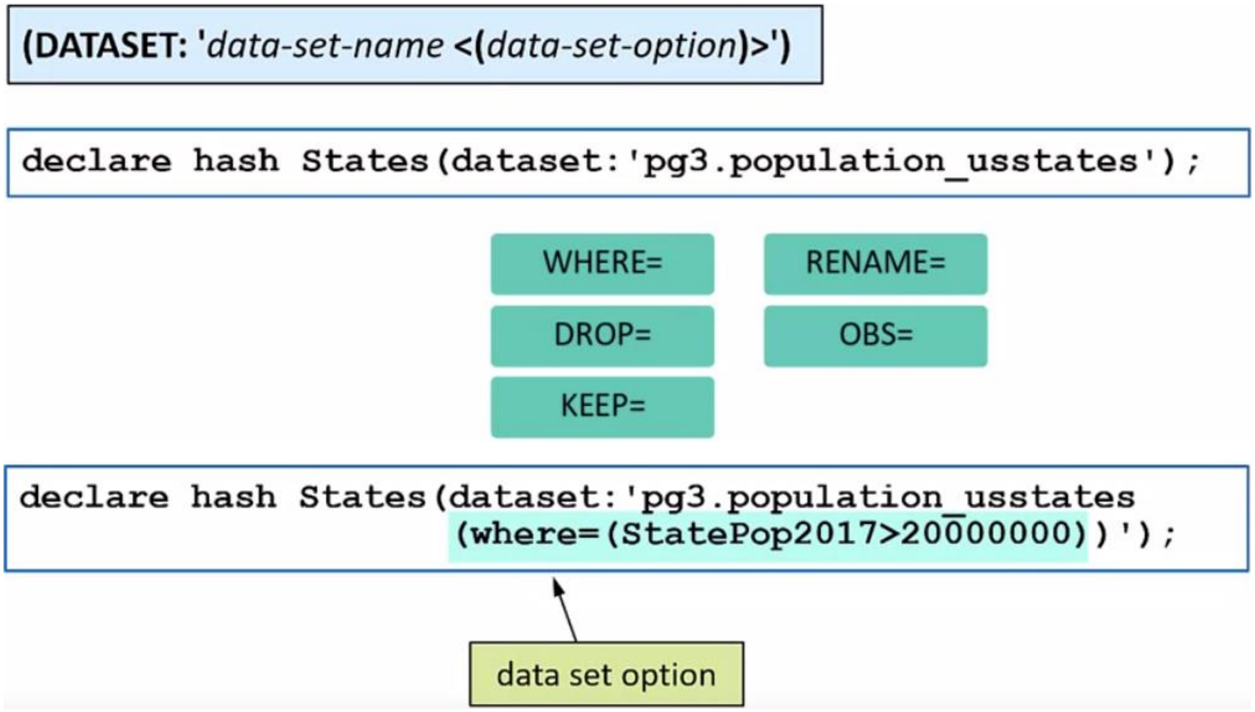
SAS Advanced Programming

SAP3 SAS Advanced Programming Techniques

SAP304 Hash Objects: Declaring Hash Objects, Defining Hash Objects, Finding Key Values in a Hash Object, Writing a Hash Object to a table, and Using Hash Iterator Objects

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. */
```



(DATASET: '*data-set-name* <(*data-set-option*)>')

```
declare hash States(dataset:'pg3.population_usstates');
```

```
declare hash States(dataset:'pg3.population_usstates
                    (where=(StatePop2017>20000000))');
```

Specifying the DATASET Argument

When specifying the DATASET argument, the table name can be specified as a character literal in quotation marks, a character column, or a character expression. Refer to demo p304d01 for examples of the three methods.

Character Literal:

declare hash States(dataset: 'pg3.population_usstates');

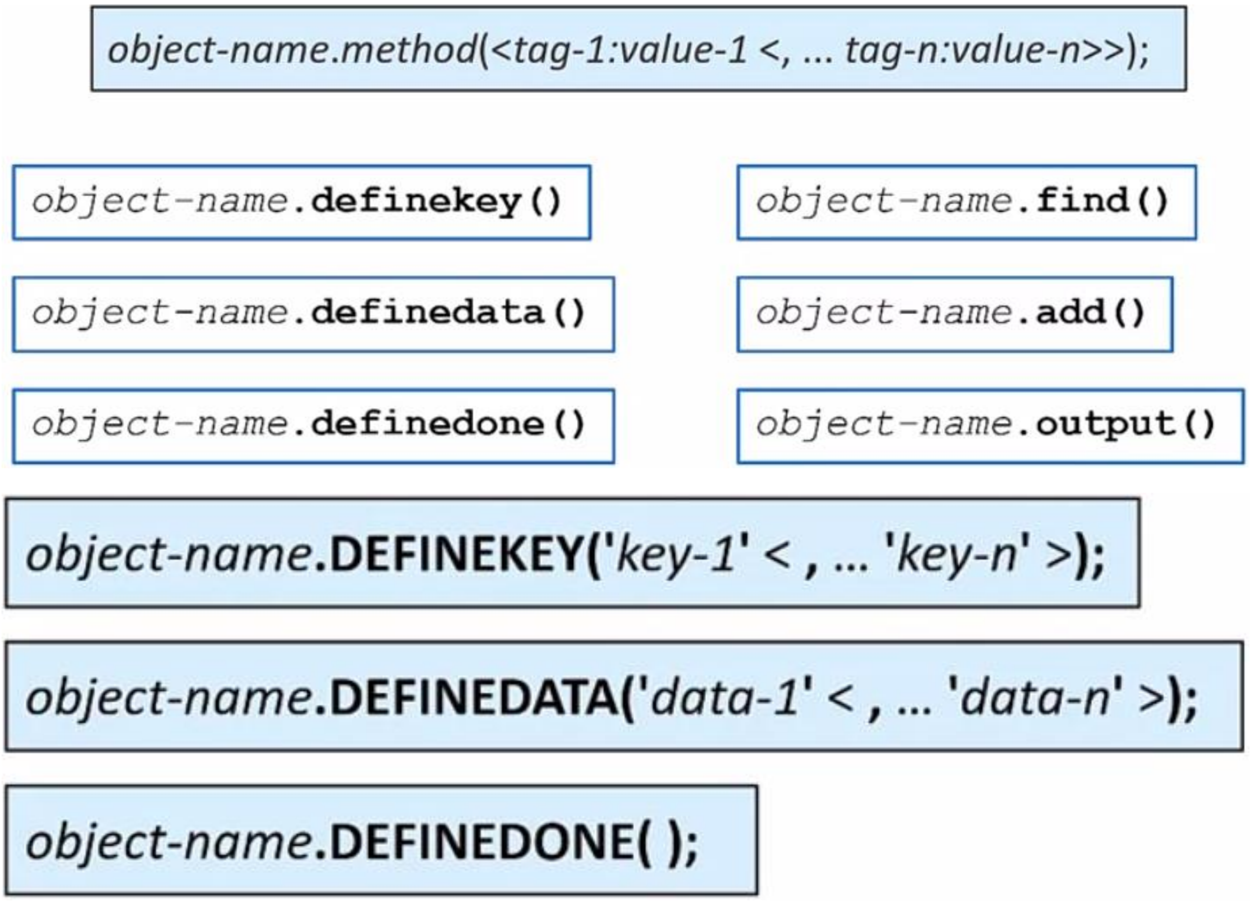
Character Column:

declare hash States(dataset: tablename);

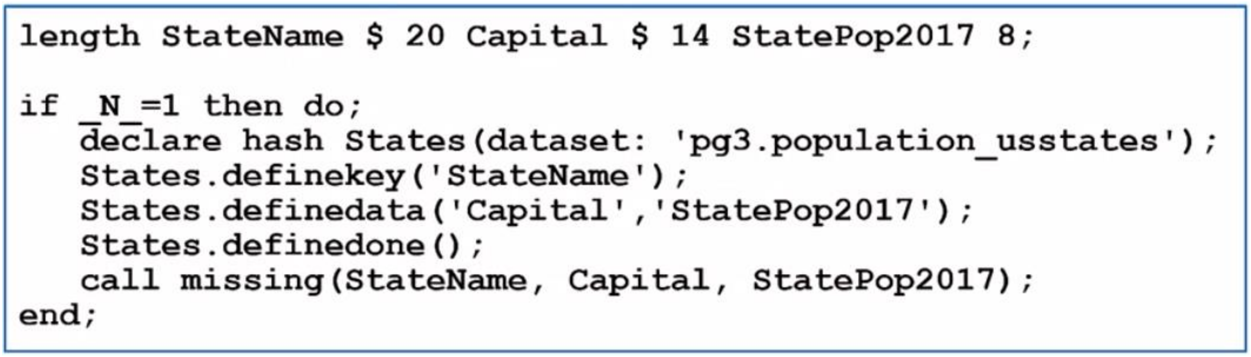
Character Expression:

declare hash States(dataset: cats('pg3.population_',location));

If the table contains duplicate keys, the default is to load the first instance in the hash object. Subsequent instances are ignored. Use the DUPLICATE argument to store the last instance in the hash object or write an error message to the SAS log. Use the MULTIDATA argument to allow multiple data items for a key.



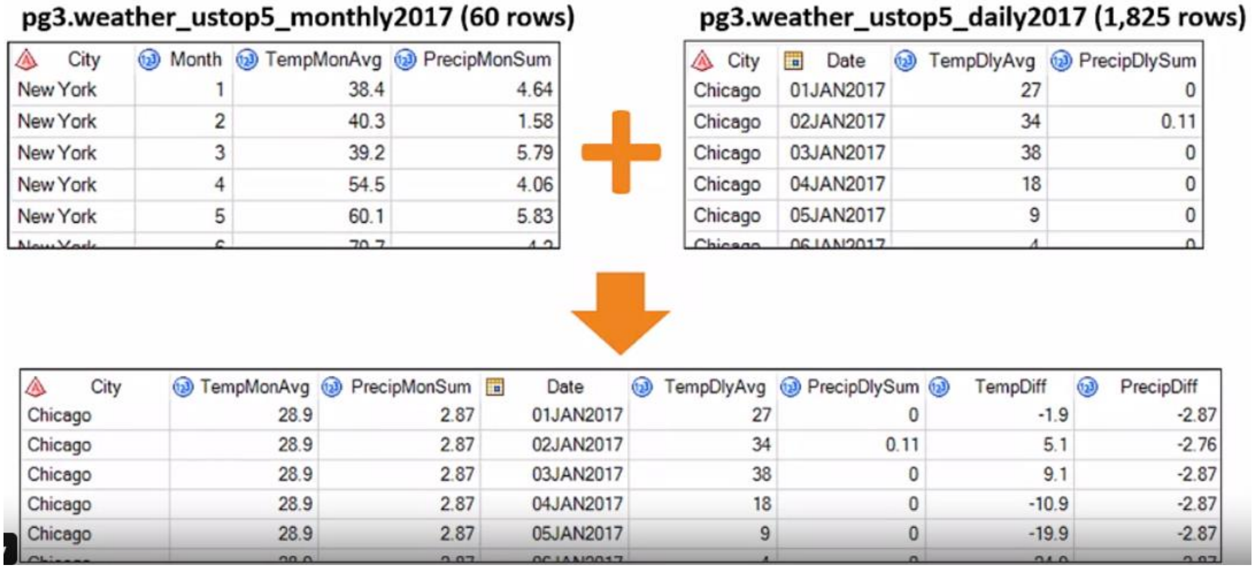
```
object-name.method(<tag-1:value-1 <, ... tag-n:value-n>>);

object-name.definekey()        object-name.find()
object-name.definedata()       object-name.add()
object-name.definedone()       object-name.output()

object-name.DEFINEKEY('key-1' < , ... 'key-n' >);

object-name.DEFINEDATA('data-1' < , ... 'data-n' >);

object-name.DEFINEDONE( );
```



```
length StateName $ 20 Capital $ 14 StatePop2017 8;

if _N_=1 then do;
   declare hash States(dataset: 'pg3.population_usstates');
   States.definekey('StateName');
   States.definedata('Capital','StatePop2017');
   States.definedone();
   call missing(StateName, Capital, StatePop2017);
end;
```

```
************************************************************;
*  Declaring and Defining a Hash Object                    *;
************************************************************;


************************************************************;
*  Demo                                               *;
*  1) Notice the syntax to declare and define the States hash  *;
*     object with one key component and two data components.   *;
*  2) Run the program. View the error in the SAS log          *;
*     concerning the undeclared key symbol StateName. This     *;
*     error appears because the key component has not been     *;
*     defined as a column in the PDV.                        *;
*     Note: You can open the DATA Step Debugger in SAS        *;
*     Enterprise Guide to see that none of the hash object     *;
*     components have been defined in the PDV.               *;
*  3) After the DATA statement, add a LENGTH statement to      *;
*     define StateName as character with a byte size of 20.    *;
*       length StateName $ 20;                               *;
*  4) Run the program. View the error in the SAS log          *;
*     concerning the undeclared data symbol Capital. This      *;
*     error appears because the data component has not been    *;
*     defined as a column in the PDV.                        *;
*  5) Add to the LENGTH statement to define Capital as        *;
*    character with a byte size of 14 and StatePop2017 as     *;
*    numeric with a byte size of 8.                          *;
*      length StateName $ 20 Capital $ 14 StatePop2017 8;     *;
*  6) Run the program. View the uninitialized notes in the SAS *;
*    log. These notes appear because SAS does not see any     *;
*    syntax that is assigning values to the three columns in  *;
*    the PDV.                                                *;
*      NOTE: Variable StateName is uninitialized.            *;
*      NOTE: Variable Capital is uninitialized.              *;
*      NOTE: Variable StatePop2017 is uninitialized.         *;
*  7) To eliminate the uninitialized notes, add a CALL MISSING *;
*    statement inside the DO block to assign a missing value  *;
*    to the three columns during the first iteration of the   *;
*    DATA step.                                              *;
*      call missing(StateName, Capital, StatePop2017);       *;
*  8) Run the program. Verify that 50 rows were read to load   *;
*    the hash object and that a new table has been created    *;
*    with 1 row of empty data.                               *;
*      NOTE: There were 50 observations read from the data    *;
*           set PG3.POPULATION_USSTATES.                     *;
*      NOTE: The data set WORK.STATECITYPOPULATION has 1      *;
*           observations and 3 variables.                    *;
*  9) As an alternative, delete the LENGTH and CALL MISSING    *;
*    statements. Add a conditional SET statement to the       *;
*    beginning of the DO block. This statement is never       *;
*    executed, but it makes a spot in the PDV for every       *;
*    column that is in the table.                            *;
*       if 0 then set pg3.population_usstates;               *;
* 10) Run the program. Verify that 50 rows were read to load   *;
*    the hash object and that a new table has been created    *;
*    with 1 row of empty data.                        *;
**********************************************************;


data work.StateCityPopulation;
          length Statename $ 20 Capital $14 StatePop2017 8;
   if _N_=1 then do;
     declare hash States(dataset: 'pg3.population_usstates');
     States.definekey('StateName');
     States.definedata('Capital','StatePop2017');
     States.definedone();
     call Missing(Statename, Capital, StatePop2017);
   end;
run;
```

```
NOTE: There were 50 observations read from the data set
PG3.POPULATION_USSTATES.
 NOTE: The data set WORK.STATECITYPOPULATION has 1 observations and 3
variables.
```



pg3.weather_ustop5_monthly2017 (60 rows)

City	Month	TempMonAvg	PrecipMonSum
New York	1	38.4	4.64
New York	2	40.3	1.58
New York	3	39.2	5.79
New York	4	54.5	4.06
New York	5	60.1	5.83

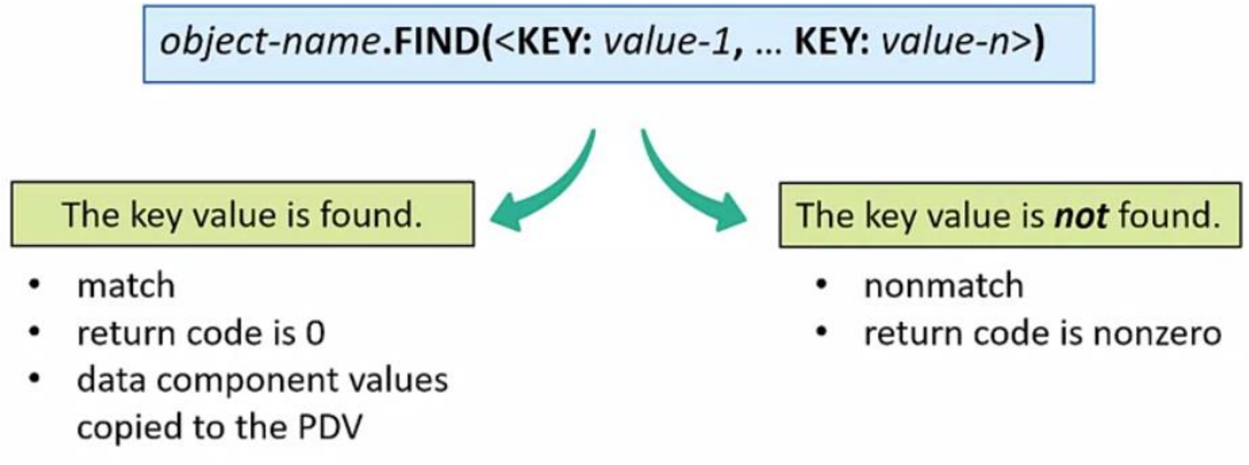
pg3.weather_ustop5_daily2017 (1,825 rows)

City	Date	TempDlyAvg	PrecipDlySum
Chicago	01JAN2017	27	0
Chicago	02JAN2017	34	0.11
Chicago	03JAN2017	38	0
Chicago	04JAN2017	18	0
Chicago	05JAN2017	9	0

City	TempMonAvg	PrecipMonSum	Date	TempDlyAvg	PrecipDlySum	TempDiff	PrecipDiff
Chicago	28.9	2.87	01JAN2017	27	0	-1.9	-2.87
Chicago	28.9	2.87	02JAN2017	34	0.11	5.1	-2.76
Chicago	28.9	2.87	03JAN2017	38	0	9.1	-2.87
Chicago	28.9	2.87	04JAN2017	18	0	-10.9	-2.87
Chicago	28.9	2.87	05JAN2017	9	0	-19.9	-2.87

