# Getting Connected — The Power of Using Hash Objects

### ABSTRACT

The Hash Object and the Hash Iterator Object<sup>1</sup>, two data step component objects, offer a simple and powerful solution for many complex tasks. This paper will describe the hash object and show how to merge a number of data sets using hash objects, base SAS, and object dot notation. It will also illustrate how the program data vector (PDV) enables the interaction of data set variables and the data variables within the objects. The hash object is an in-memory solution allowing direct access via keys during execution. It is fast and can contain an unlimited number of key items and data items. The only limitation is available memory. As memory size increases and the constraints of available memory lessen, the usefulness of hash objects can only increase.

#### INTRODUCTION

Upon her balcony in the privacy of her garden, Shakespeare's Juliet, mused on Romeo's forbidden surname,

What's in a name? That which we call a rose By any other word would smell as sweet.<sup>2</sup>

The hash object is a Data Step Component Interface (DSCI) or, more completely, a Data Step Component Object Interface. This is a descriptive name for a sweet thing. What is in the name Data Step Component Object Interface? Data Step: It can only be used in a data step. It cannot be used in a procedure. Component Object: It is a predefined structure, containing a hash function, map or multi-map, slots or hash buckets, and Adelson-Velsky and Landis balanced binary trees for referencing keys and storing data.

Using object dot notation, the programmer only needs to declare, instantiate, and refine the definition of the hash object by assigning values as key items and, optionally, data items. The hash object is then created as a variable in the program data vector at execution time, rather than compile time, and will only exist for the duration of the data step. This difference in timing creates an issue between the Data Step PDV and the object. Data Step programming must be used to create variables in the program data vector at compile time to interface with the key items and keyed data items in the hash object. Data Step programming creates valables in the PDV that are either numeric or character. Object dot notation creates a variable that is an object containing keyed data items. If composite keys are defined, they can be a mixture of character and numeric types. The data definition can also be a mixture of types. The bridge between the hash object and the data step program is the PDV. Variables created with Data Step programing at compile time host the items in the hash object and must match the attributes of the variables with which the hash object is loaded.

	\land Cust_id	A policy_id	A First_name	💧 Last_Name	🔌 city	(A) Country
1	pe12489	81880	Seth	Pecksniff	London	Eng
2	sh15878	11489		Shakespeare	London	Eng
3	sh15878	98355	Shakespeare		London	Eng
4	ma48798	22389	Thomas	Mann	Princeton	USA
5	ch12489	81889	Martin	Chuzlewit	London	Eng
6	do78521	12587	Amy	Dorrit	London	Eng
7	to12587	36901	Leo	Tolstoy	Yasnaya Polyana	Russia
8	ga78521	38525	Elizabeth	Gaskell	Knutsford	Eng
9	ni33691	66381	Nicholas	Nickleby	London	Eng
10	cl11483	22871	Arthur	Clennam	London	Eng
11	go78224	36999	Oliver	Goldsmith	London	Eng
12	ti78902	65547	Montague	Tigg	London	Eng
13	ti78902	58749	Tigg	Montague	London	Eng
14	ta48963	22158	Mark	Tapley	London	Eng
15	ch78984	44879	Chevy	Slyme	London	Eng
16	ja48548	22487	John Jarndyce		Bleak House	Eng A Policy
17	gu77892	14856	William	Guppy	London	Eng 1 11489

A Policy_id	💧 claim_no	🔌 Vin
81889	7141	JHG00000030000003
12587	7249	1FA00000050000004
22871	7681	1G1000003000008
98355	7897	JHG00000030000018
36901	8113	JHG00000030000005
	Policy_id 81889 12587 22871 98355 36901	Policy_id         claim_no           81889         7141           12587         7249           22871         7681           98355         7897           36901         8113

Data sets used for illustration. The code for the creation of the data sets is included at the back of this paper.

	A Policy_id	٨	vin		Year		Make	1	A
. 1	11489	1FA000	000030000001	2003	1	Ford		F3	
2	22389	1G100	000050000002	2005		Che	vrolet	G5	
3	81889	JHG00	000030000003	2003	1	Hone	da	H3	
4	12587	1FA000	00050000004	2005		Ford		F5	
5	36901	1GC00	000030000005	2005	6	Che	vrolet	G5	
6	38525	JHG00	000030000006	2003		Hone	da	H3	
7	66381	1FB000	000050000007	2005		Ford		F5	
8	22871	1G100	000030000008	2003		Che	G3		
9	36999	JHG00	000050000009	2005		Hone	da	H5	
10	81889	1FA000	000030000010	2003		Ford		F3	
11	12587	1G100	000050000011	2005		Che	vrolet	G5	
12	36901	JHG00	000030000012	2003		Hone	da	H3	
13	65547	1FA000	000030000013	2003		Ford	[]	F3	
14	65547	1G100	000050000014	2005		Che	vrolet	G5	
15	58749	JHG00	000030000015	2003		Hone	da	H3	
16	58749	1FA000	000050000016	2005		Ford		F5	
17	22158	1GC00	000030000017	2005		Che	vrolet	G3	
18	98355	JHG00	000030000018	2003		Hone	da	H3	
19	22158	JHG00	000030000019	2003	1	Hone	da	H3	
20	44879	1FB000	000050000020	2005		Ford	l	F5	
21	44879	1G100	000030000021	2003	1	Che	vrolet	G3	
22	22487	JHG00	000050000022	2005		Hone	da	H5	
23	14856	1FA000	000030000023	2003		Ford		F3	
24	65547	1G100	000050000024	2005		Che	vrolet	G5	
25	65547	JHG00	000030000025	2003		Hone	da	H3	

