

Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

This article presents two examples of the practical use of hash objects. The first example illustrates the use of hash objects for table lookup, and compares hash object performance with comparable code using arrays and formats. The second example uses a hash table to extract 80 top customers from a large table without sorting the data.

Introduction to DATA step component objects:

SAS now provides two pre-defined component objects for use in a DATA step: the hash object and the hash iterator object. These objects enable you to quickly and efficiently store, search, and retrieve data based on lookup keys.

The DATA step component object interface enables you to create and manipulate these component objects by using statements, attributes, and methods. You use the DATA step object dot notation to access the component object's attributes and methods in order to:

- Provide in-memory data storage and retrieval with the hash object.
- Define a data component and key component (think table and index).
- Load data from a SAS table into the hash object (input data does not have to be sorted).
- Lookup a data row based on key values.
- Extract data in sort order with the hash iterator object.
- Add or delete data rows dynamically.

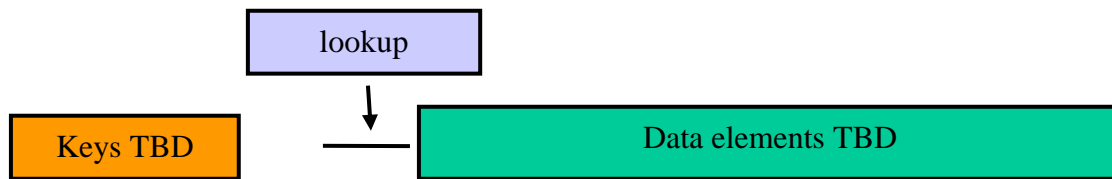
The hash and hash iterator objects have one attribute, fourteen methods, and two statements associated with them. Full documentation is included in the SAS online doc. This article provides practical examples of using these methods and statements.

Table lookup with a hash object

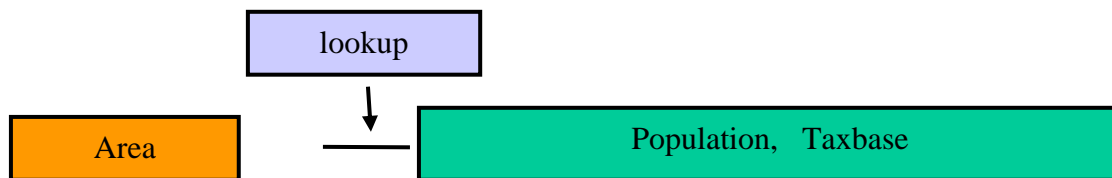
In this example, we start with an existing SAS data set, *work.areadata*, that contains demographic data for geographical areas. The key is an area code stored in the variable *area*, and the demographics that need to be extracted are the population size and the size of the tax base.

To implement table lookup with a hash object:

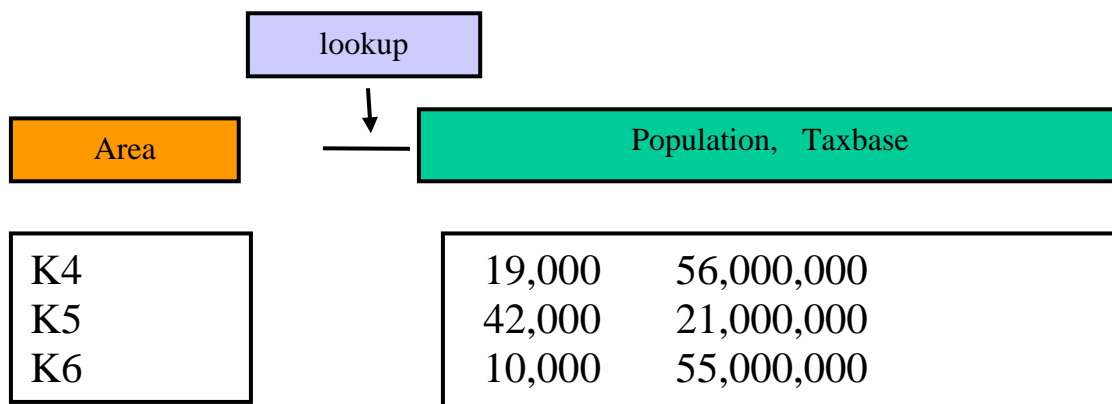
1. Create an object and name it "lookup."