```
*********************************************************;
*  Activity 4.02                             *;
*  1) Add two statements to the DATA step for the Monthly *;
*    hash object:                            *;
*    - Add a DEFINEKEY method referencing the keys of    *;
*      City and Month.                           *;
*        object-name.DEFINEKEY('key-1','key-2');        *;
*    - Add a DEFINEDATA method referencing the data of   *;
*      TempMonAvg and PrecipMonSum.                   *;
*        object-name.DEFINEDATA('data-1','data-2');      *;
*  2) Run the DATA step and confirm no errors in your SAS *;
*    log. How many rows were read from the input table   *;
*    into the hash object?                          *;
*********************************************************;

data work.Top5TempPrecip;
   length City $ 11 Month TempMonAvg PrecipMonSum 8;
   if _N_=1 then do;
      declare hash Monthly(dataset: 'pg3.weather_ustop5_monthly2017');
      /* add DEFINEKEY method for City and Month */
            Monthly.definekey('City','Month');
      /* add DEFINEDATA method for TempMonAvg and PrecipMonSum */
            Monthly.definedata('TempMonAvg','PrecipMonSum');
      Monthly.definedone();
      call missing (City, Month, TempMonAvg, PrecipMonSum);
   end;
run;
```

```
 NOTE: There were 60 observations read from the data set
PG3.WEATHER_USTOP5_MONTHLY2017.
 NOTE: The data set WORK.TOP5TEMPPRECIP has 1 observations and 4
variables.
```



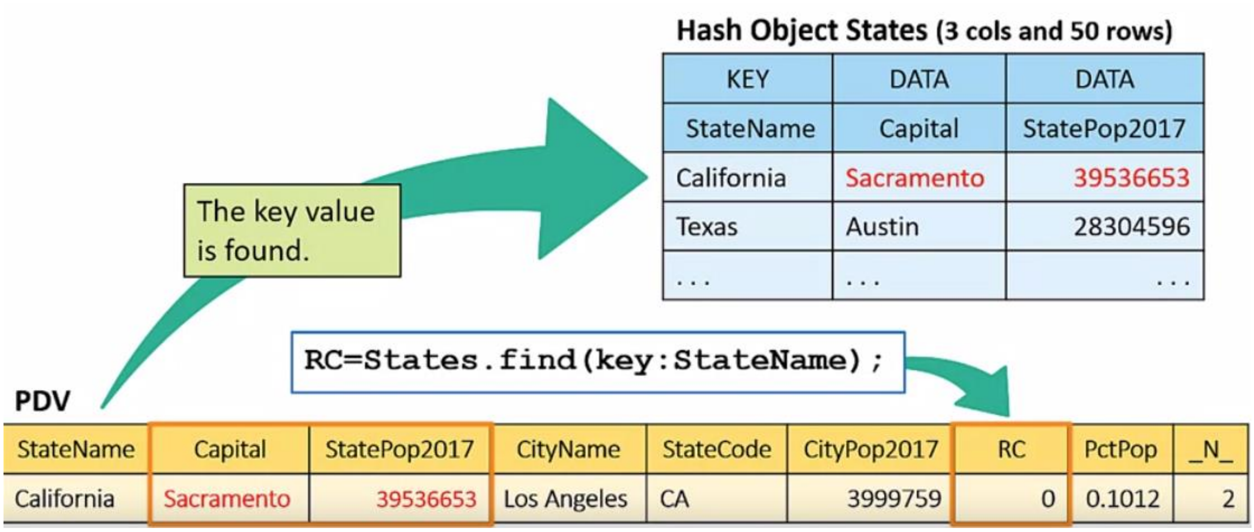
object-name.**FIND(<KEY:** *value-1*, ... **KEY:** *value-n*>**)**

The key value is found.

- match
- return code is 0
- data component values copied to the PDV

The key value is ***not*** found.

- nonmatch
- return code is nonzero



Hash Object States **(3 cols and 50 rows)**

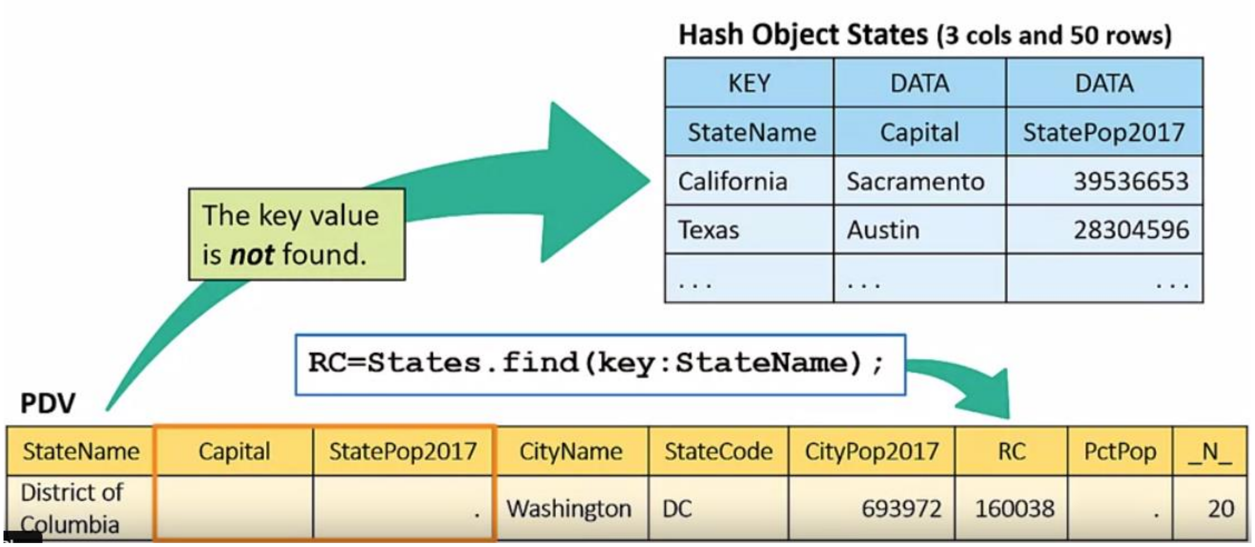
KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

The key value is found.

```
RC=States.find(key:StateName);
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop	_N_
California	Sacramento	39536653	Los Angeles	CA	3999759	0	0.1012	2



Hash Object States (3 cols and 50 rows)

KEY	DATA	DATA
StateName	Capital	StatePop2017
California	Sacramento	39536653
Texas	Austin	28304596
. . .	. . .	. . .

The key value is ***not*** found.

```
RC=States.find(key:StateName);
```

PDV

StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop	_N_
District of Columbia		.	Washington	DC	693972	160038	.	20

```
***************************************************************;
*  Performing a Table Lookup with the FIND Method            *;
***************************************************************;


data work.Top5TempPrecip;
   length City $ 11 Month TempMonAvg PrecipMonSum 8;
   if _N_=1 then do;
      declare hash Monthly(dataset: 'pg3.weather_ustop5_monthly2017');
      Monthly.definekey('City','Month');
      Monthly.definedata('TempMonAvg','PrecipMonSum');
      Monthly.definedone();
               call missing (City, Month, TempMonAvg, PrecipMonSum);
   end;
   set pg3.weather_ustop5_daily2017;
   Monthly.find(key:City,key:month(Date));
   TempDiff=TempDlyAvg-TempMonAvg;
   PrecipDiff=PrecipDlySum-PrecipMonSum;
   drop Month;
run;
```

```
***************************************************************;
*  Demo                                        *;
*  1) Notice that rows are being read from the               *;
*     pg3.population_uscities table and that StateCode is      *;
*     being used to create StateName.                      *;
*  2) Highlight and run the DATA step. View the output table   *;
*     and notice that the values of Capital and StatePop2017   *;
*     are missing. There is no syntax in the DATA step for the *;
*     StateName value to be searched in the hash object.      *;
*  3) Add an RC assignment statement after the StateName      *;
*     assignment statement to find the values of StateName in  *;
*     the hash object.                                 *;
*       RC=States.find(key:StateName);                      *;
*  4) Uncomment the PctPop assignment statement and the FORMAT *;
*     statement.                                       *;
*  5) Run the DATA step. Observe that the value of RC is 0 for *;
*     all rows except row 20. The StateName value District of  *;
*     Columbia does not have a match in the hash object.      *;
*     Notice the values of Capital and StatePop2017 are       *;
*     missing for row 20. The values were reinitialized to     *;
*     missing at the beginning of the iteration, and no values *;
*     were retrieved from the hash object.                  *;
*  6) As an alternative, delete the LENGTH and CALL MISSING    *;
*     statements. Add a conditional SET statement to the      *;
*     beginning of the DO block.                        *;
*       if 0 then set pg3.population_usstates;               *;
*  7) Run the DATA step and notice the difference for row 20.  *;
*     The Capital and StatePop2017 are retained due to the     *;
*    nature of the SET statement. Therefore, you are seeing   *;
*    the previous values for these two columns.              *;
*  8) Add the following statement after the RC assignment      *;
*    statement to set the values of Capital and StatePop2017  *;
*    to missing when a match is not found. Run the program    *;
*    and confirm the results.                               *;
*      if RC ne 0 then call missing(Capital, StatePop2017);   *;
*  9) Modify the DATA step to create an additional table that  *;
*    contains the population data for only the capital       *;
*    cities.                                            *;
*      data work.StateCityPopulation work.CapitalPopulation;  *;
*         ...                                           *;
*         output work.StateCityPopulation;                  *;
*         if Capital=CityName then                         *;
*            output work.CapitalPopulation;                *;
* 10) Run the DATA step. Verify that work.StateCityPopulation  *;
*    contains 19,500 rows and work.CapitalPopulation contains *;
*    50 rows. What would need to be added to the program if   *;
*    we wanted work.CapitalPopulation in sorted order by      *;
*    descending PctPop values?                              *;
*************************************************************;

data work.StateCityPopulation;
   length StateName $ 20 Capital $ 14 StatePop2017 8;
   if _N_=1 then do;
      declare hash States(dataset: 'pg3.population_usstates');
      States.definekey('StateName');
      States.definedata('Capital','StatePop2017');
      States.definedone();
```

```
     call missing(StateName, Capital, StatePop2017);
  end;
  set pg3.population_uscities;
  StateName=stnamel(StateCode);
         RC=States.find(Key:Statename);
  PctPop=CityPop2017/StatePop2017;
  format StatePop2017 comma14. PctPop percent8.1;
run;
```

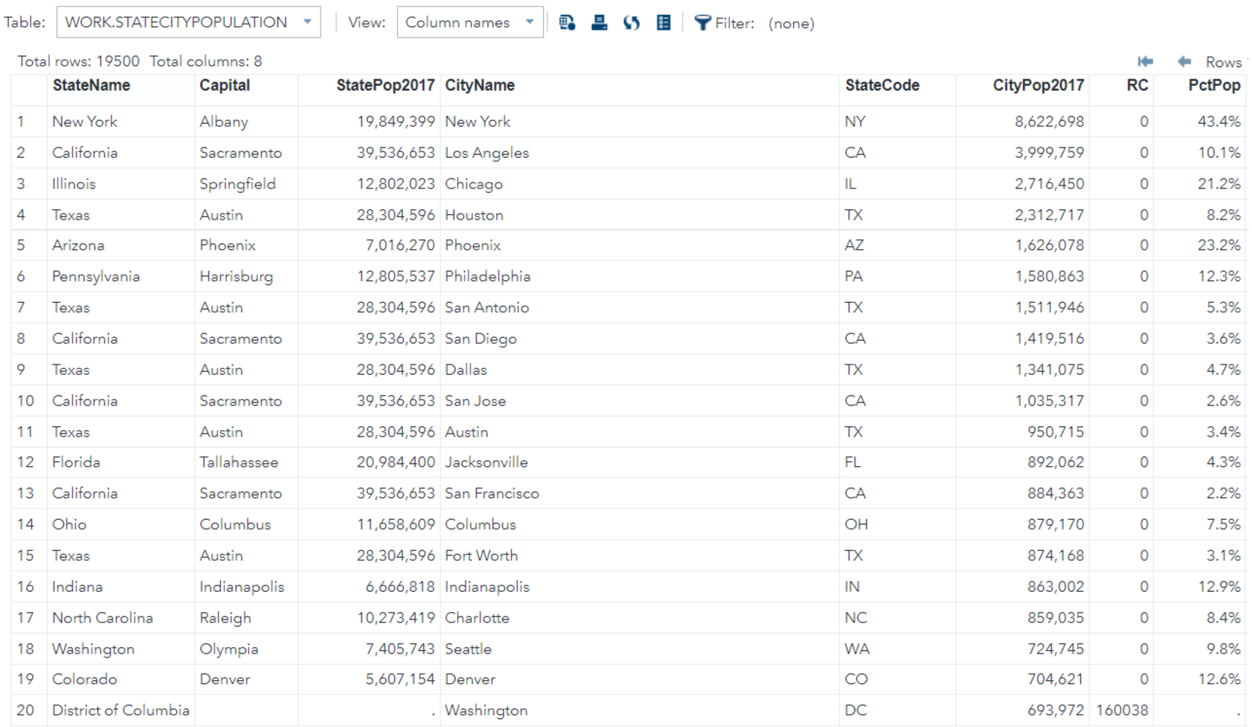


Table: WORK.STATECITYPOPULATION View: Column names Filter: (none)

Total rows: 19500 Total columns: 8

	StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop
1	New York	Albany	19,849,399	New York	NY	8,622,698	0	43.4%
2	California	Sacramento	39,536,653	Los Angeles	CA	3,999,759	0	10.1%
3	Illinois	Springfield	12,802,023	Chicago	IL	2,716,450	0	21.2%
4	Texas	Austin	28,304,596	Houston	TX	2,312,717	0	8.2%
5	Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	0	23.2%
6	Pennsylvania	Harrisburg	12,805,537	Philadelphia	PA	1,580,863	0	12.3%
7	Texas	Austin	28,304,596	San Antonio	TX	1,511,946	0	5.3%
8	California	Sacramento	39,536,653	San Diego	CA	1,419,516	0	3.6%
9	Texas	Austin	28,304,596	Dallas	TX	1,341,075	0	4.7%
10	California	Sacramento	39,536,653	San Jose	CA	1,035,317	0	2.6%
11	Texas	Austin	28,304,596	Austin	TX	950,715	0	3.4%
12	Florida	Tallahassee	20,984,400	Jacksonville	FL	892,062	0	4.3%
13	California	Sacramento	39,536,653	San Francisco	CA	884,363	0	2.2%
14	Ohio	Columbus	11,658,609	Columbus	OH	879,170	0	7.5%
15	Texas	Austin	28,304,596	Fort Worth	TX	874,168	0	3.1%
16	Indiana	Indianapolis	6,666,818	Indianapolis	IN	863,002	0	12.9%
17	North Carolina	Raleigh	10,273,419	Charlotte	NC	859,035	0	8.4%
18	Washington	Olympia	7,405,743	Seattle	WA	724,745	0	9.8%
19	Colorado	Denver	5,607,154	Denver	CO	704,621	0	12.6%
20	District of Columbia		.	Washington	DC	693,972	160038	.

```
data work.StateCityPopulation work.CapitalPopulation;
  if _N_=1 then do;
     if 0 then set pg3.population_usstates;
     declare hash States(dataset: 'pg3.population_usstates');
     States.definekey('StateName');
     States.definedata('Capital','StatePop2017');
     States.definedone();
     call missing(StateName, Capital, StatePop2017);
  end;
  set pg3.population_uscities;
  StateName=stnamel(StateCode);
          RC=States.find(Key:Statename);
          if RC ne 0 then call missing(Capital, StatePop2017);
  PctPop=CityPop2017/StatePop2017;
  output work.StateCityPopulation;
  if Capital=CityName then output work.CapitalPopulation;
  format StatePop2017 comma14. PctPop percent8.1;
run;
```