Bevery Johnson was a remarkable woman and an adventuress. She was the first woman to climb the face of El Capitan in Yosemite National Park. It took her 10 days to make the ascent. When she reached the top, a videographer was waiting to interview her. The interviewer asked her what kept her going. She said that she kept asking herself the same question, "How do you eat an elephant?" The answer, "One bite at a time." The answer is very appropos hash objects in the era of big data. The hash object is capable of containing an infinite number of key items and data items in each slot or bucket; however, since it is a memory resident object, it is constrained by available memory. To handle large amounts of data, it needs to take the data one bite at a time. To work efficiently with hash objects, it is necessary to judiciously use keep options and subsetting where options, or pre-filter data to limit the number of variables and the number of observations to what is required.

```
data _null_;
     *if n =0 then set work.cust(keep=policy id First name Last name Country);
    attrib policy id length= $ 10
        First name length= $ 15
        Last_Name length= $ 15
         Country length= $15
    if _n_=1 then do;
         dcl hash CMP(dataset:'work.cust(where=(Country="Russia"or Country="USA")');
         CMP.definekey ('policy_id');
         CMP.definedata ("policy_id", "First_name", "Last_Name", "Country");
         CMP.definedone():
         call missing(Cust_id, First_name, Last_Name, city, Country, Cust_dt);
    end:
    set work.pol;
    rc=CMP.find();
                                                                                                                policy id
                                                                                                 \wedge
                                                                                                     First name
                                                                                                                    Last Name
                                                                                                                                     Country
    if rc=0 then CMP.output(dataset: 'DumpH (where=(Country="Russia")');
                                                                                  36901
                                                                                                 Leo
                                                                                                                Tolstoy
                                                                                                                               Russia
    else CMP.output(dataset:'else1');
                                               policy id
                                                             First name
                                                                            Last Name
                                                                                             Country
                                                                        run;
                                          36901
                                                         Leo
                                                                         Tolstoy
                                                                                        Russia
                                           22389
                                                          Thomas
                                                                                        USA
                                                                         Mann
```

To write a declare statement you can use "declare" or the abbreviated "dcl". There are two ways to declare and instantiate a hash object. The first offers the convenience of combining the declaration and the instantiation in one statement:

dcl hash h\_obj ( );

The second is more versatile breaking the declaration and instantiation into separate statements. These statements can be separated by code.

```
dcl hash h_obj;
h_obj=_new_ hash ( );
```

The first statement declares the name reference, h\_obj, a hash object, and the second statement uses the operator \_new\_ to assign the object to h\_obj. This second method shows more clearly that the hash object is a variable in the PDV.

When a data step set statement is used to input values into the PDV and retrieve values from a hash object, the timing of the flow of data into any variables that share the same name has to be considered. The data step inputs its values first, and any values drawn from the hash object overwrite the values from the data step.

data work.cust2;	💧 Cus	it_id 🏼 🏼	policy_id	▲ First_name		Last_Name		city		Country
set work.cust;	1 sh15878	9	8355	Shakespeare			Londor	1	Eng	
where policy_id in ("98355","65547");	2 ti78902	6	5547	Montague	Tigg		Londor	1	Eng	
run;										
data work.share;			Cust id	First name		last Nam	e 向	10		1
if _n_=0 then set work.cust(keep=cust_id first_name Last_1	name);	sh1587	78		Sha	kespeare			0	
/*attrib policy_id length= $$10$		ti78902	2 T	iaa	Mor	ntaque			0	
First_name length= \$ 15										
Last_Name length= \$ 15										
;*/										
if $n_=1$ then do;										
dcl hash SHAR(dataset:'work.cust(where=(last_name=	="Shakespo	eare"oi	r last_name	="Montague"	)');					
SHAR.definekey ('cust id');	1									
SHAR.definedata ("First name","Last Name");				First name	A	Last Nam	e			
SHAR.definedone();			1 1	Figg	Mo	ntague				
*call missing(First name, Last Name);			2		Sha	akespeare				
end;										
set work.cust2 (keep=cust id first name Last name);										
rc=SHAR.find();										
If rc ne 0 then call missing(first name, last name);										
Shar.output(dataset:'H_Dump');										
run;										

Methods used for traversing the hash key items using a hash itereator or do loops.

With the default of argument tag Multidata: 'N'

It is easy to declare a hash iterator to traverse the data in the hash object; for example:

dcl Hiter hobji('hobj');

The methods used to traverse the data are first, next, prev, last, and setcur(key:'value'). If the argument tag Ordered: 'Y' is used in the hash declaration, the data can be traversed and output in ascending or descending order. The key items are traversed sequentially similar to the data in an SQL table with a unique key.

With the Multidata argument tag "Y"

The data in a hash object needs different methods to traverse the keys when the Multidata argument tag is "Y" because the keys must be accessed sequentially and hierarchically. The hash object has a head node or header node, and non-unique data is attached as an ordered list or data chain, so one must traverse the data across the head nodes and down the linked data.

It seems to me that the actual structure of the keys and data in a hash object most resembles a web, the strands a mesh of references and the nodes the intersections. The following is a simple conceptual representation to aid understanding the methods used with the hash iterator and do loops to extract data from a hash object.



Multidata: 'N'

There is also a object.setcur(key: 'value'). It is usually used with a lteral to begin an iteration at a specific point in the data.

If the Multidata:'Y' argument tag is used in the hash declaration, nested do loops can be used with methods to extract data from the hash object. One loop crosses the head nodes and the other runs down or up the hierarchially arranged chain of non-unique key data.