2. Define keys and data elements.



3. Load data from existing SAS data set.



Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

The SAS code to implement steps 1 through 3 is as follows:

```
data combinedA ;  
  if _n_ = 1 then do ;  
    if 0 then set work.areadata(keep=area population taxbase) ;  
    declare hash lookup(dataset: 'work.areadata') ;  
    lookup.definekey('area') ;  
    lookup.definedata('population', 'taxbase') ;  
    lookup.definedone() ;  
  end ;
```

The remainder of the DATA step for the actual lookup is:

```
  set work.basetable ;  
  lookup.find() ;  
run ;
```

Where the SAS data set *work.basetable* also contains a variable *area* that supplies a key value for the table lookup.

Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

Comparing hash objects with arrays:

- Hash objects can use a character variable, a numeric variable, or a combination of variables as the key.
- The size of a hash object is not specified when it is created.
- Hash objects can store multiple data items per key.
- One hash object can store both character and numeric data items.

The SAS code to load an array and use it for a table lookup is

```
data makeformat;
  retain fmtname 'areaid' type 'IN';
  set work.areadata(keep=area rename=(area=start));
  label = put(_n_,8.);
run;

proc format cntlin=makeformat fmtlib;
run;

data combinedC;
  array values (27,2) _temporary_;
  if _n_ = 1 then do loop = 1 to totobs;
    set work.areadata nobs=totobs;
    areaid = input(area,areaid.);
    values(areaid,1) = population;
    values(areaid,2) = taxbase;
  end;
  set work.basetable;
  areaid = input(area,areaid.);
  population = values(areaid,1);
  taxbase = values(areaid,2);
run;
```

The informat that is created maps a character area code to a sequence number in order to get around the limitation of arrays to integer index values.

Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

Comparing hash objects with formats:

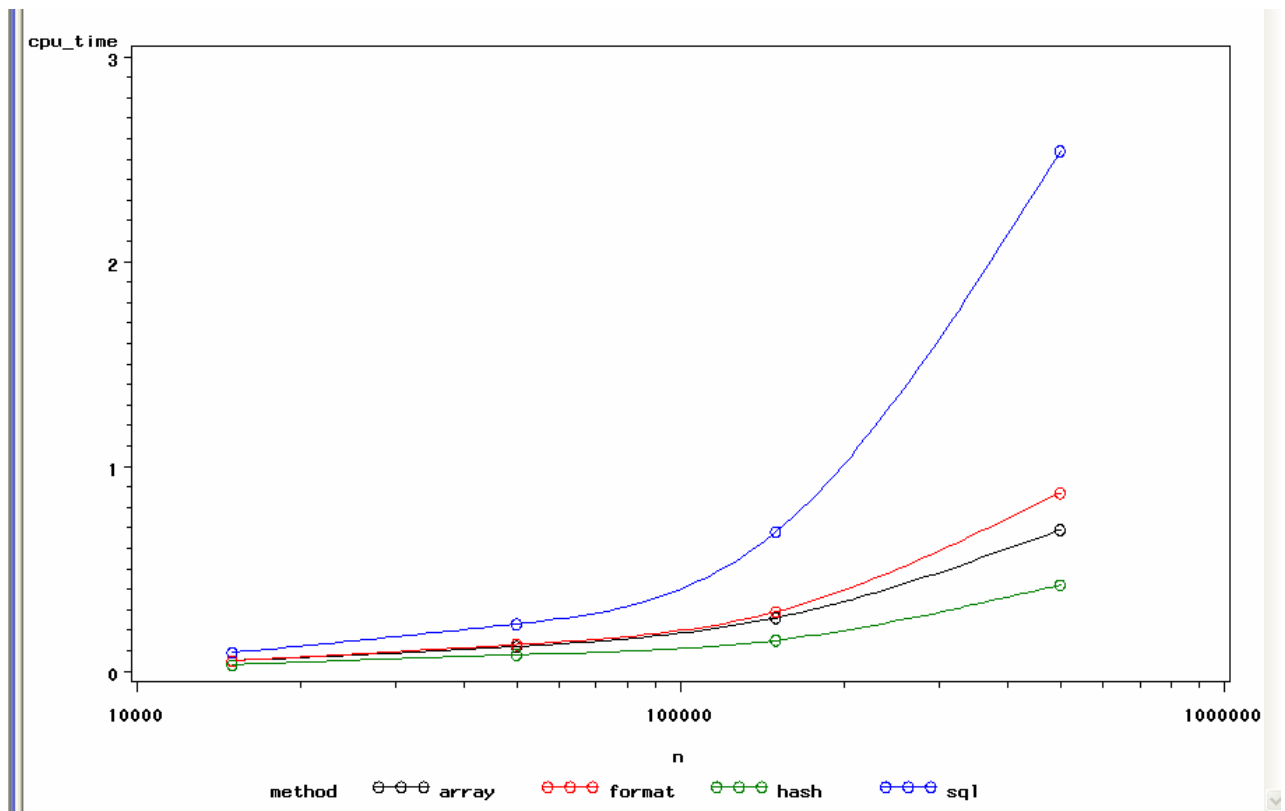
- A hash object is faster than a format.
- A hash object uses less memory than a format.
- Hash objects can store multiple data items per key.
- With large amounts of data, formats take up large amounts of disk space; hash objects do not.