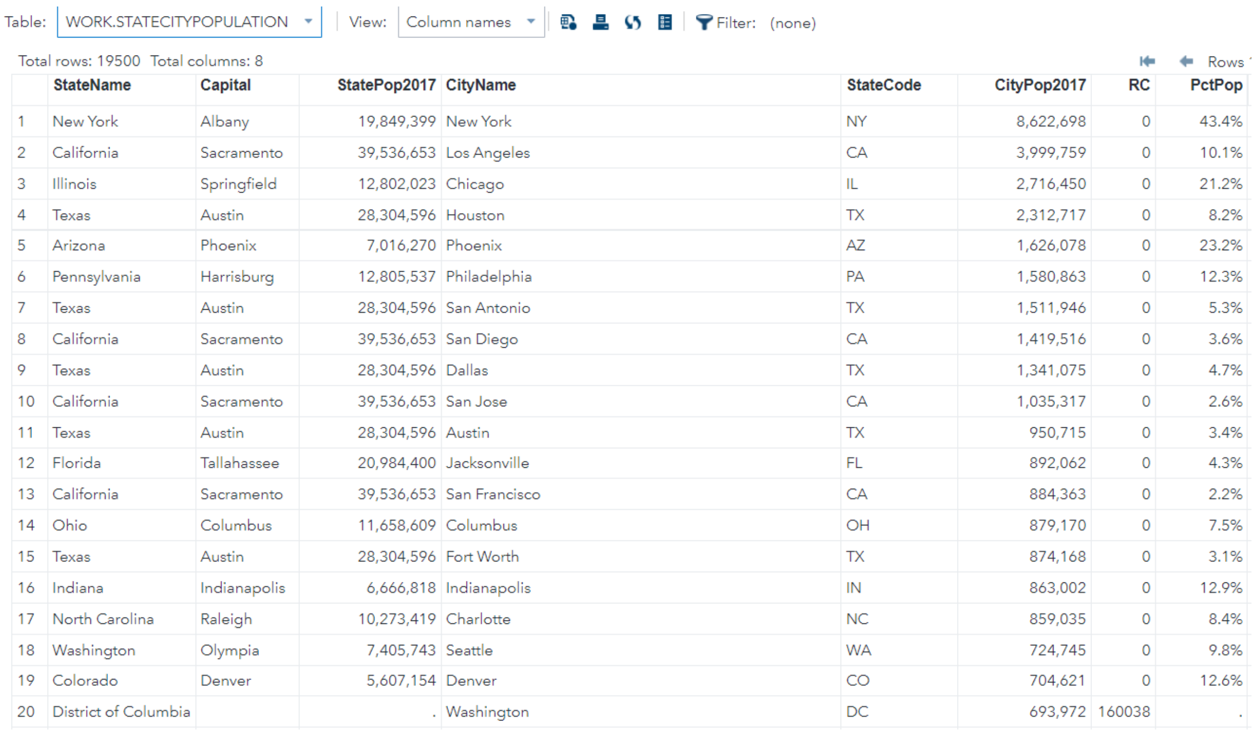


Table: WORK.STATECITYPOPULATION View: Column names Filter: (none)

Total rows: 19500 Total columns: 8

	StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop
1	New York	Albany	19,849,399	New York	NY	8,622,698	0	43.4%
2	California	Sacramento	39,536,653	Los Angeles	CA	3,999,759	0	10.1%
3	Illinois	Springfield	12,802,023	Chicago	IL	2,716,450	0	21.2%
4	Texas	Austin	28,304,596	Houston	TX	2,312,717	0	8.2%
5	Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	0	23.2%
6	Pennsylvania	Harrisburg	12,805,537	Philadelphia	PA	1,580,863	0	12.3%
7	Texas	Austin	28,304,596	San Antonio	TX	1,511,946	0	5.3%
8	California	Sacramento	39,536,653	San Diego	CA	1,419,516	0	3.6%
9	Texas	Austin	28,304,596	Dallas	TX	1,341,075	0	4.7%
10	California	Sacramento	39,536,653	San Jose	CA	1,035,317	0	2.6%
11	Texas	Austin	28,304,596	Austin	TX	950,715	0	3.4%
12	Florida	Tallahassee	20,984,400	Jacksonville	FL	892,062	0	4.3%
13	California	Sacramento	39,536,653	San Francisco	CA	884,363	0	2.2%
14	Ohio	Columbus	11,658,609	Columbus	OH	879,170	0	7.5%
15	Texas	Austin	28,304,596	Fort Worth	TX	874,168	0	3.1%
16	Indiana	Indianapolis	6,666,818	Indianapolis	IN	863,002	0	12.9%
17	North Carolina	Raleigh	10,273,419	Charlotte	NC	859,035	0	8.4%
18	Washington	Olympia	7,405,743	Seattle	WA	724,745	0	9.8%
19	Colorado	Denver	5,607,154	Denver	CO	704,621	0	12.6%
20	District of Columbia		.	Washington	DC	693,972	160038	.

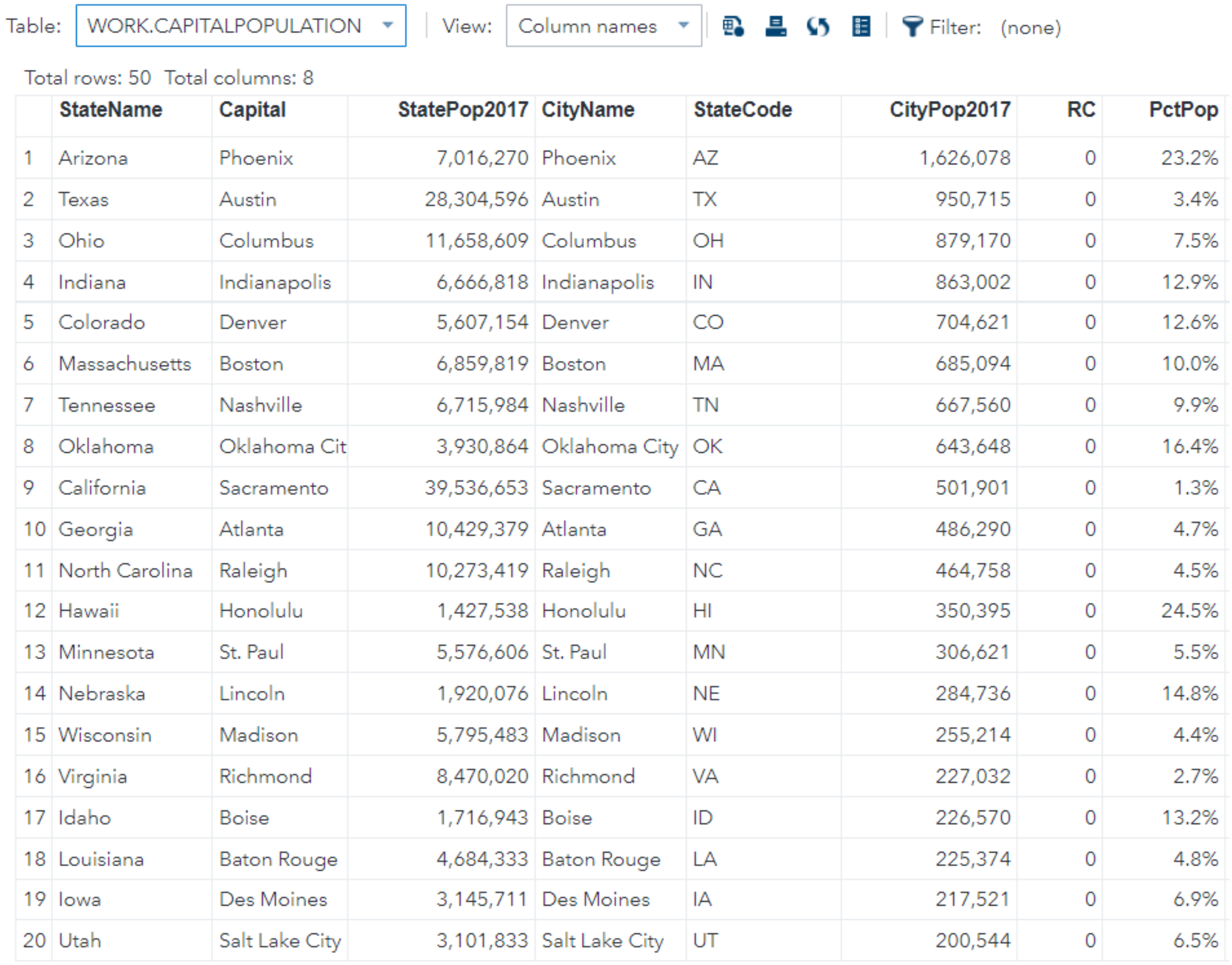


Table: WORK.CAPITALPOPULATION View: Column names Filter: (none)

Total rows: 50 Total columns: 8

	StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop
1	Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	0	23.2%
2	Texas	Austin	28,304,596	Austin	TX	950,715	0	3.4%
3	Ohio	Columbus	11,658,609	Columbus	OH	879,170	0	7.5%
4	Indiana	Indianapolis	6,666,818	Indianapolis	IN	863,002	0	12.9%
5	Colorado	Denver	5,607,154	Denver	CO	704,621	0	12.6%
6	Massachusetts	Boston	6,859,819	Boston	MA	685,094	0	10.0%
7	Tennessee	Nashville	6,715,984	Nashville	TN	667,560	0	9.9%
8	Oklahoma	Oklahoma Cit	3,930,864	Oklahoma City	OK	643,648	0	16.4%
9	California	Sacramento	39,536,653	Sacramento	CA	501,901	0	1.3%
10	Georgia	Atlanta	10,429,379	Atlanta	GA	486,290	0	4.7%
11	North Carolina	Raleigh	10,273,419	Raleigh	NC	464,758	0	4.5%
12	Hawaii	Honolulu	1,427,538	Honolulu	HI	350,395	0	24.5%
13	Minnesota	St. Paul	5,576,606	St. Paul	MN	306,621	0	5.5%
14	Nebraska	Lincoln	1,920,076	Lincoln	NE	284,736	0	14.8%
15	Wisconsin	Madison	5,795,483	Madison	WI	255,214	0	4.4%
16	Virginia	Richmond	8,470,020	Richmond	VA	227,032	0	2.7%
17	Idaho	Boise	1,716,943	Boise	ID	226,570	0	13.2%
18	Louisiana	Baton Rouge	4,684,333	Baton Rouge	LA	225,374	0	4.8%
19	Iowa	Des Moines	3,145,711	Des Moines	IA	217,521	0	6.9%
20	Utah	Salt Lake City	3,101,833	Salt Lake City	UT	200,544	0	6.5%

```
************************************************************;
*  LESSON 4, PRACTICE 1                                    *;
************************************************************;
```

/*Practice Level 1: Performing a Table Lookup Using One Key

If necessary, start SAS Studio before you begin.

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

Create a hash object based on the pg3.np_codelookup table. This table contains national park information.

Read the columns ParkCode, State, and GrossAcres from the pg3.np_acres2 table.

This table contains acreage amounts for the national parks.

Look up the uppercase value of ParkCode in the hash object to retrieve values of ParkName and Type.

Open the p304p01.sas program in the practices folder. Review the DATA step syntax.

```
end;
```

What is the name of the hash object being created?

What table is loading the hash object?

What is the name of the key component?

What are the names of the data components?

:Add an assignment statement to the DATA step to create the column RC, which is equal to the return code from finding the ParkCode value

in the hash object. Run the DATA step. View the log and the output table.

Based on the log:

How many rows from the pg3.np_codelookup table were read into the hash object?

How many rows were read from the pg3.np_acres2 table?

How many rows are in the output table?

Based on the output table,

How many ParkCode values are not found in the hash object (RC not equal to 0)?

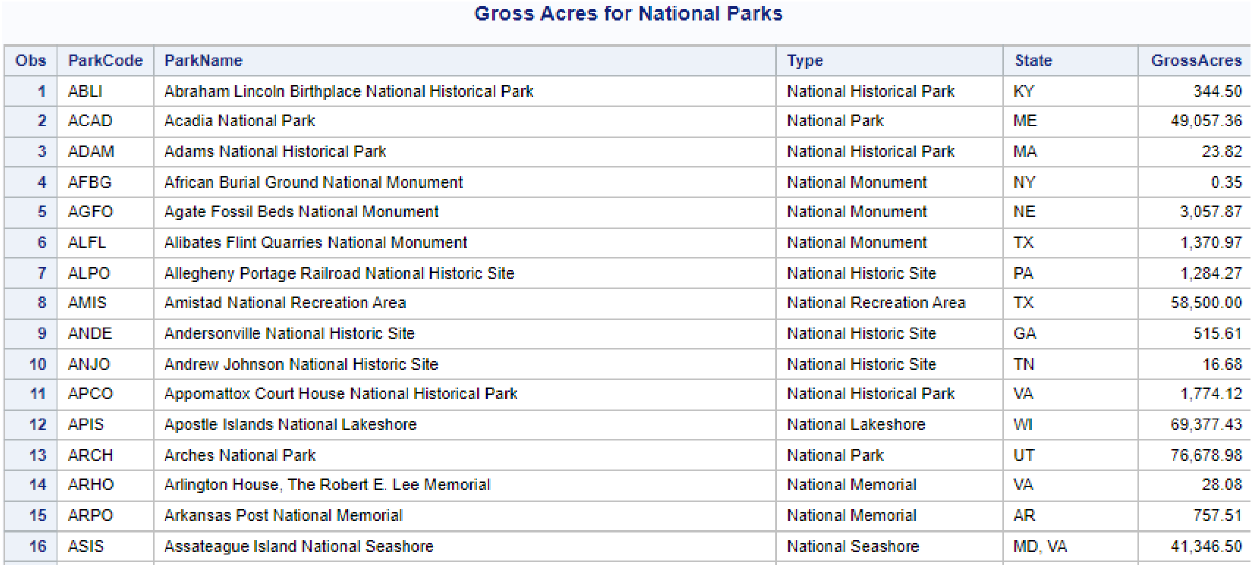
Add a subsetting IF statement to output only the RC values that are equal to 0 (matches).

Add a DROP statement to eliminate the RC column.

Run the program and view the results.

How many data rows are in the results? Note: Type a numeric value.

```
*/
data work.acreage;
   length ParkCode $ 4 ParkName $ 115 Type $ 28;
   if _N_=1 then do;
      declare hash ParkDesc(dataset:'pg3.np_codelookup');
      ParkDesc.definekey('ParkCode');
      ParkDesc.definedata('ParkName','Type');
      ParkDesc.definedone();
      call missing(ParkCode,ParkName,Type);
   end;
   set pg3.np_acres2;
   ParkCode=upcase(ParkCode);
   /* add an assignment statement */
         RC=ParkDesc.find(Key:ParkCode);
   /* add a subsetting IF statment */
         if RC=0 then output work.acreage;
   /* add a DROP statement */
         drop RC;
run;

title 'Gross Acres for National Parks';
proc print data=work.acreage;
run;
title;
```



Gross Acres for National Parks

Obs	ParkCode	ParkName	Type	State	GrossAcres
1	ABLI	Abraham Lincoln Birthplace National Historical Park	National Historical Park	KY	344.50
2	ACAD	Acadia National Park	National Park	ME	49,057.36
3	ADAM	Adams National Historical Park	National Historical Park	MA	23.82
4	AFBG	African Burial Ground National Monument	National Monument	NY	0.35
5	AGFO	Agate Fossil Beds National Monument	National Monument	NE	3,057.87
6	ALFL	Alibates Flint Quarries National Monument	National Monument	TX	1,370.97
7	ALPO	Allegheny Portage Railroad National Historic Site	National Historic Site	PA	1,284.27
8	AMIS	Amistad National Recreation Area	National Recreation Area	TX	58,500.00
9	ANDE	Andersonville National Historic Site	National Historic Site	GA	515.61
10	ANJO	Andrew Johnson National Historic Site	National Historic Site	TN	16.68
11	APCO	Appomattox Court House National Historical Park	National Historical Park	VA	1,774.12
12	APIS	Apostle Islands National Lakeshore	National Lakeshore	WI	69,377.43
13	ARCH	Arches National Park	National Park	UT	76,678.98
14	ARHO	Arlington House, The Robert E. Lee Memorial	National Memorial	VA	28.08
15	ARPO	Arkansas Post National Memorial	National Memorial	AR	757.51
16	ASIS	Assateague Island National Seashore	National Seashore	MD, VA	41,346.50

```
NOTE: There were 713 observations read from the data set
PG3.NP_CODELOOKUP.
 NOTE: There were 368 observations read from the data set
PG3.NP_ACRES2.
 NOTE: The data set WORK.ACREAGE has 366 observations and 5 variables.
```

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

```
*  LESSON 4, PRACTICE 2                         *;
**********************************************************;
```

/*Practice Level 2: Performing a Table Lookup Using Three Keys

If necessary, start SAS Studio before you begin.

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

Create a hash object based on the pg3.storm_range table. This table contains four wind measurements for

each combination of StartYear, Name, and Basin. Read the columns from the pg3.storm_summary_cat345 table.