There are also methods for operating in the oposite direction.

object.find ()

object.has\_prev (result: r ) object.

object.find\_prev( )

Object dot notation has two attribute statements. One returns the number of head keys and the other returns the size in bytes of the keys and data in a single hash Item. These can be used in a calculation to get the approximate size of the hash object. When using the Multidata argument tag 'Y' the size of hash items can greatly vary.

variable\_name=object.Item\_size
variable\_name=object.Num\_items

data work.loop out all		A Policy_id	🔌 vin	A Year	A Make	💧 Cust_id	A Hist nam	Last Nam	🕑 rc 😡	r
		1 11489	1FA00000030000001	2003	Ford	sh15878		Shakespeare	0	
work.loop_out_unq		2 98355	JHG00000030000018	2003	Honda	sh15878	Shakespeare	Marrie	0	0
work.loop_out_dup;		4 81389	JHG000003000002	2005	Honda	ch12489	Martin	Chuzlewit	0	0
attrib Policy id length \$ 10		5 81389	1FA00000030000010	2003	Ford	ch12489	Martin	Chuzlewit	0	1
auto i oney_iu iengui- \$ 10		6 12587	1FA0000005000004	2005	Ford	do78521	Amy	Dorrit	0	0
vin length= \$ 17		7 12587	1G10000050000011	2005	Chevrolet	do78521	Amy	Dorrit	0	1
Vear length-\$4		8 36901 9 36901	IHG0000030000012	2005	Hooda	101258/ to12587	Leo	Tolstoy	0	1
$\int dt = \int dt = \frac{1}{\sqrt{2}} \frac{dt}{dt} = \frac{1}{\sqrt$		10 38525	JHG0000003000006	2003	Honda	ga78521	Elizabeth	Gaskell	0	0
Make length= \$17		11 66381	1FB00000050000007	2005	Ford	ni33691	Nicholas	Nickleby	0	0
•	1	12 22871	1G1000003000008	2003	Chevrolet	cl11483	Arthur	Clennam	0	0
, TC 1.1 1		13 36999	JHG0000005000009	2005	Honda	g078224 678902	Oliver	Goldsmith	0	0
If $n_{=1}$ then do;		15 65547	1G10000050000014	2005	Chevrolet	\$78902	Montague	Tiog	0	1
dcl hash lp(dataset;'work.pol', ordered:'Y', multidata:'Y');		16 65547	1G1000005000024	2005	Chevrolet	178902	Montague	Tigg	0	1
ln defineken ('Delien id').		17 65547	JHG000003000025	2003	Honda	ti78902	Montague	Tigg	0	1
ip.definekey( Policy_id );		18 58749	JHG0000003000015	2003	Honda	178902	Tigg	Montague	0	0
lp.definedata('Policy_id', 'vin', 'year', 'make');		20 22158	1GC000003000017	2005	Chevrolet	1/03U2 ts48963	Mark	Tapley	0	0
In definedone():		21 22158	JHG00000030000019	2003	Honda	ta48963	Mark	Tapley	0	1
ip.dcmicdolic(),		22 44879	1F8000005000020	2005	Ford	ch78984	Chevy	Styme	0	0
Call missing (Policy_id, Vin, Year, Make);		23 44879	1G1000003000021	2003	Chevrolet	ch78984	Chevy	Slyme	0	1
end		24 22407	1FA000003000022	2005	Ford	18+00+0 gu77892	William	Guppy	0	0
do until(eof1);										
set work.cust (keep=Policy id Cust id First name last name) end	eof1;									
re-ln find():	A Policy id	Vi Vi	n Year	Mak	ce 🕼 Cu	istic 🕢 Fi	inst name	Last Name	ball re bal	r
ic-ip.inid(),	1 81889	1FA0000030	2003	Ford	ch12489	Martin	C	uzlewit	0	-
* output work.loop_out;	2 12587	1G10000050	000011 2005	Chevrolet	do78521	Amy	De	rrit	0	1
If $(rc = 0)$ then do:	3 36901	JHG0000030	000012 2003	Honda	to12587	Leo	To	istoy	0	1
$\frac{1}{1}$	4 65547	1G10000050	000014 2005	Chevrolet	678902	Montag	jue Ti	19	0	1
output work.loop_out_unq;	5 65547	1G10000050	000024 2005	Chevrolet	678902	Monta	pue Ti	99	0	1
output work.loop out all: /*unique*/	6 65547	JHG0000030	000025 2003	Honda	ti78902	Monta	gue Ti	9	0	1
$1 \qquad 1 \qquad 1 \qquad 1 \qquad 1$	7 58749	1FA00000500	000016 2005	Ford	ti78902	Tigg	Ma	intague	0	1
Ip.nas_next (result: r);	8 22158	JHG0000030	000019 2003	Honda	ta48963	Mark	Ta	pley	0	1
do while (r ne 0);	9 44879	1G10000030	000021 2003	Chevrolet	ch78984	Chevy	S	me	0	1
re-ln find next():										
re-ip.inia_next(),	Ant		·- A *				irst na 🔥	Land March	00	

output work.loop\_out\_all; /\*non-unique\*/ output work.loop\_out\_dup; lp.has\_next (result: r); end;

end;

end;

```
lp.output(dataset: 'H2_dump');
```

stop;

	A Policy_id	\land vin	A Year	A Make	A Cust_id	A Hist na	A Last_Name	🕑 rc		r
1	11489	1FA00000030000001	2003	Ford	sh15878		Shakespeare	(	)	
2	98355	JHG000003000018	2003	Honda	sh15878	Shakespeare	1	(	)	0
3	22389	1G1000005000002	2005	Chevrolet	ma48798	Thomas	Mann	(	)	0
4	81889	JHG000003000003	2003	Honda	ch12489	Martin	Chuzlewit	(	)	0
5	12587	1FA00000050000004	2005	Ford	do78521	Amy	Dorrit	(	)	0
6	36901	1GC000003000005	2005	Chevrolet	to12587	Leo	Tolstoy	(	)	0
7	38525	JHG000003000006	2003	Honda	ga78521	Elizabeth	Gaskell	(	)	0
8	66381	1FB000005000007	2005	Ford	ni33691	Nicholas	Nickleby	(	)	0
9	22871	1G1000003000008	2003	Chevrolet	d11483	Arthur	Clennam	(	)	0
10	36999	JHG000005000009	2005	Honda	go78224	Oliver	Goldsmith	(	)	0
11	65547	1FA00000030000013	2003	Ford	ti78902	Montague	Tigg	(	)	0
12	58749	JHG000003000015	2003	Honda	ti78902	Tigg	Montague	(	)	0
13	22158	1GC0000030000017	2005	Chevrolet	ta48963	Mark	Tapley	(	)	0
14	44879	1FB00000050000020	2005	Ford	ch78984	Chevy	Slyme	(	)	0
15	22487	JHG000005000022	2005	Honda	ja48548	John	Jarndyce	(	)	0
16	14856	1FA00000030000023	2003	Ford	gu77892	William	Guppy	(	)	0