The SAS code to use SAS formats directly is:

```
data combinedB;  
set work.basetable;  
population = input(area,pop.);  
taxbase = input(area,tax.);  
run;
```

You need to create two informats in this case, because a single informat returns only one value at a time.

Note that the hash object code is shorter and less complex than either the array based code or the format based code. It is also more efficient. The following benchmark compares the costs of using the hash object code with arrays, formats and a standard SQL inner join.



Case Study: Finding the Top 80

An organization has a history table with information on approximately 20 million customers. This includes the amount of purchases made each month for 48 months, as well as demographic data. Knowledge workers need to extract the 80 top customers that satisfy a particular profile based on the demographic data. Customers are ranked based on the value of their past purchases.

The first few rows in the history data that is available are:

customerID	phone	address	purchase1
1000001	513 9201233	17639 street name, city name, ON, L5P 1M7	120
1000002	(905)6061405	42639 street name, city name, ON, M2P 1M7	170
1000003	(905)6988306	43660 street name, city name, ON, K1P 1M7	231
1000004	905-463-4117	13910 street name, city name, ON, M6P 1M7	294
1000005	513 7864257	26883 street name, city name, ON L8P 1M7 Canada	336
1000006	(416) 973 7142	27844 street name, city name, ON, K8P1M7	373
1000007	513 404 5575	27700 street name, city name, ON, L4P1M7	421

- Customer ID is the primary key.
- Purchase1 is one of the 48 columns containing total purchases made during each of the past 48 months.
- Phone number – multiple positions for area code.
- Address field with postal code in multiple formats.

There are a number of business rules for assigning a total value to a customer.

- Total value of a customer is the weighted mean of purchases over the past 48 months, with purchases during the past 12 months weighted double.
- Only include customers in the 905 telephone area.
- Only include customers with postal codes starting with “L”.