This table contains information such as MaxWindMPH and MinPressure for each combination of StartDate, Name, and Basin

for category 3, 4, and 5 storms. Look up the appropriate values in the hash object to retrieve the four wind measurements.

1) Open the p304p02.sas program in the practices folder. Add statements to the DATA step to create a hash object named Storm.

Add a DECLARE statement to load the hash object Storm with the table pg3.storm_range.

Use the DEFINEKEY method to specify the key components of StartYear, Name, and Basin.

Use the DEFINEDATA method to specify the data components of Wind1, Wind2, Wind3, and Wind4.

Use the DEFINEDONE method to complete the hash object.

2) Add an assignment statement to create the column ReturnCode, which is equal to the return code from finding the key values

in the hash object. You will need to use the YEAR function on StartDate for the first key value followed by the key values

of Name and Basin. Note: Key values must be specified in the FIND method in the same order as specified with the DEFINEKEY method.

3) Run the DATA step. View the log and the results.

How many rows from the pg3. storm_range table were read into the hash object?

How many rows were read from the pg3. storm_summary_cat345 table?

How many rows are in the output table?

How many key values are not found in the hash object (ReturnCode not equal to 0)?

Why is the StartYear column set to missing values?

Modify the assignment statement to be a subsetting IF statement to output only the FIND values that are equal to 0 (matches).

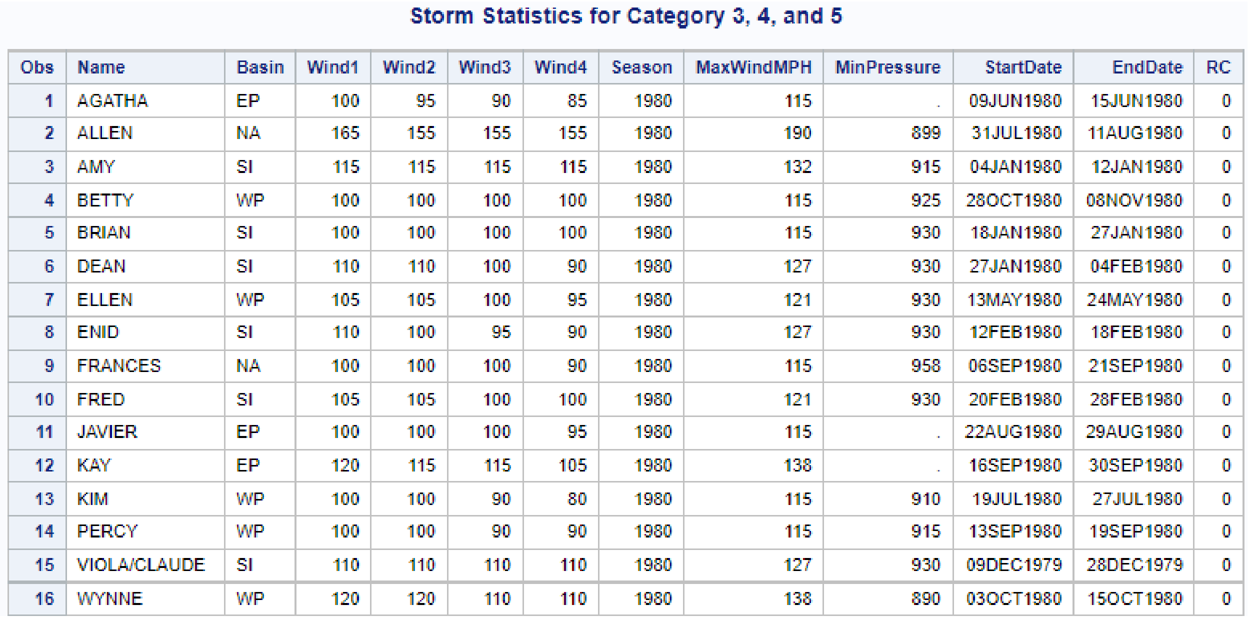
Drop the StartYear column.

Run the program and verify the results.

How many data rows are in the results? Note: Type a numeric value.

```
*/

data work.storm_cat345_facts;

  if _N_=1 then do;

    if 0 then set pg3.storm_range;

    /* add a DECLARE statement */

                declare hash Storm(dataset:'pg3.storm_range');

    /* use the DEFINEKEY, DEFINEDATA, and DEFINEDONE methods */

                Storm.definekey('StartYear','Name','Basin');

                Storm.definedata('Wind1','Wind2','Wind3','Wind4');

                Storm.definedone();

                call missing(Wind1,Wind2,Wind3,Wind4);

  end;

  set pg3.storm_summary_cat345;

  /* add an assignment statement and later

    modify to a subsetting IF statement */

        RC=Storm.find(Key:Year(StartDate),Key:Name,Key:Basin);

        if RC=0 then output work.storm_cat345_facts;

  /* add a DROP statement */

        drop StartYear;

run;
```

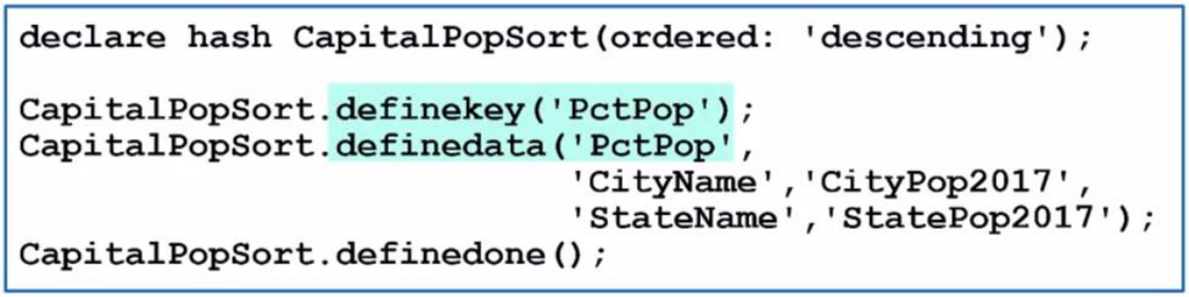
```
title 'Storm Statistics for Category 3, 4, and 5';

proc print data=work.storm_cat345_facts;

run;

title;
```



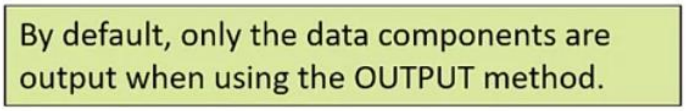
Storm Statistics for Category 3, 4, and 5

Obs	Name	Basin	Wind1	Wind2	Wind3	Wind4	Season	MaxWindMPH	MinPressure	StartDate	EndDate	RC
1	AGATHA	EP	100	95	90	85	1980	115	.	09JUN1980	15JUN1980	0
2	ALLEN	NA	165	155	155	155	1980	190	899	31JUL1980	11AUG1980	0
3	AMY	SI	115	115	115	115	1980	132	915	04JAN1980	12JAN1980	0
4	BETTY	WP	100	100	100	100	1980	115	925	28OCT1980	08NOV1980	0
5	BRIAN	SI	100	100	100	100	1980	115	930	18JAN1980	27JAN1980	0
6	DEAN	SI	110	110	100	90	1980	127	930	27JAN1980	04FEB1980	0
7	ELLEN	WP	105	105	100	95	1980	121	930	13MAY1980	24MAY1980	0
8	ENID	SI	110	100	95	90	1980	127	930	12FEB1980	18FEB1980	0
9	FRANCES	NA	100	100	100	90	1980	115	958	06SEP1980	21SEP1980	0
10	FRED	SI	105	105	100	100	1980	121	930	20FEB1980	28FEB1980	0
11	JAVIER	EP	100	100	100	95	1980	115	.	22AUG1980	29AUG1980	0
12	KAY	EP	120	115	115	105	1980	138	.	16SEP1980	30SEP1980	0
13	KIM	WP	100	100	90	80	1980	115	910	19JUL1980	27JUL1980	0
14	PERCY	WP	100	100	90	90	1980	115	915	13SEP1980	19SEP1980	0
15	VIOLA/CLAUDE	SI	110	110	110	110	1980	127	930	09DEC1979	28DEC1979	0
16	WYNNE	WP	120	120	110	110	1980	138	890	03OCT1980	15OCT1980	0

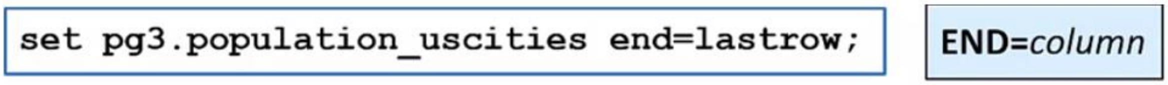
```
NOTE: There were 2959 observations read from the data set
PG3.STORM_RANGE.
 NOTE: There were 570 observations read from the data set
PG3.STORM_SUMMARY_CAT345.
 NOTE: The data set WORK.STORM_CAT345_FACTS has 568 observations and
12 variables.
```



```
declare hash CapitalPopSort(ordered: 'descending');

CapitalPopSort.definekey('PctPop');
CapitalPopSort.definedata('PctPop',
                          'CityName','CityPop2017',
                          'StateName','StatePop2017');
CapitalPopSort.definedone();
```

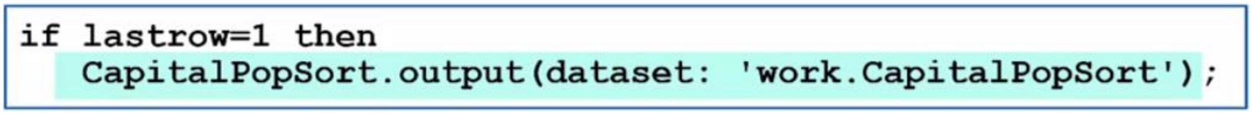


By default, only the data components are output when using the OUTPUT method.



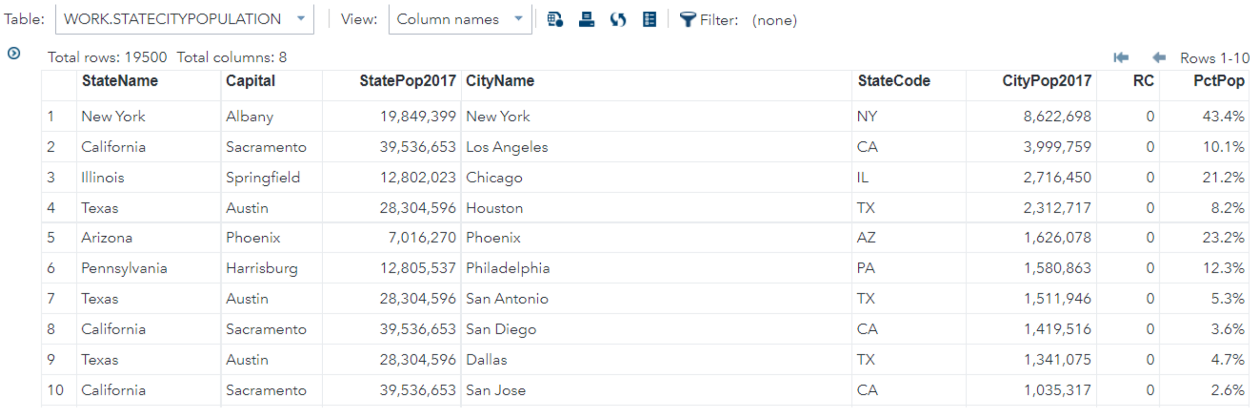
```
set pg3.population_uscities end=lastrow;
```

END=*column*



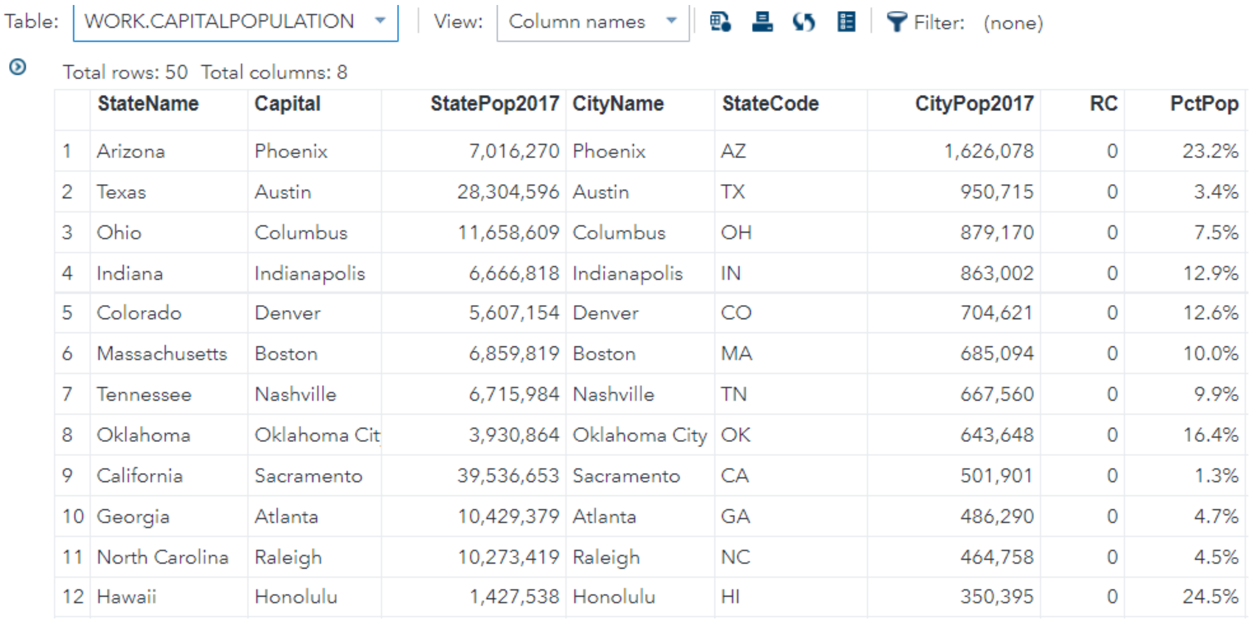
```
if lastrow=1 then
   CapitalPopSort.output(dataset: 'work.CapitalPopSort');
```

```
************************************************************************;
*  Creating a Table with the ADD and OUTPUT Methods          *;
************************************************************************;


************************************************************************;
*  Demo                                        *;
*  1) Run the program. Verify that work.StateCityPopulation    *;
*    contains 19,500 rows and work.CapitalPopulation contains *;
*    50 rows.                                  *;
*  2) Delete work.CapitalPopulation from the DATA statement.   *;
*  3) Remove the /* and */ from around the statements that are *;
*    creating the hash object CapitalPopSort.              *;
*      declare hash CapitalPopSort(ordered: 'descending');    *;
*      CapitalPopSort.definekey('PctPop');                *;
*      CapitalPopSort.definedata('PctPop',                *;
*                         'CityName','CityPop2017',    *;
*                          'StateName','StatePop2017'); *;
*     CapitalPopSort.definedone();                      *;
*  4) Add the END= option to the SET statement for the cities  *;
*    table.                                    *;
*     set pg3.population_uscities end=lastrow;              *;
*  5) Modify the conditional statement to add data to the      *;
*    CapitalPopSort hash object instead of writing the output *;
*    to a table.                                  *;
*     if Capital=CityName then CapitalPopSort.add();        *;
*  6) Add the following statement to output the CapitalPopSort *;
*    hash object to a table.                           *;
*     if lastrow=1 then                               *;
*     CapitalPopSort.output(dataset: 'work.CapitalPopSort'); *;
*  7) Run the program. Verify that work.StateCityPopulation    *;
*    contains 19,500 rows and work.CapitalPopSort contains *;
*    50 rows sorted by descending PctPop. Are there any       *;
*    duplicate values of PctPop in work.CapitalPopSort?       *;
*  8) It appears that there are duplicate values of PctPop in  *;
*    work.CapitalPopSort. For example, rows 23 and 24 show a  *;
*    PctPop value of 4.3% and rows 25 and 26 show a PctPop    *;
*    value of 4.1%.                                  *;
*  9) Comment out the format for PctPop in the FORMAT         *;
*    statement. Run the program. Verify that there are no     *;
*    duplicate values for PctPop in work.CapitalPopSort.      *;
*     format StatePop2017 comma14. /* PctPop percent8.1 */;  *;
* 10) Modify the calculation of PctPop to round the value,     *;
*    which will now produce duplicate values. Run the         *;
*    program. Notice the error messages in the SAS log,       *;
*    ERROR: Duplicate key. Nine error messages exist due to   *;
*    the duplicate values for 4.3, 4.1, 3.4, 2.6, 1.2, 0.9    *;
*    (2), 0.7, and 0.6.                                       *;
*      PctPop=round(CityPop2017/StatePop2017*100,.1);         *;
**************************************************************;

data work.StateCityPopulation work.CapitalPopulation;
   if _N_=1 then do;
      if 0 then set pg3.population_usstates;
      declare hash States(dataset: 'pg3.population_usstates');
      States.definekey('StateName');
      States.definedata('Capital','StatePop2017');
      States.definedone();
              /*
      declare hash CapitalPopSort(ordered: 'descending');
      CapitalPopSort.definekey('PctPop');
      CapitalPopSort.definedata('PctPop',
                         'CityName','CityPop2017',
                         'StateName','StatePop2017');
      CapitalPopSort.definedone();
              */
   end;
   set pg3.population_uscities;
   StateName=stnamel(StateCode);
   RC=States.find(key:StateName);
   if RC ne 0 then call missing(Capital, StatePop2017);
   PctPop=CityPop2017/StatePop2017;
   output work.StateCityPopulation;
   if Capital=CityName then output work.CapitalPopulation;
   format StatePop2017 comma14. PctPop percent8.1;
run;
```