attrib	RROR_ Policy_id	First_nam	e Last_	Name	City	Vin Year	Make	Abbr_vin	Claim	_No
	φ 15 τ ( NL - 1	μ. .1 φ.15	· 1 ·	<u>μ</u>					1	
policy_id length= \$ 10 First_name length=	= \$ 15 Last_Name le	ngtn =  15	city lengi	tn =  13	)	10				
vin length= \$ 17 Year length= \$ 4 Make le	ength= \$ 17 Abbr_Vi	n length= \$	2 claim_	_no leng	th =	10;				
If $n_=1$ then do;										
dcl hash a(dataset: 'Cust(where=(city="Lo	ndon")',ordered:'Y'	);								
a.definekey('policy id');										
a definedata('first_name' 'last_name' 'city	<i>/</i> ')·	The do	loop ent	ters the	first va	lues into th	e PDV	: Policy_id, V	/in, and	
a definedone():	);	Abbr_	vin. the a	i.find me	ethod ı	uses the valu	ie in th	e PDV to retu	rieve	
a.definedone(),		First_n	ame, Las	st_name	e, and c	ity from ha	sh obj	ect a. The b.f.	ind meth	od
		uses a	concaten	ation of	value	s in the vin	numbe	r to match the	e values	of
dcl hash b(dataset: 'pol');		Abbr	vin the ke	ev varia	hle in	the h object	and th	e value of ve	ar and m	ake
b.definekey('Abbr_vin');		ore ret	ived Fir	olly the	o find	method use	and the l	zeve policy i	and vir	n to
<pre>b.definedata('year','make');</pre>					2 C.IIIIU		es une i	ceys policy_i		110
b.definedone();		retrive	the clain	n_no fro	om the	c object.				
dcl hash c(dataset:'claim'):										
c.definekev('policy_id' 'vin').										
c definedata('claim_no');										
e definedana();										
c.definedone(),										
		A policy i	d 💧 First_name	e 💧 Last_Na	me 💧 city	vin vin	A Year	💧 Make 💧 claim n	a rc 🚺	31,3 ( 31,d
		policy_ 1 11489	d 👌 First_name	e 🔌 Last_Na Stakespeare	me 🔌 city London	V 💩 vin 1FA0000030000001	2003	Make Actain Ford	0 a_rc	<b>b_rc ⊗ c_rc</b> 0 160038
call missing(policy_id, First_name, Last_name)	ne, city, vin, Year,	olicy_ 1 11489 2 22389	d 🔌 First_name	e 💩 Last_Na Shakespeare	me 🔌 city London	7 💩 vin 1F4000003000001 1G10000050000002	2003 2005	Anter Calaim_s Ford Chevrolet	0 a_rc 0 1 0 160038	b_rc <mark>⊗ c_rc</mark> 0 160038 0 160038
call missing(policy_id, First_name, Last_nar Make, claim_no);	ne, city, vin, Year,	policy_ 1 11489 2 22389 3 81889 3 81889	d First_name	e A Last_Na Stakespeare Chuzlewit	London	V Vin 1FA000003000001 1G1000005000002 JHG000003000003 F	2003 2005 2003	Make Claim_s Ford Chevrolet Honda 7141	0 3_rc 0 160038	b_rc  c_rc 0 160038 0 160038 0 0 0
call missing(policy_id, First_name, Last_name, Make, claim_no);	ne, city, vin, Year,	policy     1     11489     2     22389     3     81889     4     12587     x     36901	d Arry	e A Last_Na Shakespeare Chuzlewit Domit	London London London London	Vie vie 154000000000001 1610000050000002 JHG000000000000000000000000000000000000	2003 2005 2005 2005 2005 2003	Make Sclaim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet	0 a_rc 0 150038	b_nc 3 c_nc 0 160038 0 160038 0 0 0 0
call missing(policy_id, First_name, Last_name, Make, claim_no);	ne, city, vin, Year,	<ul> <li>policy_j</li> <li>1 11489</li> <li>2 22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36901</li> <li>5 36505</li> </ul>	d First_name Marin Any	e 🔌 Last_Na Shakespeare Chuzlewit Domit	me 🔌 city Landon Landon Landon	Keiner Construction	2003 2005 2005 2005 2005 2005 2003 2003	Make Claim_s Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda	0 a_rc 0 0 0 160038 0 0 160038 160038	b_nc 3 c_nc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038 0 160038
call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (cof2);	ne, city, vin, Year,	<ul> <li>policy_j</li> <li>1</li> <li>11489</li> <li>2</li> <li>22389</li> <li>3</li> <li>81889</li> <li>4</li> <li>12587</li> <li>5</li> <li>36901</li> <li>6</li> <li>30525</li> <li>7</li> <li>66381</li> </ul>	d First_name Marin Amy Nicholas	e 💩 Last_Na Shakespeare Chuzlewit Domit Nickleby	me 💩 city London London London	via           1FA0000030000001           1G10000050000002           JHG0000050000003           1FA0000050000004           1GC000000000005           JHG0000030000005           JHG0000030000005           JHG00000000000005           JHG000000000000005	2003 2005 2005 2003 2005 2003 2003 2003	Make Claim r Ford Chevrolet Honda 7141 Ford 7243 Chevrolet Honda Ford	0 a_rc 0 0 160038 0 0 160038 160038 0	b_rc         c_rc           0         160038           0         160038           0         0           0         0           0         160038           0         160038           0         160038           0         160038           0         160038
call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2);	ne, city, vin, Year,	<ul> <li>policy_j</li> <li>1 11489</li> <li>2 22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36901</li> <li>6 38525</li> <li>7 66381</li> <li>8 22871</li> </ul>	d i First_name Marin Any Nicholas Arbur	e Ast_Na Shakespeare Chuziewit Domit Nickleby Clennam	me (a) city London London London London London	Via           1F40000030000001           1G10000050000001           1G10000050000003           JHG0000030000003           1F40000050000004           1G20000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005	Image: Weak of the second se	Make claim r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7581	0 3 a_rc 0 160038 0 160038 160038 0 0 0 0 0	b_re         c_re           0         160038           0         160038           0         0           0         0           0         0           0         160038           0         160038           0         160038           0         160038           0         160038           0         0
call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo	ne, city, vin, Year, f2;	<ul> <li>policy_j</li> <li>1 11489</li> <li>2 22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36801</li> <li>6 36525</li> <li>7 66361</li> <li>8 22871</li> <li>9 36999</li> <li>9 36999</li> </ul>	d First_name Martin Amy Nicholas Arthur Oliver	e A Last_Na Shakespeare Chuzlewit Domit Nickleby Clennam Goldsmith	me 💩 city London London London London London	Via           1F4000003000001           1G10000050000001           1G10000050000003           JHG0000050000004           1G20000030000005           JHG0000030000005           JHG0000050000007           1G10000050000007           1G10000030000008           JHG00000000000000	Image: Wear         Year           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2003         2005           2005         2005	Make claim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7681 Honda	0 3 a_rc 0 0 160038 0 0 160038 160038 0 0 0 0 0 0	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160058 0 160058