The final extract will contain the top 80 customers in descending order by total value:

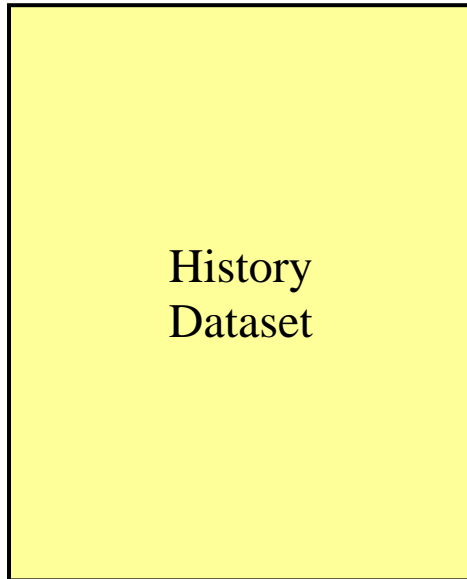
	totalvalue	customerID	phone	address
1	5413	1001299	(905)4058224	27627 street name, city name, ON, K7P1M7
2	5411	1097199	(905) 767 9094	52279 street name, city name, ON K3P1M7
3	5405	1093399	905 2106868	17896 street name, city name, ON, K9P 1M7
4	5403	1078799	905-523-1200	5491 street name, city name, ON, L9P1M7
5	5403	1026999	513 6651082	1681 street name, city name, ON K2P1M7
...
75	5388	1101199	(513) 905 8283	48860 street name, city name, ON, L8P 1M7
76	5388	1132299	513-521-7283	28575 street name, city name, ON K6P 1M7 Canada
77	5388	1025899	416-424-8155	54921 street name, city name, ON L8P 1M7 Canada
78	5388	1129999	(905)9606406	20864 street name, city name, ON M6P1M7
79	5388	1147299	(905) 797 5730	57641 street name, city name, ON, M1P1M7
80	5388	1009999	(513)8246037	40992 street name, city name, ON, L7P 1M7

Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

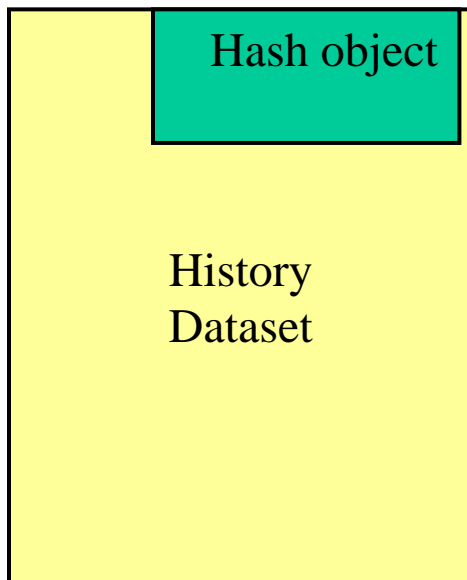
There are at least two strategies for finding the top 80 customers:

- SAS®9: Use a single data step and one sequential read of the data.
- The alternative: Calculate value in a data step and then sort the resulting table.

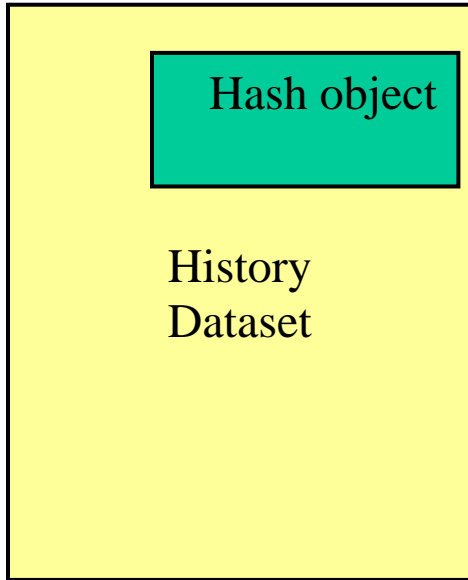
The process for using a hash object as a moving window is as follows:



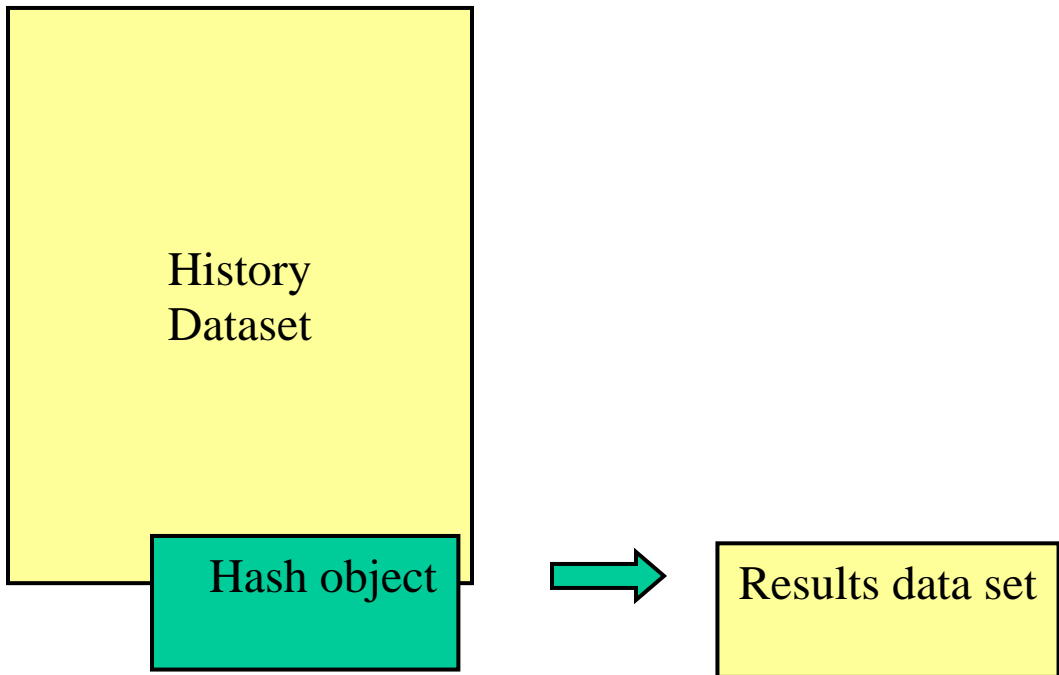
Step 1: Start with the history SAS data set.



Step 2: Define and fill a hash object with the first 80 rows of the history SAS data set.



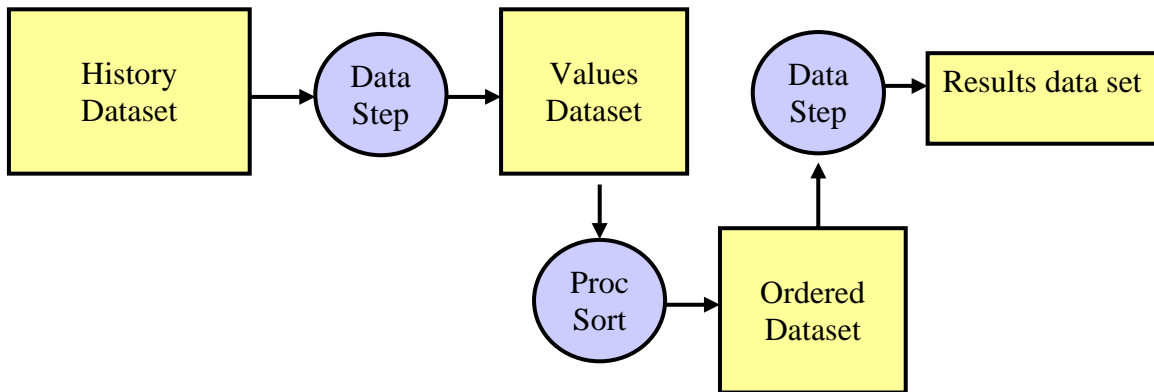
Step 3: Move hash object down one row at a time. If the new row is larger than the smallest item currently in the hash object, replace that smallest item by the new row



Step 4: When hash object reaches the bottom, it contains the top 80 items in the history SAS dataset. Output the contents of the hash object to a results SAS dataset.