Total rows: 19500 Total columns: 8

	StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop
1	New York	Albany	19,849,399	New York	NY	8,622,698	0	43.4%
2	California	Sacramento	39,536,653	Los Angeles	CA	3,999,759	0	10.1%
3	Illinois	Springfield	12,802,023	Chicago	IL	2,716,450	0	21.2%
4	Texas	Austin	28,304,596	Houston	TX	2,312,717	0	8.2%
5	Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	0	23.2%
6	Pennsylvania	Harrisburg	12,805,537	Philadelphia	PA	1,580,863	0	12.3%
7	Texas	Austin	28,304,596	San Antonio	TX	1,511,946	0	5.3%
8	California	Sacramento	39,536,653	San Diego	CA	1,419,516	0	3.6%
9	Texas	Austin	28,304,596	Dallas	TX	1,341,075	0	4.7%
10	California	Sacramento	39,536,653	San Jose	CA	1,035,317	0	2.6%



Total rows: 50 Total columns: 8

	StateName	Capital	StatePop2017	CityName	StateCode	CityPop2017	RC	PctPop
1	Arizona	Phoenix	7,016,270	Phoenix	AZ	1,626,078	0	23.2%
2	Texas	Austin	28,304,596	Austin	TX	950,715	0	3.4%
3	Ohio	Columbus	11,658,609	Columbus	OH	879,170	0	7.5%
4	Indiana	Indianapolis	6,666,818	Indianapolis	IN	863,002	0	12.9%
5	Colorado	Denver	5,607,154	Denver	CO	704,621	0	12.6%
6	Massachusetts	Boston	6,859,819	Boston	MA	685,094	0	10.0%
7	Tennessee	Nashville	6,715,984	Nashville	TN	667,560	0	9.9%
8	Oklahoma	Oklahoma Cit	3,930,864	Oklahoma City	OK	643,648	0	16.4%
9	California	Sacramento	39,536,653	Sacramento	CA	501,901	0	1.3%
10	Georgia	Atlanta	10,429,379	Atlanta	GA	486,290	0	4.7%
11	North Carolina	Raleigh	10,273,419	Raleigh	NC	464,758	0	4.5%
12	Hawaii	Honolulu	1,427,538	Honolulu	HI	350,395	0	24.5%

```
data work.StateCityPopulation;
   if _N_=1 then do;
      if 0 then set pg3.population_usstates;
      declare hash States(dataset: 'pg3.population_usstates');
      States.definekey('StateName');
      States.definedata('Capital','StatePop2017');
      States.definedone();

      declare hash CapitalPopSort(ordered: 'descending');
```

```
    CapitalPopSort.definekey('PctPop');
    CapitalPopSort.definedata('PctPop',
                  'CityName','CityPop2017',
                  'StateName','StatePop2017');
    CapitalPopSort.definedone();

  end;
  set pg3.population_uscities end=lastrow;
  StateName=stnamel(StateCode);
  RC=States.find(key:StateName);
  if RC ne 0 then call missing(Capital, StatePop2017);
  PctPop=CityPop2017/StatePop2017;
  output work.StateCityPopulation;
  if Capital=CityName then CapitalPopSort.add();
        if lastrow=1 then
                CapitalPopSort.output(dataset: 'work.CapitalPopSort');
  format StatePop2017 comma14. /*PctPop percent8.1*/;
run;
```

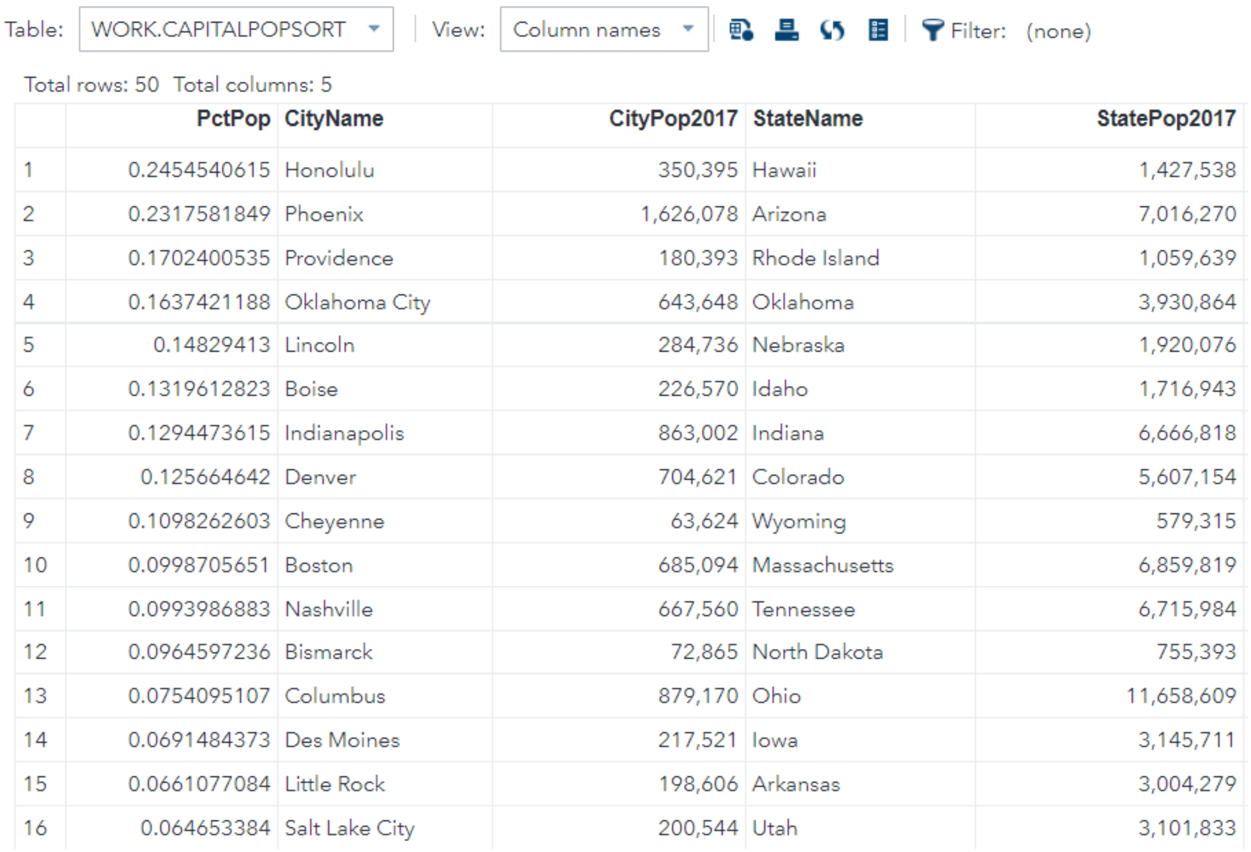
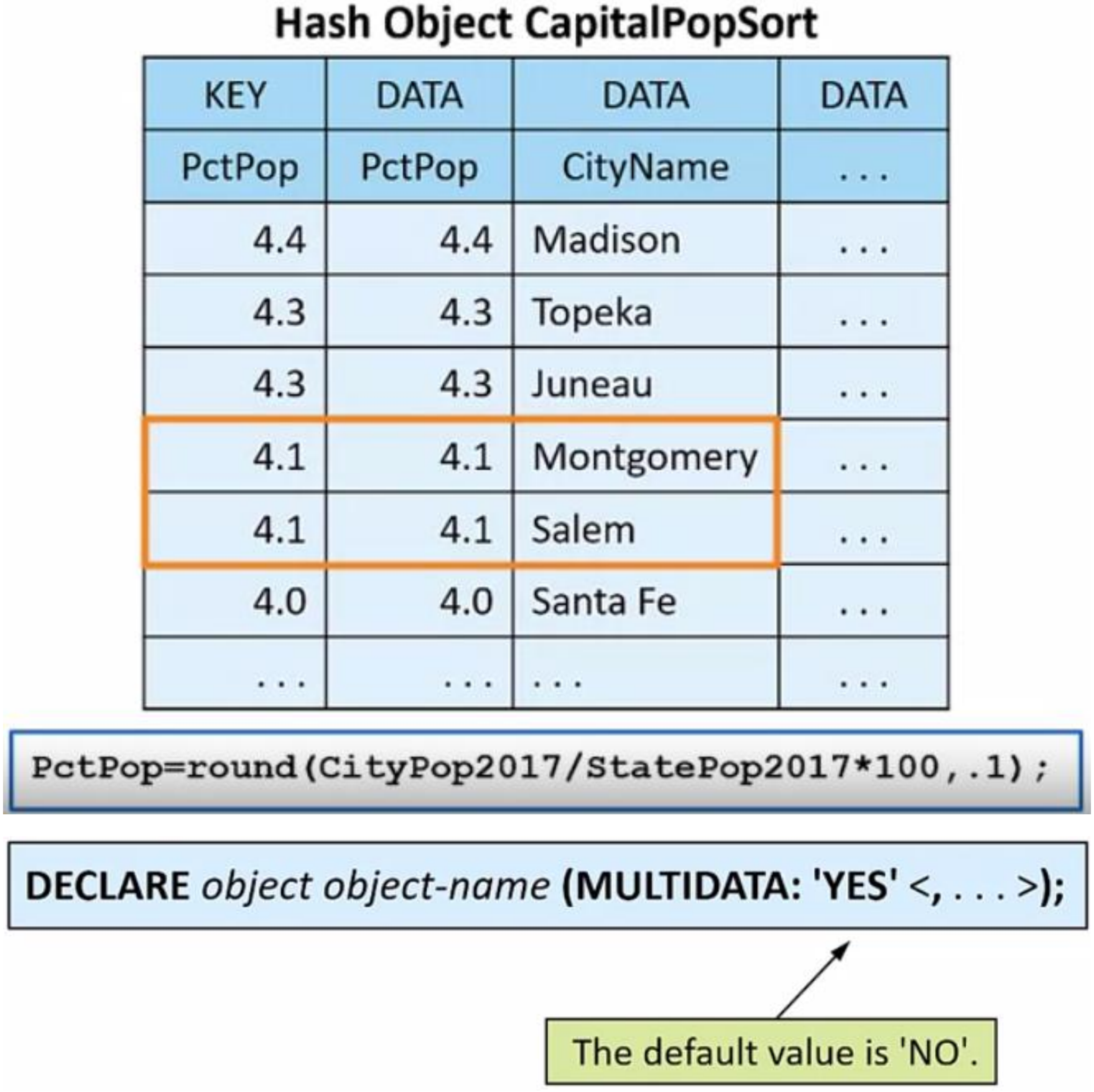


Table: WORK.CAPITALPOPSORT View: Column names Filter: (none)

Total rows: 50 Total columns: 5

	PctPop	CityName	CityPop2017	StateName	StatePop2017
1	0.2454540615	Honolulu	350,395	Hawaii	1,427,538
2	0.2317581849	Phoenix	1,626,078	Arizona	7,016,270
3	0.1702400535	Providence	180,393	Rhode Island	1,059,639
4	0.1637421188	Oklahoma City	643,648	Oklahoma	3,930,864
5	0.14829413	Lincoln	284,736	Nebraska	1,920,076
6	0.1319612823	Boise	226,570	Idaho	1,716,943
7	0.1294473615	Indianapolis	863,002	Indiana	6,666,818
8	0.125664642	Denver	704,621	Colorado	5,607,154
9	0.1098262603	Cheyenne	63,624	Wyoming	579,315
10	0.0998705651	Boston	685,094	Massachusetts	6,859,819
11	0.0993986883	Nashville	667,560	Tennessee	6,715,984
12	0.0964597236	Bismarck	72,865	North Dakota	755,393
13	0.0754095107	Columbus	879,170	Ohio	11,658,609
14	0.0691484373	Des Moines	217,521	Iowa	3,145,711
15	0.0661077084	Little Rock	198,606	Arkansas	3,004,279
16	0.064653384	Salt Lake City	200,544	Utah	3,101,833



Hash Object CapitalPopSort

KEY	DATA	DATA	DATA
PctPop	PctPop	CityName	. . .
4.4	4.4	Madison	. . .
4.3	4.3	Topeka	. . .
4.3	4.3	Juneau	. . .
4.1	4.1	Montgomery	. . .
4.1	4.1	Salem	. . .
4.0	4.0	Santa Fe	. . .
. . .	. . .	. . .	. . .

```
PctPop=round(CityPop2017/StatePop2017*100,.1);
```

DECLARE *object object-name* **(MULTIDATA: 'YES'** <**,** . . . >**);**

The default value is 'NO'.

```
*******************************************************;
*  Activity 4.03                                  *;
*  1) Run the program and view the duplicate key errors   *;
*     in the SAS log.                             *;
*  2) Add the MULTIDATA: 'YES' argument to the DECLARE    *;
*     statement for the CapitalPopSort hash object.      *;
*     Arguments are separated with a comma.             *;
*  3) Run the program. View the work.CapitalPopSort      *;
*    table. Do you see duplicate PctPop values?          *;
******************************************************;


data work.StateCityPopulation;
   if _N_=1 then do;
      if 0 then set pg3.population_usstates;
      declare hash States(dataset: 'pg3.population_usstates');
      States.definekey('StateName');
      States.definedata('Capital','StatePop2017');
      States.definedone();
      declare hash CapitalPopSort(Multidata: 'Yes', ordered: 'descending');
      CapitalPopSort.definekey('PctPop');
      CapitalPopSort.definedata('PctPop',
                               'CityName','CityPop2017',
                               'StateName','StatePop2017');
      CapitalPopSort.definedone();
   end;
   set pg3.population_uscities end=lastrow;
   StateName=stnamel(StateCode);
   RC=States.find(key:StateName);
   if RC ne 0 then call missing(Capital, StatePop2017);
   PctPop=round(CityPop2017/StatePop2017*100,.1);
   output work.StateCityPopulation;
   if Capital=CityName then CapitalPopSort.add();
   if lastrow=1 then
      CapitalPopSort.output(dataset: 'work.CapitalPopSort');
   format StatePop2017 comma14.;
run; ******************************************************;
```

```
*  Activity 4.03                              *;
*  1) Run the program and view the duplicate key errors   *;
*     in the SAS log.                              *;
*  2) Add the MULTIDATA: 'YES' argument to the DECLARE    *;
*     statement for the CapitalPopSort hash object.       *;
*     Arguments are separated with a comma.             *;
*  3) Run the program. View the work.CapitalPopSort       *;
*     table. Do you see duplicate PctPop values?          *;
********************************************************;


data work.StateCityPopulation;
   if _N_=1 then do;
     if 0 then set pg3.population_usstates;
     declare hash States(dataset: 'pg3.population_usstates');
     States.definekey('StateName');
     States.definedata('Capital','StatePop2017');
     States.definedone();
     declare hash CapitalPopSort(Multidata: 'Yes', ordered: 'descending');
     CapitalPopSort.definekey('PctPop');
     CapitalPopSort.definedata('PctPop',
                        'CityName','CityPop2017',
                        'StateName','StatePop2017');
     CapitalPopSort.definedone();
   end;
   set pg3.population_uscities end=lastrow;
   StateName=stnamel(StateCode);
   RC=States.find(key:StateName);
   if RC ne 0 then call missing(Capital, StatePop2017);
   PctPop=round(CityPop2017/StatePop2017*100,.1);
   output work.StateCityPopulation;
   if Capital=CityName then CapitalPopSort.add();
   if lastrow=1 then
     CapitalPopSort.output(dataset: 'work.CapitalPopSort');
   format StatePop2017 comma14.;
run;
```