call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find();	ne, city, vin, Year, f2;	<ul> <li>policy_j</li> <li>1 11489</li> <li>2 22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36801</li> <li>6 38525</li> <li>7 66381</li> <li>8 22871</li> <li>9 36899</li> <li>10 81889</li> <li>12887</li> </ul>	d First_name Martin Arty Nicholas Arthur Oliver Martin Artu	e Alast_Na Shakaspeare Chuzlewit Domit Nickleby Clennam Goldsmith Chuzlewit Dowit	me 🔌 city Landon Landon Landon Landon Landon Landon	Vie     1F#2000003000001     1F#2000003000000     JHG2000005000002     JHG2000005000000     1F#20000030000005     JHG20000030000005     1F800000030000005     1F800000030000008     JHG20000050000001     1F#20000030000001	Year 2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005	Make claim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7681 Honda Ford Chevrolet	0 3 a_rc 0 160038 0 160038 160038 0 0 0 0 0 0 0 0 0 0	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2);    set work.Pol (keep=Policy_id vin) end=eo    a_rc=a.find();    if a_rc ne 0 then call missing(First_name,</pre>	ne, city, vin, Year, f2; last_name, city);	<ul> <li>policy_1</li> <li>11489</li> <li>22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36901</li> <li>6 38525</li> <li>7 66381</li> <li>8 22871</li> <li>9 36999</li> <li>10 81889</li> <li>11 12587</li> <li>12 36801</li> </ul>	d First_name Martin Amy Nicholas Arthur Oliver Martin Amy	e Alast_Na Shakespeare Chuzlewit Donit Nickleby Clennam Goldsmith Chuzlewit Donit	me () city Landon Landon Landon Landon Landon Landon Landon Landon	Vie     1F#2000003000001      1F#2000003000000      JHG000003000003      1F#2000003000003      1F#2000003000005      JHG000003000005      1F8000003000005      JHG000003000005      JHG00000300005      JHG0000300005      JHG000030005      JHG00003005      JHG00003005      JHG00003005      JHG00003005      JHG00003005      JHG00003005      JHG00003      JHG000030005      JHG00003      JHG00003      JHG00003      JHG000	2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003	Make Claim r      Ford     Chevrolet     Honda 7141     Ford 7249     Chevrolet     Honda     Ford     Chevrolet 7681     Honda     Ford     Chevrolet     Honda	0 3 arc 9 1 160038 0 160038 0 160038 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),substr(vin, 2,1))); substr(vin, 2,1)); substr(vin, 2,1); substr(vin, 2,</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1)));	<ul> <li>policy_j</li> <li>1</li> <li>11489</li> <li>2</li> <li>22389</li> <li>3</li> <li>31889</li> <li>4</li> <li>12587</li> <li>5</li> <li>36901</li> <li>6</li> <li>36525</li> <li>7</li> <li>66381</li> <li>8</li> <li>22871</li> <li>9</li> <li>36999</li> <li>10</li> <li>81889</li> <li>11</li> <li>12587</li> <li>12</li> <li>36901</li> <li>13</li> <li>65547</li> </ul>	d First_name Martin Any Nocholas Arthur Oliver Martin Any Montague	e Aust_Na Shakespeare Chuzlewit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg	me () city London London London London London London London	Konstantia      Konstantia      Konstantia	2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005	Make claim_s Ford Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Ford Chevrolet Honda Ford Ford Ford Chevrolet Honda Ford Ford Ford Ford Ford Ford Ford Ford	o a_rc 0 160038 0 160038 160038 0 0 0 0 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re         c_rc           0         160038           0         160038           0         0           0         0           0         0           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sub- if b_rc ne 0 then call missing (Year, make)</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1)));	<ul> <li>policy_1</li> <li>11489</li> <li>22389</li> <li>381889</li> <li>412587</li> <li>536901</li> <li>638525</li> <li>766381</li> <li>822871</li> <li>936999</li> <li>1081889</li> <li>112587</li> <li>1236547</li> <li>1465547</li> </ul>	d First_name Martin Amy Nicholas Arthur Oliver Martin Amy Montague Montague	e Ast, Na Shakespeare Chuzlewit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg Tigg	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Via           1F#40000030000001           1G10000050000002           JHG0000030000003           1F#40000050000004           1G20000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000005           JHG0000030000001           JHG0000030000010           JHG0000030000011           JHG0000030000012           JHG000003000013           JG1000003000013	2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2005	Make claim r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7681 Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda	o a_rc 0 160038 0 160038 160038 0 0 0 0 0 160038 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re         c_rc           0         160038           0         160038           0         0           0         0           0         0           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038           0         160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sub if b_rc ne 0 then call missing (Year, make) c_rc=c find(key: policy_id_key: vin);</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1)));	<ul> <li>policy_j</li> <li>1 11489</li> <li>2 22389</li> <li>3 81889</li> <li>4 12587</li> <li>5 36801</li> <li>6 38525</li> <li>7 66381</li> <li>8 22871</li> <li>9 36999</li> <li>10 81889</li> <li>11 12587</li> <li>12 36901</li> <li>13 65547</li> <li>14 65547</li> <li>15 58749</li> </ul>	d First_name Martin Arty Nicholas Arthur Oliver Martin Arty Mortague Mortague Tigg	e A Last_Na Shakaspeare Chuzlewit Domit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg Tigg Montague	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon	Via           1F#2000003000001           1G10000050000002           JHG20000030000003           1F#20000050000004           1G20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000000           JHG20000050000010           JHG20000050000010           JHG20000030000010           JHG20000030000010           JHG20000030000012           JHG20000030000013           JHG20000030000014           JHG20000030000014           JHG2000003000015	2003 2005 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005	Make claim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7681 Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda	o a_rc 0 0 160038 0 0 160038 160038 0 0 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re         c_rc           0         160038           0         