Using Hash Objects in SAS 9
 By Bill Fehlner, SAS Institute
 July 2005

An alternative strategy, using PROC SORT, has the following data flow:



The technique using PROC SORT requires more memory, more scratch disk space and more CPU cycles than the technique using a hash object.

The following benchmark compares the cost of the technique using the hash object with the technique using PROC SORT.

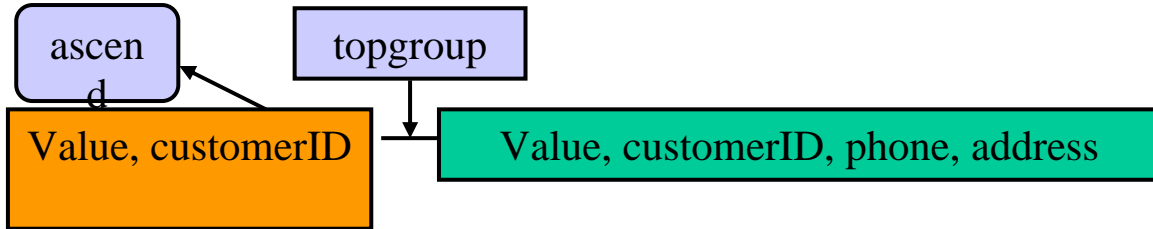
Total rows	Group Size	Hash CPU	SORT CPU	Ratio
10001	80	.05	.09	1.80
50001	80	.12	.24	2.00
250001	80	.50	1.12	2.24
1000001	80	1.88	5.03	2.67
4000001	80	6.95	19.56	2.81

Each CPU value is an average over three runs.

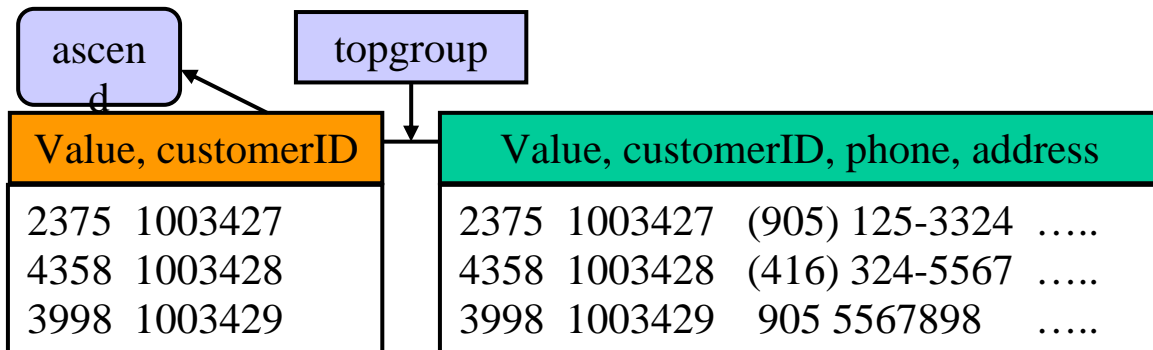
Note one other major advantage of the hash object in this situation: Unlike the SORT based technique, the amount of memory and the amount of scratch space required by the hash object technique does not increase as the size of the history SAS data set increases.

The first steps in implementing a top 80 search with a hash object is similar to using a hash object as a lookup table, namely:

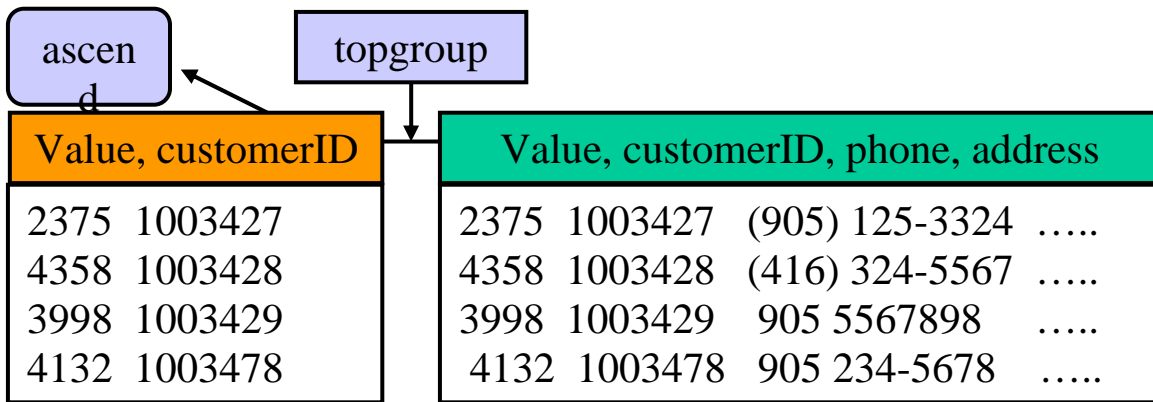
1. Create an object and name it "topgroup".
2. Define keys and data elements.
3. Create a hash iterator object to allow access to the data in the hash object in key order.



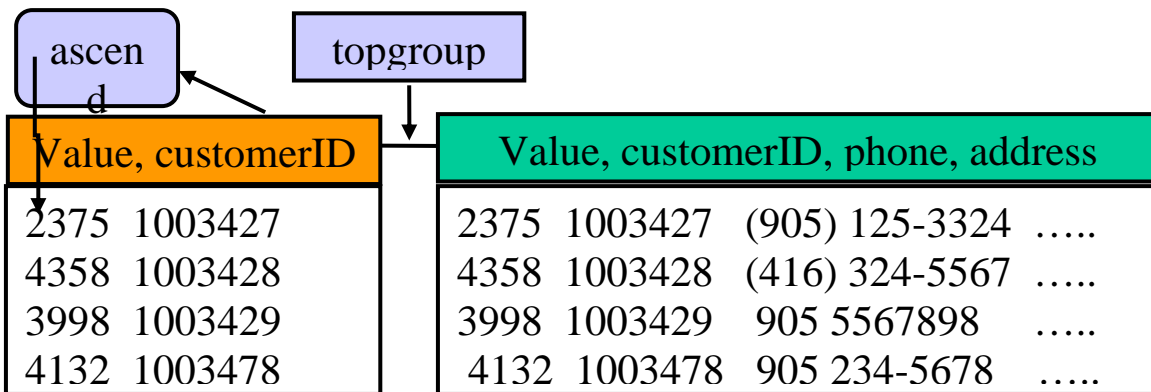
4. Load the first 80 rows of data from the history SAS data set.