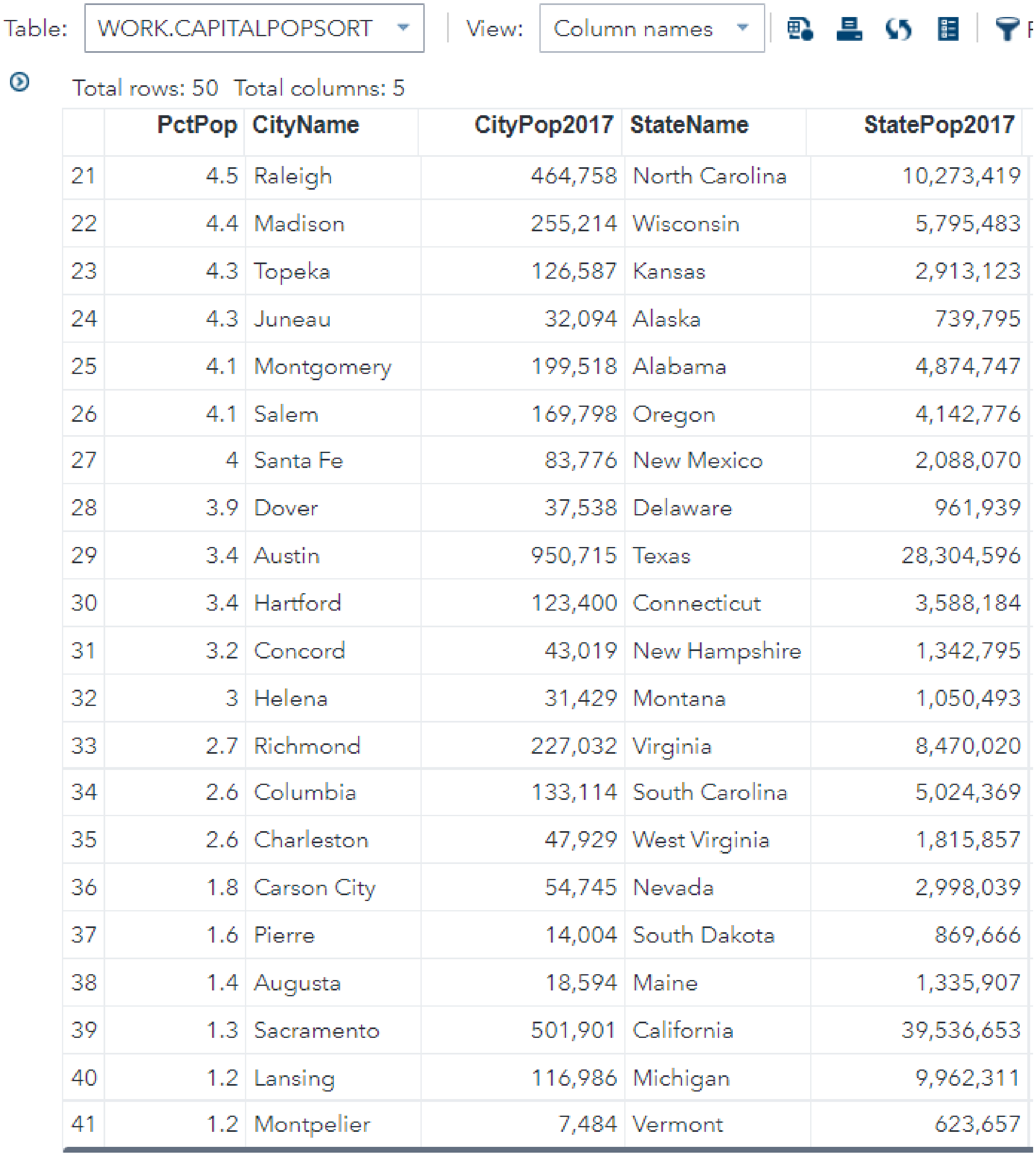


Table: WORK.CAPITALPOPSORT   View: Column names

Total rows: 50 Total columns: 5

	PctPop	CityName	CityPop2017	StateName	StatePop2017
21	4.5	Raleigh	464,758	North Carolina	10,273,419
22	4.4	Madison	255,214	Wisconsin	5,795,483
23	4.3	Topeka	126,587	Kansas	2,913,123
24	4.3	Juneau	32,094	Alaska	739,795
25	4.1	Montgomery	199,518	Alabama	4,874,747
26	4.1	Salem	169,798	Oregon	4,142,776
27	4	Santa Fe	83,776	New Mexico	2,088,070
28	3.9	Dover	37,538	Delaware	961,939
29	3.4	Austin	950,715	Texas	28,304,596
30	3.4	Hartford	123,400	Connecticut	3,588,184
31	3.2	Concord	43,019	New Hampshire	1,342,795
32	3	Helena	31,429	Montana	1,050,493
33	2.7	Richmond	227,032	Virginia	8,470,020
34	2.6	Columbia	133,114	South Carolina	5,024,369
35	2.6	Charleston	47,929	West Virginia	1,815,857
36	1.8	Carson City	54,745	Nevada	2,998,039
37	1.6	Pierre	14,004	South Dakota	869,666
38	1.4	Augusta	18,594	Maine	1,335,907
39	1.3	Sacramento	501,901	California	39,536,653
40	1.2	Lansing	116,986	Michigan	9,962,311
41	1.2	Montpelier	7,484	Vermont	623,657

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

/*Practice Level 1: Creating a Sorted Table from a Hash Object

If necessary, start SAS Studio before you begin.

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

This practice involves looking up the values of ParkCode in a hash object to retrieve values of ParkName and Type.

The output table work.acreage contains national park information in the default sorted order of ParkCode.

Modify the starter program to create an additional output table, work.acreage_sort, which contains the same data

as work.acreage but in sorted order by descending GrossAcres.

Open the p304p04.sas program in the practices folder. This program contains the solution to practice 1.

Review the DATA step syntax. Run the program and view the results.

Declare and define a hash object named Acreage. Specify an ORDERED argument of DESCENDING and a MULTIDATA argument of YES.

Define a key component of GrossAcres and data components of ParkCode, ParkName, Type, State, and GrossAcres.

Change the subsetting IF statement for RC to be a conditional IF-THEN statement.

If RC is equal to 0, then add the data to the Acreage hash object.

Add an IF-THEN statement before the DROP statement to output the hash object Acreage to a table named work.acreage_sort

if the last row has been read from the input table.

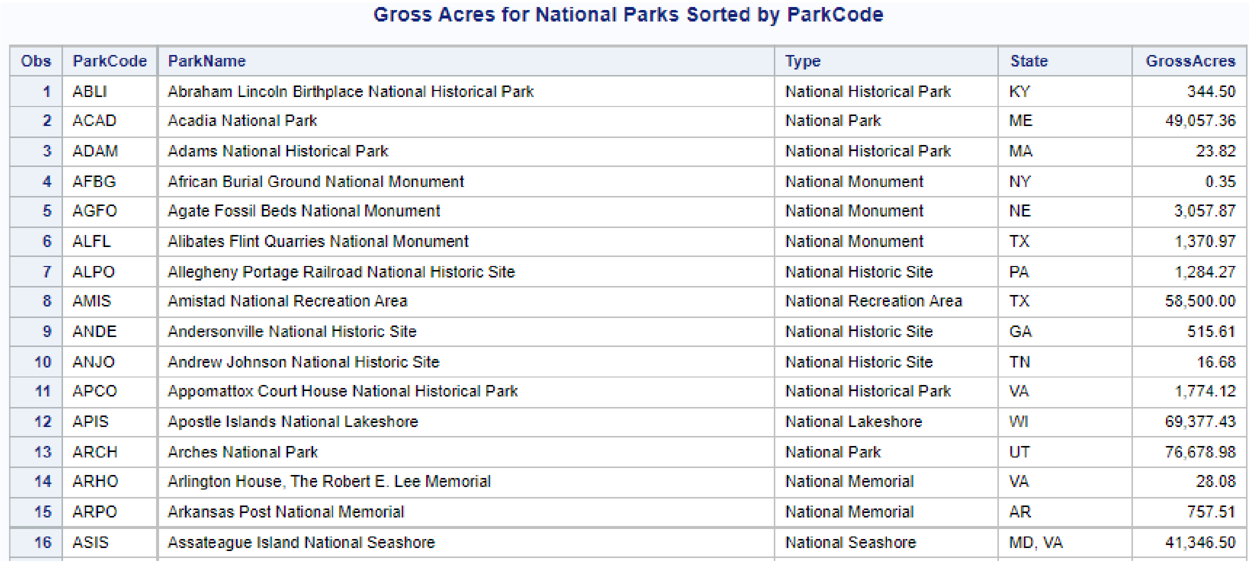
Add an END= option to the SET statement to create a column named Last.

Add a PROC PRINT step for the table work.acreage_sort.

What is the smallest value of GrossAcres? Note: Type the value as it appears in the results.

```
*/

data work.acreage;

   length ParkCode $ 4 ParkName $ 115 Type $ 28;

   if _N_=1 then do;

      declare hash ParkDesc(dataset:'pg3.np_codelookup');

      ParkDesc.definekey('ParkCode');

      ParkDesc.definedata('ParkName','Type');

      ParkDesc.definedone();
```

```
     call missing(ParkCode,ParkName,Type);
     /* declare and define a hash object */


   end;
   set pg3.np_acres2 end=lastrow;
   ParkCode=upcase(ParkCode);
   RC=ParkDesc.find(key:ParkCode);
   /* change the subsetting IF statement to be IF/THEN statement */
   if RC=0;
   /* add an IF/THEN statement */


   drop RC;
run;


title 'Gross Acres for National Parks Sorted by ParkCode';
proc print data=work.acreage;
run;
title;
```

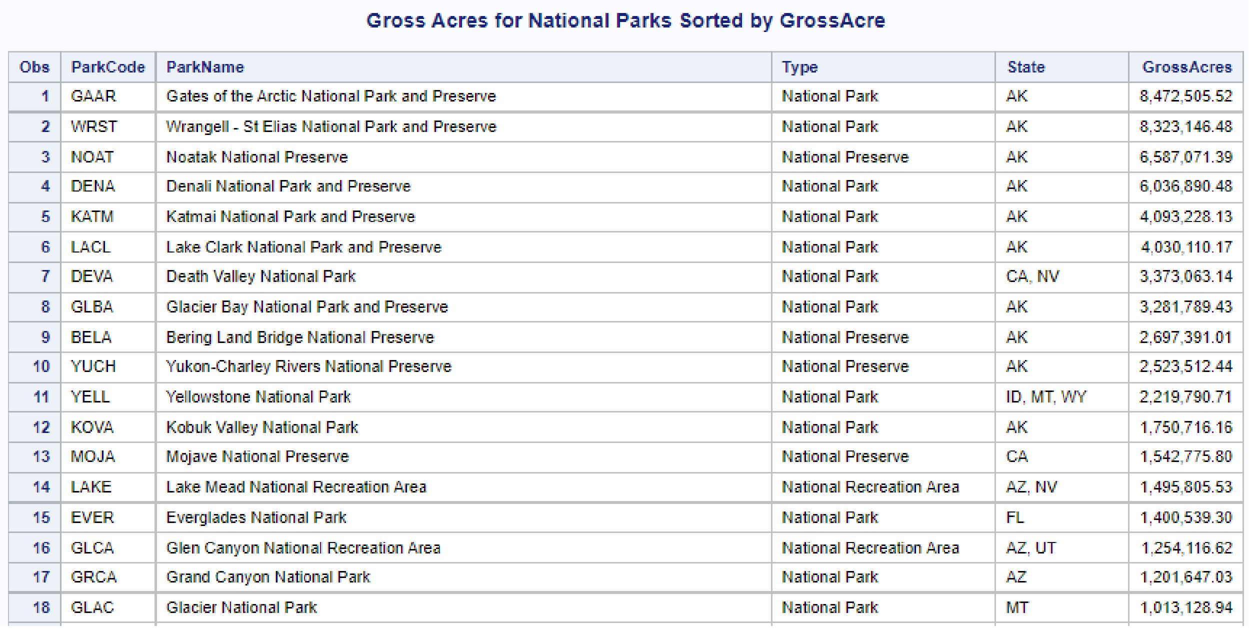


Gross Acres for National Parks Sorted by ParkCode

Obs	ParkCode	ParkName	Type	State	GrossAcres
1	ABLI	Abraham Lincoln Birthplace National Historical Park	National Historical Park	KY	344.50
2	ACAD	Acadia National Park	National Park	ME	49,057.36
3	ADAM	Adams National Historical Park	National Historical Park	MA	23.82
4	AFBG	African Burial Ground National Monument	National Monument	NY	0.35
5	AGFO	Agate Fossil Beds National Monument	National Monument	NE	3,057.87
6	ALFL	Alibates Flint Quarries National Monument	National Monument	TX	1,370.97
7	ALPO	Allegheny Portage Railroad National Historic Site	National Historic Site	PA	1,284.27
8	AMIS	Amistad National Recreation Area	National Recreation Area	TX	58,500.00
9	ANDE	Andersonville National Historic Site	National Historic Site	GA	515.61
10	ANJO	Andrew Johnson National Historic Site	National Historic Site	TN	16.68
11	APCO	Appomattox Court House National Historical Park	National Historical Park	VA	1,774.12
12	APIS	Apostle Islands National Lakeshore	National Lakeshore	WI	69,377.43
13	ARCH	Arches National Park	National Park	UT	76,678.98
14	ARHO	Arlington House, The Robert E. Lee Memorial	National Memorial	VA	28.08
15	ARPO	Arkansas Post National Memorial	National Memorial	AR	757.51
16	ASIS	Assateague Island National Seashore	National Seashore	MD, VA	41,346.50

```
data work.acreage;

  length ParkCode $ 4 ParkName $ 115 Type $ 28;

  if _N_=1 then do;

    declare hash ParkDesc(dataset:'pg3.np_codelookup');

    ParkDesc.definekey('ParkCode');

    ParkDesc.definedata('ParkName','Type');

    ParkDesc.definedone();

    call missing(ParkCode,ParkName,Type);

    /* declare and define a hash object */

                declare hash Acreage(Multidata: 'Yes', Ordered: 'descending');

                Acreage.definekey('GrossAcres');

                Acreage.definedata('ParkCode','ParkName','Type','State','GrossAcres');

                Acreage.definedone();

  end;

  set pg3.np_acres2 end=lastrow;

  ParkCode=upcase(ParkCode);

  RC=ParkDesc.find(key:ParkCode);

  /* change the subsetting IF statement to be IF/THEN statement */

  if RC=0 then Acreage.add();

  /* add an IF/THEN statement */

        if lastrow=1 then

          Acreage.output(dataset: 'work.acreage_sort');

  drop RC;

run;


title 'Gross Acres for National Parks Sorted by ParkCode';

proc print data=work.acreage;

run;

title;
```

```
title 'Gross Acres for National Parks Sorted by GrossAcre';

proc print data=work.acreage_sort;

run;

title;
```



Gross Acres for National Parks Sorted by GrossAcre

Obs	ParkCode	ParkName	Type	State	GrossAcres
1	GAAR	Gates of the Arctic National Park and Preserve	National Park	AK	8,472,505.52
2	WRST	Wrangell - St Elias National Park and Preserve	National Park	AK	8,323,146.48
3	NOAT	Noatak National Preserve	National Preserve	AK	6,587,071.39
4	DENA	Denali National Park and Preserve	National Park	AK	6,036,890.48
5	KATM	Katmai National Park and Preserve	National Park	AK	4,093,228.13
6	LACL	Lake Clark National Park and Preserve	National Park	AK	4,030,110.17
7	DEVA	Death Valley National Park	National Park	CA, NV	3,373,063.14
8	GLBA	Glacier Bay National Park and Preserve	National Park	AK	3,281,789.43
9	BELA	Bering Land Bridge National Preserve	National Preserve	AK	2,697,391.01
10	YUCH	Yukon-Charley Rivers National Preserve	National Preserve	AK	2,523,512.44
11	YELL	Yellowstone National Park	National Park	ID, MT, WY	2,219,790.71
12	KOVA	Kobuk Valley National Park	National Park	AK	1,750,716.16
13	MOJA	Mojave National Preserve	National Preserve	CA	1,542,775.80
14	LAKE	Lake Mead National Recreation Area	National Recreation Area	AZ, NV	1,495,805.53
15	EVER	Everglades National Park	National Park	FL	1,400,539.30
16	GLCA	Glen Canyon National Recreation Area	National Recreation Area	AZ, UT	1,254,116.62
17	GRCA	Grand Canyon National Park	National Park	AZ	1,201,647.03
18	GLAC	Glacier National Park	National Park	MT	1,013,128.94

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

*  LESSON 4, PRACTICE 5                                 *;

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

/*Practice Level 2: Eliminating a PROC SORT step by Creating a Sorted Table from a Hash Object

If necessary, start SAS Studio before you begin.

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

This practice involves looking up the values of StartYear, Name, and Basin to retrieve four wind measurement values.

The final output table, work.storm_cat345_facts, contains storm information in default sorted order of Season and Name.

Modify the starter program to create an additional output table, work.cat345_sort, that contains the data in sorted order

by descending MaxWindMPH, Season, and Name.

Open the p304p05.sas program in the practices folder.

Run the DATA step and verify that the output table is sorted by ascending Season and Name.

Run the PROC SORT and PROC PRINT steps. Verify that the results are sorted by descending MaxWindMPH, Season, and Name.

In the DATA step, declare and define a hash object named StormSort. Specify an ORDERED argument of DESCENDING and

a MULTIDATA argument of YES.

Define key components for the StormSort hash object based on the columns specified in the BY statement of the PROC SORT step.

Define data components for the StormSort hash object based on the columns specified in the KEEP= option

in the PROC SORT statement. Change the subsetting IF statement to be a conditional IF-THEN statement.

If the FIND method is equal to 0, then add the data to the StormSort hash object.

Add an END= option to the SET statement to create a column named Last.