160038           0         0           0         0           0         0           0         160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sull if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if a_ra ne 0 then call missing(claim ne);</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); o; <b>claim_no</b>	<ul> <li>policy_1</li> <li>11489</li> <li>222089</li> <li>381889</li> <li>412587</li> <li>536901</li> <li>638525</li> <li>766381</li> <li>822871</li> <li>93699</li> <li>1081889</li> <li>1112587</li> <li>129601</li> <li>1365647</li> <li>1465647</li> <li>1568749</li> <li>1658749</li> <li>1658749</li> </ul>	d First_name Martin Arty Nicholas Arthur Oliver Martin Arty Mortague Mortague Tigg Tigg	e A Last_Na Shakaspeare Chuzlewit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg Tigg Tigg Montague Montague	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon		2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003	Make claim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda Ford Chevrolet 7581 Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda Ford Chevrolet Honda	0 3 a_rc 0 160038 0 160038 160038 0 0 0 0 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sub if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no);</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); ); 1 7681	<ul> <li>policy_1</li> <li>11489</li> <li>22389</li> <li>381889</li> <li>12587</li> <li>36901</li> <li>38525</li> <li>66381</li> <li>22871</li> <li>36999</li> <li>81889</li> <li>123687</li> <li>1236801</li> <li>136547</li> <li>146547</li> <li>1558749</li> <li>1658749</li> <li>1658749</li> <li>1722158</li> <li>18255</li> </ul>	d First_name Martin Any Nicholas Arbu Oliver Martin Any Montague Togg Togg Togg Mark Stalesceare	e Atast_Na Shakespeare Chuzlewit Donit Donit Chuzlewit Donit Chuzlewit Donit Tigg Tigg Montague Nontague Tapley	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Vie     1F#2000003000001      1F#2000003000000      1F#2000003000003      1F#2000003000005      JHG2000003000007      1F#2000003000001      1F#2000003000011      JHG2000003000012      1F#2000003000011      JHG2000003000011      JHG2000003000015      1F#2000003000015      1F#20000030000015      1F#2000003000015      1F#2000013000015      1F#200001      1F#200001      1F#200000      1F#200001      1F#200001      1F#20      1F#20000	2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003	Make claim_s      Ford      Chevnolet      Honda      Trai      Ford      Chevnolet      Honda      T887      Ford      Chevnolet      Ford      Ford      Chevnolet      Ford      Ford	0 3 a_rc 0 150038 0 0 150038 150038 150038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sult if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no); c.output(dataset:'cl');</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); ); 1 7681 2 7249	<ul> <li>policy_1</li> <li>11489</li> <li>22389</li> <li>381889</li> <li>412587</li> <li>536901</li> <li>638525</li> <li>66381</li> <li>822871</li> <li>36999</li> <li>1081889</li> <li>112587</li> <li>1236901</li> <li>1365547</li> <li>1465547</li> <li>1558749</li> <li>1658749</li> <li>1658749</li> <li>1658749</li> <li>1658749</li> <li>1658749</li> <li>1722158</li> <li>18 90355</li> <li>19 22158</li> </ul>	d First_name Martin Arty Nicholas Arthur Oliver Martin Arty Montague Togo Togo Togo Mark Shakespeare Mark	e Atast_Na Shakespeare Chuzlewit Donit Donit Nickleby Clennam Goldsmith Chuzlewit Donit Donit Tigg Tigg Montague Montague Tapley Tapley	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Vie           1F#2000003000001           1F40000030000001           1G10000050000003           JHG000003000003           1F40000030000004           1GC0000030000005           JHG0000030000005           JHG0000030000007           1G10000030000008           JHG00000000000000           JHG0000000000000           JHG0000000000000           JHG0000000000000           JHG000000000000001           JHG000000000000000000000000000000000000	2003 2005 2005	Make claim r      Ford     Ford     Chevnolet     Honda 7141     Ford 7243     Chevnolet     Honda     Ford     Chevnolet 7881     Honda     Ford     Chevnolet 8     Honda     Ford     Chevnolet     Honda     T897     Honda	a_rc 3	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sult if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no); c.output(dataset:'cl'); output work.mg;</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); ; 1 7681 2 7249 3 8113	<ul> <li>policy_</li> <li>1</li> <li>11489</li> <li>2</li> <li>22389</li> <li>3</li> <li>3</li> <li>3</li> <li>3</li> <li>4</li> <li>12587</li> <li>5</li> <li>36901</li> <li>6</li> <li>36525</li> <li>7</li> <li>66381</li> <li>8</li> <li>22871</li> <li>9</li> <li>36939</li> <li>10</li> <li>81889</li> <li>11</li> <li>12587</li> <li>12</li> <li>36901</li> <li>13</li> <li>65547</li> <li>14</li> <li>65547</li> <li>15</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>17</li> <li>22158</li> <li>18</li> <li>9055</li> <li>19</li> <li>22158</li> <li>20</li> <li>44879</li> </ul>	d First_name Martin Arty Nicholas Arthur Oliver Martin Arty Mortague Mortague Tigg Tigg Tigg Mark Shakespeare Mark Chevy	e A Last_Na Shakaspeare Chuzlewit Domit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg Tigg Nortague Nortague Tapley Slyme	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Via           1F#2000003000001           1G10000050000002           JHG20000030000003           1F#20000050000004           1G20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000005           JHG20000030000001           JHG20000030000010           JHG20000030000010           JHG20000030000011           JHG20000030000012           1F#20000030000013           1G10000050000014           JHG2000003000015           1F400000300001015           JHG200000300001015           JHG200000300000101           JHG2000003000001015           JHG2000003000001015           JHG2000003000001017           JHG2000003000001017           JHG2000003000001017           JHG20000030000018           JHG20000030000019           JHG20000030000019           JHG20000030000019           JHG20000030000019	2003 2005 2005	Make Claim, s      Ford      Chevrolet Honda     7141 Ford     7243 Chevrolet Honda Ford Ford Chevrolet Honda Ford Ford Chevrolet Honda Ford Ford Ford Ford Ford Ford Ford Ford	0 3 a_rc 0 160038 0 160038 160038 0 0 0 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re         c_rc           0         160038           0         160038           