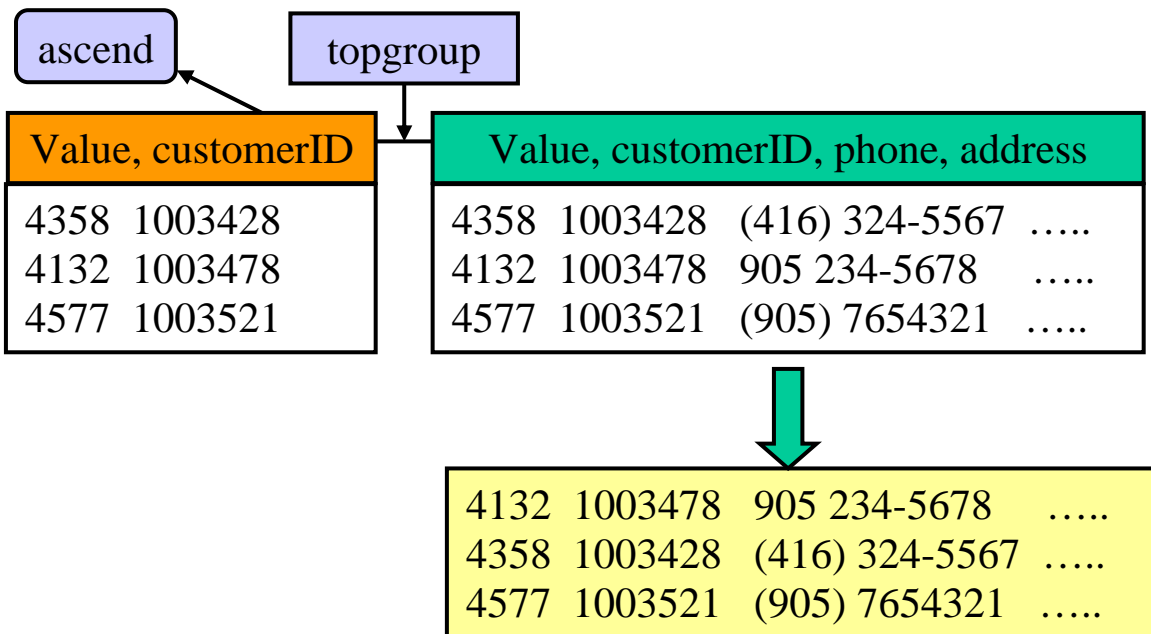
5. Add data from program data vector when the calculated value is larger than the current smallest stored value.



6. Locate the current smallest value.
7. Remove the item corresponding to the smallest value.



8. Continue to add and remove items until all the history data has been processed.
9. Write results out to a SAS dataset in ascending order.



Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

The code to implement the process outlined above is as follows. Comments indicate which code segments correspond to the steps described above.

```
data _null_ ;
  length rc 8;
  retain minvalue mincustomerid;
  /* declare variables in hash object */
  length totalvalue 8 customerID 8 phone $ 15 address $50;
  format totalvalue 10.;
  /* initialize hash object and hash iterator (steps 1 to 3) */
  if _n_=1 then do;
    if 0 then set work.history(keep=totalvalue customerid phone address);
    declare hash topgroup(ordered='ascending');
    declare hiter ascend('topgroup');
    topgroup.definekey('totalvalue','customerid');
    topgroup.definedata('totalvalue', 'customerID','phone','address');
    topgroup.definedone();

    /* load first block of rows into hash object (step 4)*/
    do loop = 1 to 80;
      link getdatarow;
      link calctotalvalue;
      rc = topgroup.add();
    end;
    rc = ascend.first();
    minvalue = totalvalue;
    mincustomerid=customerid;
    rc = ascend.next(); /* avoid pointing to a row you may remove later*/
  end; /* the _n_=1 loop */

  /* process remaining data rows (step 8) */
  link getdatarow;
  link calctotalvalue;

  /* test if value is greater than minimum (step 5, 6, and 7) */
  if totalvalue gt minvalue then do;
    rc = topgroup.add();
    rc = topgroup.remove(KEY:minvalue,KEY:mincustomerid);
    rc = ascend.first();
    minvalue = totalvalue;
    mincustomerid = customerid;
    rc = ascend.next(); /* avoid pointing to a row you will remove later*/
  end;
```

Using Hash Objects in SAS 9
By Bill Fehlner, SAS Institute
July 2005

```
        /* output the results (step 9) */  
if eof then do;  
    rc = topgroup.output(dataset:"work.results");  
end;  
return;  
  
getdatarow: /* read row from input dataset */  
    set work.history end=eof;  
return;  
  
calctotalvalue: /* use simple business rule to calculate total value */  
    totalvalue = (2* sum(of purchase37-purchase48) + sum(of purchase1-purchase36))/  
                (2* n(of purchase37-purchase48) + n(of purchase1-purchase36));  
return;  
  
run;
```

Conclusion:

The hash object provides an efficient, convenient mechanism for quick data storage and retrieval. The code required for implementing a hash object is no more complex (and may be less) than alternative strategies used in SAS 8.