Add an IF-THEN statement before the KEEP statement to output the hash object StormSort to a table named work.cat345_sort

if the last row has been read from the input table.

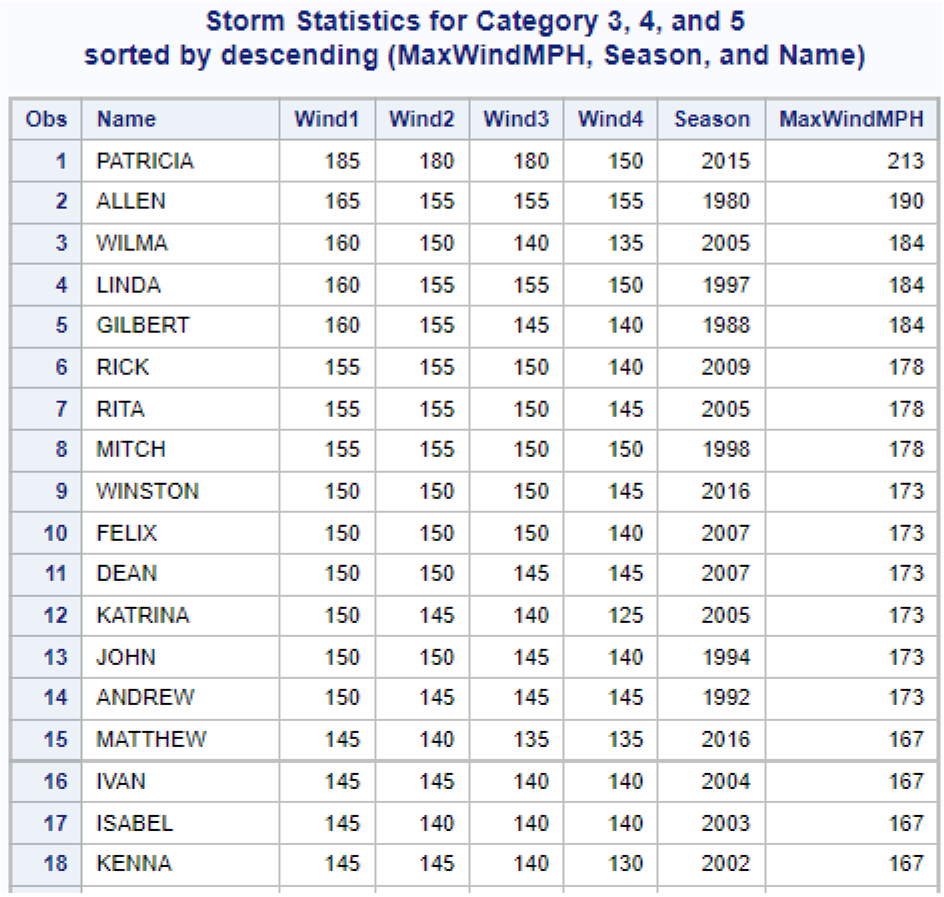
Delete the PROC SORT step.

Run the program and view the PROC PRINT results.

How many storms have a maximum wind speed of 173 mph? Note: Type a numeric value.

```
*/

data work.storm_cat345_facts;

   if _N_=1 then do;

      if 0 then set pg3.storm_range;

      declare hash Storm(dataset:'pg3.storm_range');

      Storm.definekey('StartYear','Name','Basin');

      Storm.definedata('Wind1','Wind2','Wind3','Wind4');

      Storm.definedone();
```

```
    end;
    set pg3.storm_summary_cat345;
    if Storm.find(key:year(StartDate),key:Name,key:Basin)=0;
    keep Name Basin Wind1-Wind4 Season MaxWindMPH StartDate;
run;

proc sort data=work.storm_cat345_facts
          out=work.cat345_sort
             (keep=Season Name Wind1-Wind4 MaxWindMPH);
    by descending MaxWindMPH descending Season descending Name;
run;

title1 'Storm Statistics for Category 3, 4, and 5';
title2 'sorted by descending (MaxWindMPH, Season, and Name)';
proc print data=work.cat345_sort;
run;
title;
```



Storm Statistics for Category 3, 4, and 5
sorted by descending (MaxWindMPH, Season, and Name)

Obs	Name	Wind1	Wind2	Wind3	Wind4	Season	MaxWindMPH
1	PATRICIA	185	180	180	150	2015	213
2	ALLEN	165	155	155	155	1980	190
3	WILMA	160	150	140	135	2005	184
4	LINDA	160	155	155	150	1997	184
5	GILBERT	160	155	145	140	1988	184
6	RICK	155	155	150	140	2009	178
7	RITA	155	155	150	145	2005	178
8	MITCH	155	155	150	150	1998	178
9	WINSTON	150	150	150	145	2016	173
10	FELIX	150	150	150	140	2007	173
11	DEAN	150	150	145	145	2007	173
12	KATRINA	150	145	140	125	2005	173
13	JOHN	150	150	145	140	1994	173
14	ANDREW	150	145	145	145	1992	173
15	MATTHEW	145	140	135	135	2016	167
16	IVAN	145	145	140	140	2004	167
17	ISABEL	145	140	140	140	2003	167
18	KENNA	145	145	140	130	2002	167

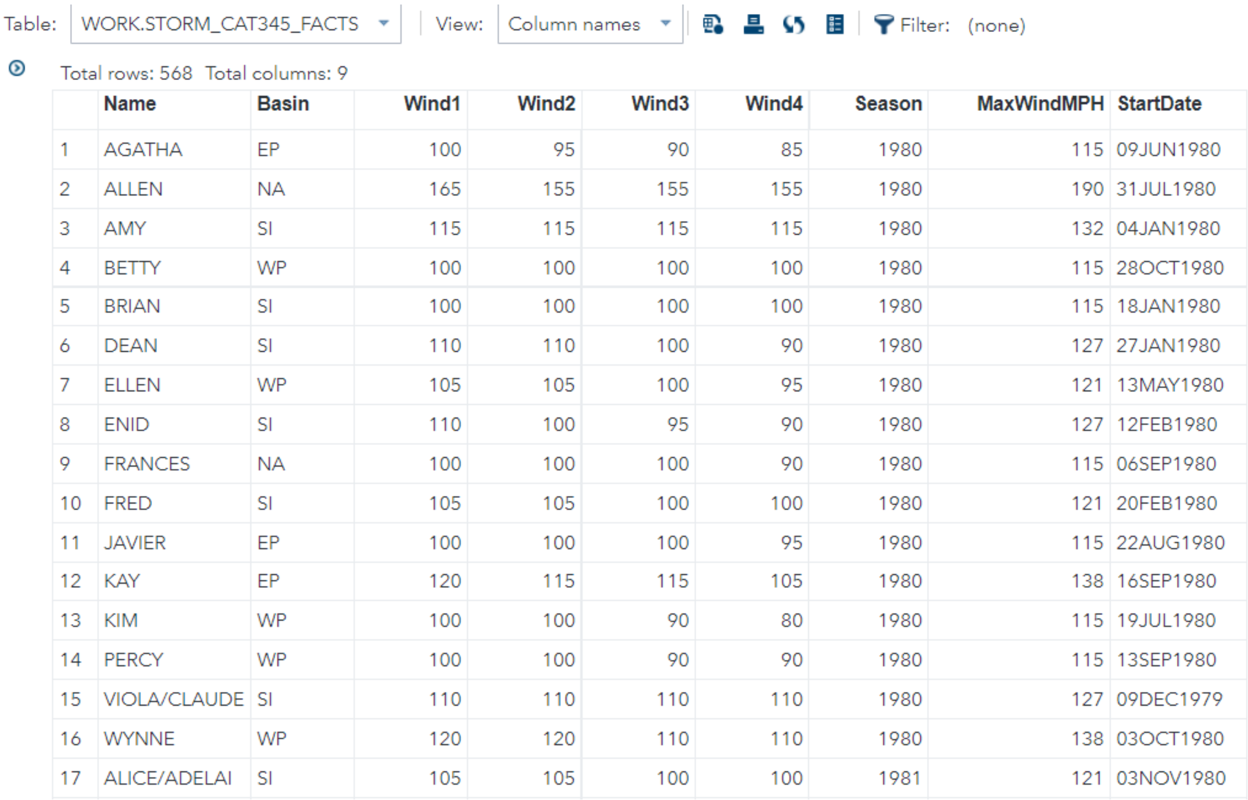
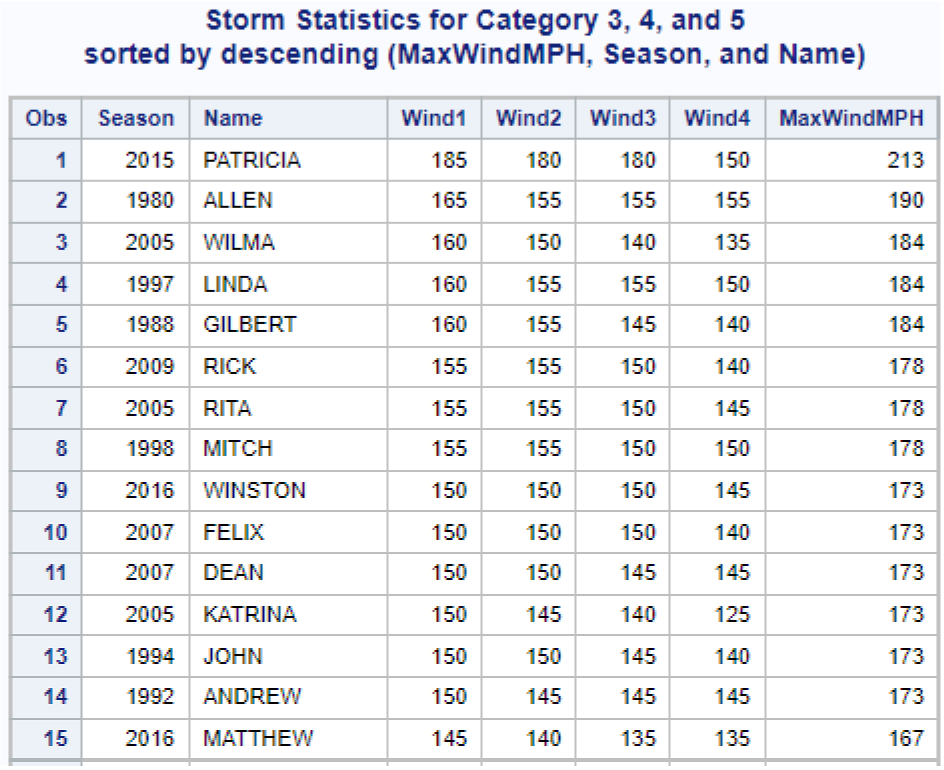


Table: WORK.STORM_CAT345_FACTS ▾ View: Column names ▾ Filter: (none)

Total rows: 568 Total columns: 9

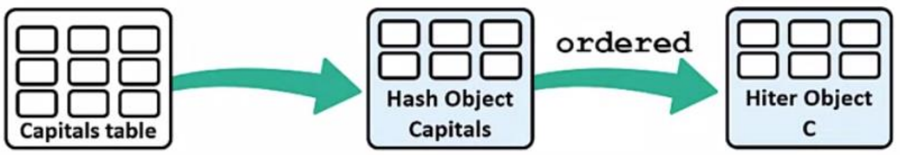
	Name	Basin	Wind1	Wind2	Wind3	Wind4	Season	MaxWindMPH	StartDate
1	AGATHA	EP	100	95	90	85	1980	115	09JUN1980
2	ALLEN	NA	165	155	155	155	1980	190	31JUL1980
3	AMY	SI	115	115	115	115	1980	132	04JAN1980
4	BETTY	WP	100	100	100	100	1980	115	28OCT1980
5	BRIAN	SI	100	100	100	100	1980	115	18JAN1980
6	DEAN	SI	110	110	100	90	1980	127	27JAN1980
7	ELLEN	WP	105	105	100	95	1980	121	13MAY1980
8	ENID	SI	110	100	95	90	1980	127	12FEB1980
9	FRANCES	NA	100	100	100	90	1980	115	06SEP1980
10	FRED	SI	105	105	100	100	1980	121	20FEB1980
11	JAVIER	EP	100	100	100	95	1980	115	22AUG1980
12	KAY	EP	120	115	115	105	1980	138	16SEP1980
13	KIM	WP	100	100	90	80	1980	115	19JUL1980
14	PERCY	WP	100	100	90	90	1980	115	13SEP1980
15	VIOLA/CLAUDE	SI	110	110	110	110	1980	127	09DEC1979
16	WYNNE	WP	120	120	110	110	1980	138	03OCT1980
17	ALICE/ADELAI	SI	105	105	100	100	1981	121	03NOV1980

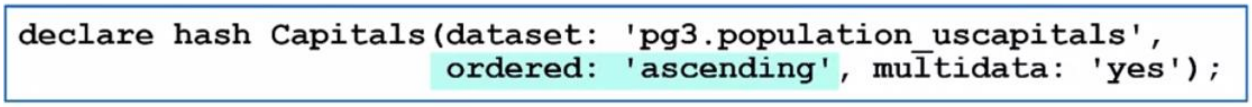
```
data work.storm_cat345_facts;
   if _N_=1 then do;
     if 0 then set pg3.storm_range;
     declare hash Storm(dataset:'pg3.storm_range');
     Storm.definekey('StartYear','Name','Basin');
     Storm.definedata('Wind1','Wind2','Wind3','Wind4');
     Storm.definedone();
     declare hash StormSort(Multidata: 'Yes', ordered: 'descending');
     StormSort.definekey('MaxWindMPH','Season','Name');
     StormSort.definedata('Season','Name','Wind1','Wind2','Wind3','Wind4','MaxWindMPH');
     StormSort.definedone();
   end;
   set pg3.storm_summary_cat345 end=lastrow;
   if Storm.find(key:year(StartDate),key:Name,key:Basin)=0 then StormSort.add();
   if lastrow=1 then
          StormSort.output(dataset: 'work.cat345_sort');
   keep Name Basin Wind1-Wind4 Season MaxWindMPH StartDate;
run;

title1 'Storm Statistics for Category 3, 4, and 5';
title2 'sorted by descending (MaxWindMPH, Season, and Name)';
proc print data=work.cat345_sort;
run;
title;
```



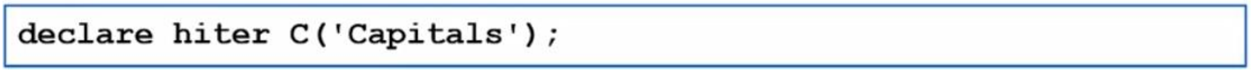
Storm Statistics for Category 3, 4, and 5
sorted by descending (MaxWindMPH, Season, and Name)

Obs	Season	Name	Wind1	Wind2	Wind3	Wind4	MaxWindMPH
1	2015	PATRICIA	185	180	180	150	213
2	1980	ALLEN	165	155	155	155	190
3	2005	WILMA	160	150	140	135	184
4	1997	LINDA	160	155	155	150	184
5	1988	GILBERT	160	155	145	140	184
6	2009	RICK	155	155	150	140	178
7	2005	RITA	155	155	150	145	178
8	1998	MITCH	155	155	150	150	178
9	2016	WINSTON	150	150	150	145	173
10	2007	FELIX	150	150	150	140	173
11	2007	DEAN	150	150	145	145	173
12	2005	KATRINA	150	145	140	125	173
13	1994	JOHN	150	150	145	140	173
14	1992	ANDREW	150	145	145	145	173
15	2016	MATTHEW	145	140	135	135	167

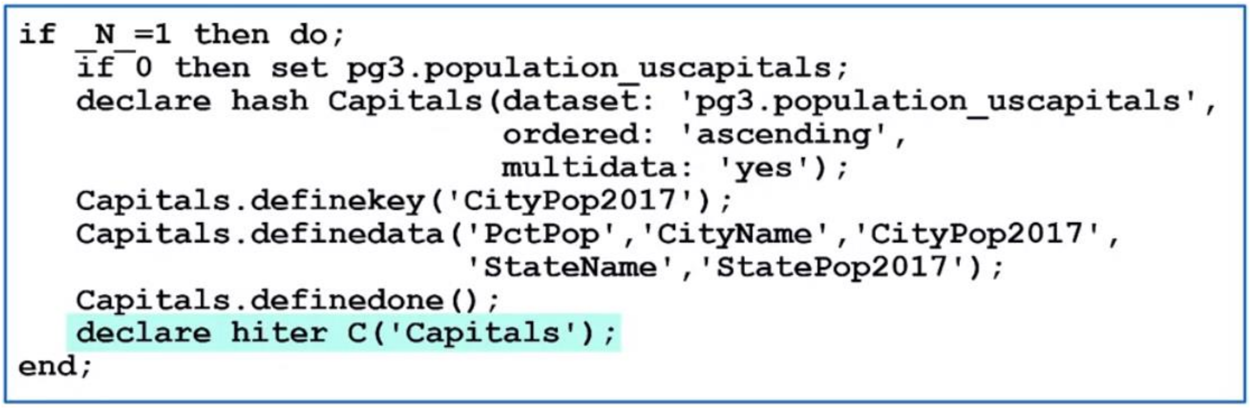




```
declare hash Capitals(dataset: 'pg3.population_uscapitals',
                        ordered: 'ascending', multidata: 'yes');
```