0         0           0         0           0         0           0         0           0         160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sul if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no); c.output(dataset:'cl'); output work.mg; /*If (C_rc=0) then output work.mg;*/</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); ); 1 7681 2 7249 3 8113 4 7141	<ul> <li>policy_</li> <li>1</li> <li>11489</li> <li>2</li> <li>22089</li> <li>3</li> <li>81889</li> <li>4</li> <li>12587</li> <li>5</li> <li>36001</li> <li>6</li> <li>38525</li> <li>7</li> <li>66381</li> <li>8</li> <li>22871</li> <li>9</li> <li>36899</li> <li>10</li> <li>81889</li> <li>11</li> <li>12587</li> <li>12</li> <li>36901</li> <li>13</li> <li>65547</li> <li>14</li> <li>65547</li> <li>15</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>17</li> <li>22158</li> <li>18</li> <li>90555</li> <li>19</li> <li>22158</li> <li>20</li> <li>44879</li> <li>21</li> <li>44879</li> <li>21</li> <li>44879</li> </ul>	d First_name Martin Acty Nicholas Acthur Oliver Oliver Martin Amy Montague Tigg Mark Stalespeare Mark Chevy Chevy Chevy	e A Last_Na Shakaspeare Chuzlewit Domit Domit Nickleby Clennam Goldsmith Chuzlewit Domit Tigg Tigg Montague Tagley Tagley Slyme Slyme	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Viiii           1F#2000003000001           1G1000005000002           JHG2000005000002           JHG2000055000004           1G1000005000005           JHG20000030000005           JHG20000050000005           JHG20000050000007           1G10000050000001           JHG20000050000010           JHG20000050000010           JHG20000050000011           JHG20000050000012           1F#40000030000012           1F40000050000014           JHG20000050000015           1F40000050000016           JHG20000000000017           JHG200000000000018           JHG20000000000018           JHG200000000000019           1FE00000050000018           JHG20000000000000019           JHG2000000000000000018           JHG20000000000000000018           JHG20000000000000000019           JHG20000000000000000019           JHG2000000000000000019           JHG200000000000000000000000000019           JHG2000000000000000000000000000000000000	2003 2005 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2005	Make claim_r Ford Chevrolet Honda 7141 Ford 7249 Chevrolet Honda 7249 Chevrolet 7681 Honda 7681 Honda 7681 Honda 7681 Chevrolet 7681 Honda 7681 Chevrolet 7681 Honda 7897 Honda 7897 Honda 7897 Honda 7897	0 3 a_rc 0 160038 0 160038 160038 0 0 0 0 0 0 160038 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re         c_rc           0         160038           0         160038           0         0           0         0           0         0           0         0           0         160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sul) if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no); c.output(dataset:'cl'); output work.mg; /*If (C_rc=0) then output work.mg;*/ end:</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); );	<ul> <li>policy_</li> <li>1</li> <li>11489</li> <li>2</li> <li>22389</li> <li>3</li> <li>81889</li> <li>4</li> <li>12587</li> <li>5</li> <li>36901</li> <li>6</li> <li>38525</li> <li>7</li> <li>66381</li> <li>8</li> <li>22871</li> <li>9</li> <li>36999</li> <li>10</li> <li>81889</li> <li>11</li> <li>12587</li> <li>12</li> <li>36901</li> <li>13</li> <li>65547</li> <li>14</li> <li>65547</li> <li>15</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>16</li> <li>58749</li> <li>17</li> <li>22158</li> <li>20</li> <li>44879</li> <li>21</li> <li>44879</li> <li>22</li> <li>2487</li> <li>24</li> </ul>	d First_name Martin Any Nicholas Arbur Oliver Martin Any Mortague Mortague Tigg Tigg Mark Shakespeare Mark Chevy Chevy	e A Last_Na Shakaspeare Chuzlewit Donit Nickleby Clennam Goldsmith Chuzlewit Donit Donit Tigg Tigg Montague Tapley Sigme Sigme	me & city Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon Landon	Via           1F#2000003000001           1G1000005000002           JHG2000005000003           1F#2000005000004           1F#2000005000004           1G1000005000000           1F#2000005000000           JHG2000005000000           JHG2000005000000           1F#20000050000000           JHG20000050000001           1G10000050000010           1G10000050000011           JHG20000050000011           JHG20000050000011           JHG20000050000011           JHG20000050000015           1F#2000000000000016           JHG20000000000000017           JHG2000000000000000018           JHG200000000000019           JHG20000000000000001           JHG2000000000000000000000000001           JHG20000000000000000000000000000000001           JHG2000000000000000000000000000000000000	2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003	Make Claim e Ford Ford Chevnolet Honda Ford Chevnolet Ford Ford Chevnolet Ford Ford Ford Ford Ford Ford Ford Ford	a_rc         0           160038         0           0         160038           160038         0           0         0	b_re 3 c_rc 0 160038 0 160038 0 0 0 0 0 0 0 160038 0 160038
<pre>call missing(policy_id, First_name, Last_name, Make, claim_no); end; do until (eof2); set work.Pol (keep=Policy_id vin) end=eo a_rc=a.find(); if a_rc ne 0 then call missing(First_name, b_rc=b.find(key: cats((substr(vin, 2,1)),sult if b_rc ne 0 then call missing (Year, make) c_rc=c.find(key: policy_id, key: vin); if c_rc ne 0 then call missing(claim_no); c.output(dataset:'cl'); output work.mg; /*If (C_rc=0) then output work.mg;*/ end; Stop:</pre>	ne, city, vin, Year, f2; last_name, city); bstr(vin,10,1))); );	policy_ 1 [1489 2 22389 3 81889 4 12587 5 36901 6 30525 7 66381 8 22871 9 36999 10 81889 10 81889 10 81889 11 12587 12 36901 13 65547 14 65547 15 58749 16 58749 16 58749 16 58749 17 22158 18 98355 19 22158 20 44879 21 44879 21 44879 22 44879 24 65547	d First_name Martin Any Nicholas Arbu Oliver Martin Any Oliver Martin Any Montague Togg Togg Mark Shakespeare Mark Chevy Chevy Villiam Montague	e Atast_Na Shakespeare Chuzlewit Donit Donit Donit Chuzlewit Donit Chuzlewit Donit Donit Tigg Tigg Montague Tagley Tagley Styme Styme Styme Ton	me & city Landon		2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005 2003 2005	Make claim_s      Ford      Chevrolet Honda     7141 Ford     7249 Chevrolet Honda Ford Chevrolet Ford Chevrolet Honda Ford Chevrolet Ford Ford Chevrolet Ford Ford Chevrolet Ford Ford Ford Ford Ford Ford Ford Ford	0 0 150038 0 0 150038 0 0 0 150038 150038 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b_re