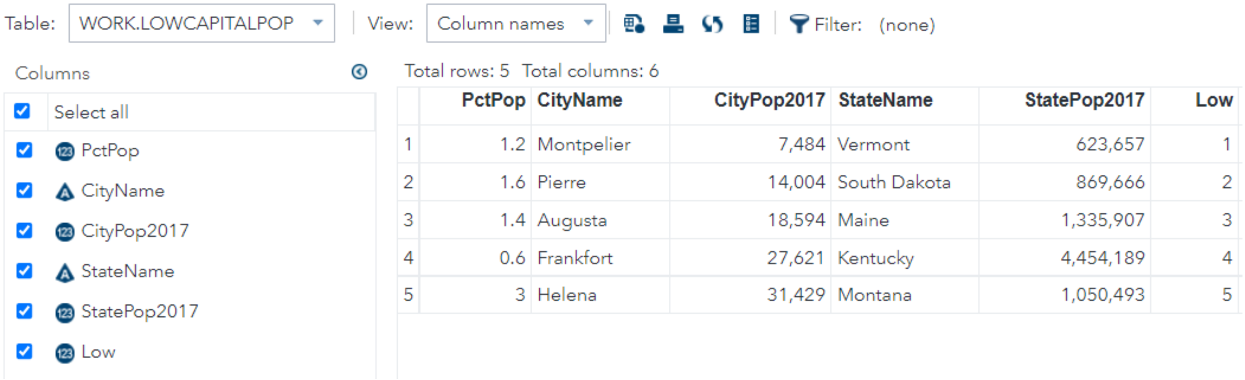
```
declare hiter C('Capitals');
```



```
if _N_=1 then do;
   if 0 then set pg3.population_uscapitals;
   declare hash Capitals(dataset: 'pg3.population_uscapitals',
                         ordered: 'ascending',
                         multidata: 'yes');
   Capitals.definekey('CityPop2017');
   Capitals.definedata('PctPop','CityName','CityPop2017',
                       'StateName','StatePop2017');
   Capitals.definedone();
   declare hiter C('Capitals');
end;
```

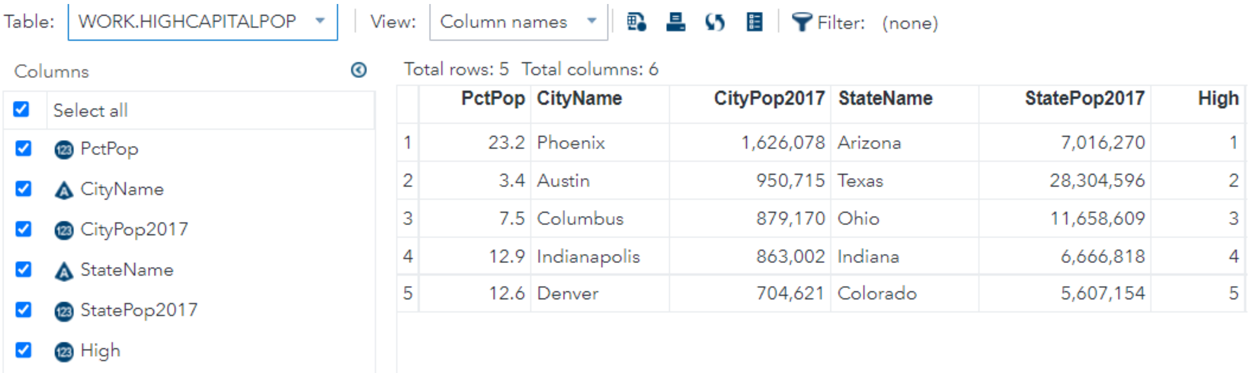
***;

```
*  Reading Data in Forward and Reverse Direction            *;
***************************************************************;

***************************************************************;
*  Demo                                                      *;
*  1) Notice the syntax for declaring and defining the hash   *;
*     object Capitals.                                        *;
*  2) Add a DECLARE statement after the Capitals hash object  *;
*     has been defined to create a hash iterator object       *;
*     named C.                                                *;
*       declare hiter C('Capitals');                          *;
*  3) Notice the remaining syntax of the DATA step. The first *;
*     DO loop iterates five times. The first time reads in the *;
*     first item of the hash iterator object, and the         *;
*     remaining iterations read the next item of the hash     *;
*     iterator object. The last DO loop also iterates five    *;
*     times. The first time reads in the last item of the hash *;
*     iterator object, and the remaining iterations read the  *;
*     previous item of the hash iterator object.              *;
*  4) Run the program. Verify that the LowCapitalPop table    *;
*     contains low city populations and the HighCapitalPop    *;
*     contains high city populations.                         *;
***************************************************************;

data work.LowCapitalPop(drop=High)
   work.HighCapitalPop(drop=Low);
  if _N_=1 then do;
    if 0 then set pg3.population_uscapitals;
    declare hash Capitals(dataset: 'pg3.population_uscapitals',
                  ordered: 'ascending',
                  multidata: 'yes');
    Capitals.definekey('CityPop2017');
    Capitals.definedata('PctPop','CityName','CityPop2017',
                'StateName','StatePop2017');
    Capitals.definedone();
          declare hiter c('Capitals');
  end;
  do Low=1 to 5;
    if Low=1 then C.first();
    else C.next();
    output work.LowCapitalPop;
  end;
  do High=1 to 5;
    if High=1 then C.last();
    else C.prev();
    output work.HighCapitalPop;
  end;
run;
```



Total rows: 5 Total columns: 6

	PctPop	CityName	CityPop2017	StateName	StatePop2017	Low
1	1.2	Montpelier	7,484	Vermont	623,657	1
2	1.6	Pierre	14,004	South Dakota	869,666	2
3	1.4	Augusta	18,594	Maine	1,335,907	3
4	0.6	Frankfort	27,621	Kentucky	4,454,189	4
5	3	Helena	31,429	Montana	1,050,493	5



Total rows: 5 Total columns: 6

	PctPop	CityName	CityPop2017	StateName	StatePop2017	High
1	23.2	Phoenix	1,626,078	Arizona	7,016,270	1
2	3.4	Austin	950,715	Texas	28,304,596	2
3	7.5	Columbus	879,170	Ohio	11,658,609	3
4	12.9	Indianapolis	863,002	Indiana	6,666,818	4
5	12.6	Denver	704,621	Colorado	5,607,154	5

Hash Object Advantages and Disadvantages

There are advantages and disadvantages when using hash objects.

Advantages of Hash Objects	**Disadvantages of Hash Objects**
simplifies programs by eliminating the need for multiple steps	different syntax and concept
fast processing time because the hash object is stored in memory	memory requirement
created at execution time, so dynamic sizing	
character and numeric data can be included	

```
***********************************************************;
*  LESSON 4, PRACTICE 7                                    *;
***********************************************************;
```

/*Practice Level 1: Reading Data in Forward and Reverse Direction

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.np_acres table contains acreage amounts for national parks.

Use a hash iterator to create work.LowAcres, which contains the 10 parks with the lowest number of acres, and work.HighAcres, which contains the 10 parks with the highest number of acres.

Open the p304p07.sas program in the practices folder. Review the DATA step syntax that is creating the Acres hash object.

Add a DECLARE statement to create the hash iterator named A, which is associated with the Acres hash object.

In the first DO loop, read the first 10 rows of the hash iterator,

which will be the national parks with the lowest number of acres. Output each row to work.LowAcres.

In the last DO loop, read the last 10 rows of the hash iterator,

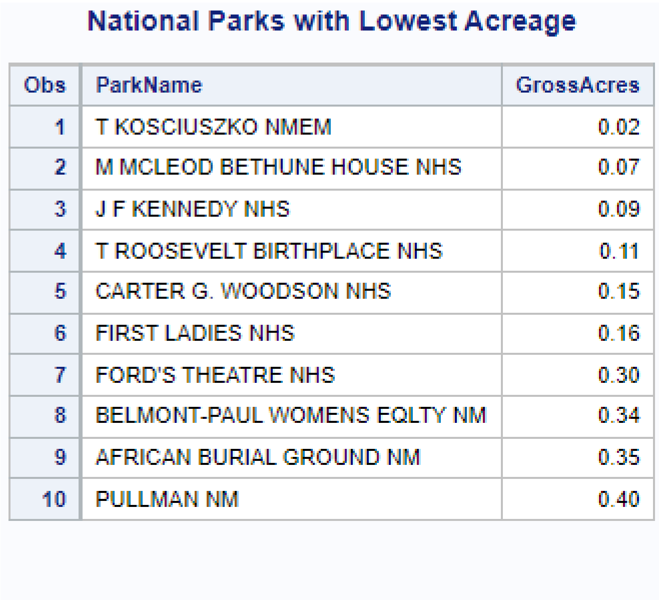
which will be the national parks with the highest number of acres. Output each row to work.HighAcres.

Run the program and view the results.

What is the GrossAcres value for the national park with the 10th highest acreage?

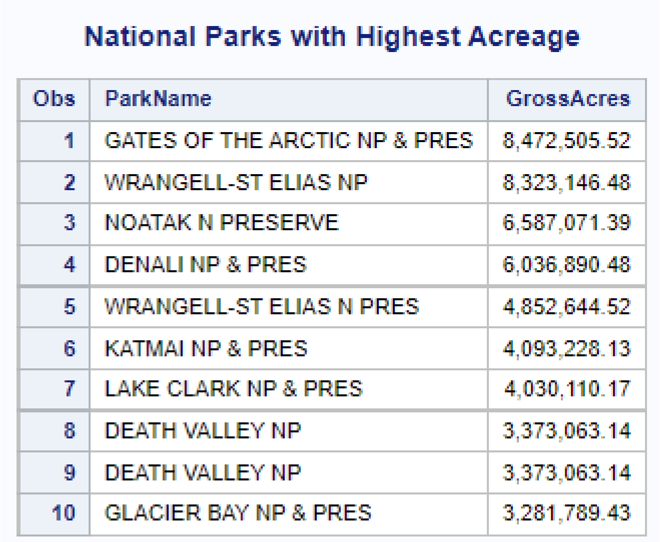
```
*/

data work.LowAcres work.HighAcres;

  if _N_=1 then do;

    if 0 then set pg3.np_acres(keep=ParkName GrossAcres);

    declare hash Acres(dataset:'pg3.np_acres',

                 ordered:'ascending', multidata:'yes');

    Acres.definekey('GrossAcres');

    Acres.definedata('ParkName','GrossAcres');

    Acres.definedone();

    /* declare a hash iterator */

    declare hiter A('Acres');

  end;

  do i=1 to 10;

    /* retrieve parks with the lowest number of acres */

          if i=1 then A.first();

    else A.next();

    output work.LowAcres;
```

```
   end;
   do i=1 to 10;
     /* retrieve parks with the highest number of acres */
             if i=1 then a.last();
                    else A.prev();
                    output work.HighAcres;
   end;
   drop i;
run;

title 'National Parks with Lowest Acreage';
proc print data=work.LowAcres;
run;
title;

title 'National Parks with Highest Acreage';
proc print data=work.HighAcres;
run;
title;
```



National Parks with Lowest Acreage

Obs	ParkName	GrossAcres
1	T KOSCIUSZKO NMEM	0.02
2	M MCLEOD BETHUNE HOUSE NHS	0.07
3	J F KENNEDY NHS	0.09
4	T ROOSEVELT BIRTHPLACE NHS	0.11
5	CARTER G. WOODSON NHS	0.15
6	FIRST LADIES NHS	0.16
7	FORD'S THEATRE NHS	0.30
8	BELMONT-PAUL WOMENS EQLTY NM	0.34
9	AFRICAN BURIAL GROUND NM	0.35
10	PULLMAN NM	0.40



National Parks with Highest Acreage

Obs	ParkName	GrossAcres
1	GATES OF THE ARCTIC NP & PRES	8,472,505.52
2	WRANGELL-ST ELIAS NP	8,323,146.48
3	NOATAK N PRESERVE	6,587,071.39
4	DENALI NP & PRES	6,036,890.48
5	WRANGELL-ST ELIAS N PRES	4,852,644.52
6	KATMAI NP & PRES	4,093,228.13
7	LAKE CLARK NP & PRES	4,030,110.17
8	DEATH VALLEY NP	3,373,063.14
9	DEATH VALLEY NP	3,373,063.14
10	GLACIER BAY NP & PRES	3,281,789.43

```
************************************************************;
*  LESSON 4, PRACTICE 8                               *;
************************************************************;
```

/*Practice Level 2: Reading Data in Forward and Reverse Directions

The pg3.storm_final table contains storm statistics such as MaxWindMPH for seasons 1980 through 2017.

Use a hash iterator to create work.LowWind, which contains the five storms with the lowest maximum wind speeds,

and work.HighWind, which contains the five storms with the highest maximum wind speeds.

Open the p304p08.sas program in the practices folder.

Review the DATA step syntax that is defining the Storm hash object.

Add a DECLARE statement to declare the hash object named Storm. Load the hash object with the pg3.storm_final table,

excluding the MaxWindMPH values that are missing. Specify the ORDERED argument with a value of ASCENDING and

the MULTIDATA argument with a value of YES.

Add another DECLARE statement to create the hash iterator names Stm, which is associated with the Storm hash object.

In the first DO loop, read the first five rows of the hash iterator, which will be the storms with the lowest maximum winds.

Output each row to work.LowWind.

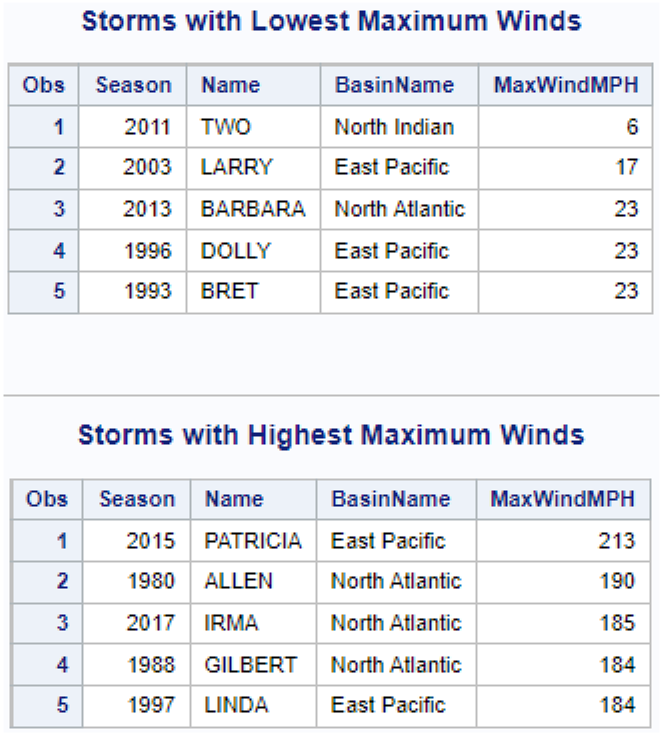
In the last DO loop, read the last five rows of the hash iterator, which will be the storms with the highest maximum winds.

Output each row to work.HighWind.

Run the program and view the results.

Based on the SAS log, how many observations were read from the data set and loaded into the hash object?

```
*/

data work.LowWind work.HighWind;

  if _N_=1 then do;

    if 0 then set pg3.storm_final

            (keep=Season Name BasinName MaxWindMPH);

    /* declare a hash object */

          declare hash Storm(dataset: 'pg3.storm_final(where=(MaxWindMPH ne .))',

                                                    ordered: 'ascending', multidata: 'yes');

    Storm.definekey('MaxWindMPH');

    Storm.definedata('Season','Name','BasinName','MaxWindMPH');

    Storm.definedone();

    /* declare a hash iterator */

          declare hiter Stm('Storm');
  end;
  do i=1 to 5;
    /* retrieve storms with the lowest maximum winds */
              if i=1 then Stm.first();
              else Stm.next();
              output work.LowWind;
  end;
  do i=1 to 5;
    /* retrieve storms with the highest maximum winds */
              if i=1 then Stm.last();
              else Stm.prev();
              output work.HighWind;
  end;
  drop i;
run;

title 'Storms with Lowest Maximum Winds';
proc print data=work.LowWind;
run;
title;

title 'Storms with Highest Maximum Winds';
proc print data=work.HighWind;
run;
title;
```



Storms with Lowest Maximum Winds

Obs	Season	Name	BasinName	MaxWindMPH
1	2011	TWO	North Indian	6
2	2003	LARRY	East Pacific	17
3	2013	BARBARA	North Atlantic	23
4	1996	DOLLY	East Pacific	23
5	1993	BRET	East Pacific	23

Storms with Highest Maximum Winds

Obs	Season	Name	BasinName	MaxWindMPH
1	2015	PATRICIA	East Pacific	213
2	1980	ALLEN	North Atlantic	190
3	2017	IRMA	North Atlantic	185
4	1988	GILBERT	North Atlantic	184
5	1997	LINDA	East Pacific	184