ARGUMENT TAGS (Dataset: dataset: name<(options)>') Is a very convenient way to load a hash object. A do loop is another alternative.

(Duplicate: 'R' | 'E') Determines how duplicates are delt with. The 'R' replaces the first data item and the 'E' stops the program when a duplicate is detected.

(Multidata: 'Y' | 'Yes') Maps multiple data items to a key item

(Ordered: 'A' | 'Y' | 'D' | 'N') Sorts the data using the key items like the by in a proc sort. An efficient feature if one may need sorted data later. It can make a proc sort unnessary.

#### (hashexp: n)

Sets the number of hash slots or buckets. n is an exponent of 2. For 9.3 the default for the number of hash buckets is 8 or 256 slots. To work efficiently the key items must not be distributed too densely or too thinly in the slots. 9 (512 slots) or 10 (1024 slorts) is recommended for one million items.

Calculates the number of slots:

data hashexp_arg;	
format exponent 4.;	
buckets comma10.;	Check size of memory
do exponent = $0$ to $20$ ;	•
buckets = $2^{**}$ exponent;	data _null_;
output;	amt=getoption('xmrlmem')
end;	put amt=;
run;	run;

(Suminc: 'variable-name') Use as a counter. The variable-name keeps a count in the PDV.

The definedata method has one argument tag: ALL:'Y'. It will define all the data loaded into the hash object as data items. When you use the definedata method if you want the key data ouput it must be defined as a key and as a data item.

The object.definedone is used to finalize the intialization of the hash object. It has one agrument tag: (Memrc: 'y'). It is used to evaluate whether the method was successful or not. A return code of 0 indicates that the hash object was loaded and a non-zero indicates failure, usually because of insuffiecient memory. The non-zero code can be used with the object.delete method to remove the hash object.

```
%let dsn1=work.cust;
%let out_dsn1=Mro_varlist;
%let dsn_cmp=work.pol;
proc contents data=&dsn1
    out=work.&out_dsn1 (keep=libname memname name type length nobs varnum)
    noprint;
run;
proc sort data=work.&out_dsn1;
    by varnum;
run;
/*dataset for the hash object*/
proc sql noprint inobs=1;
    select libname, memname
    into :H_lib, :H_dsn
    from &out_dsn1;
                                                  39
                                                                %put &H lib &H dsn;
quit;
                                                  WORK
                                                             CUST
%put &H_lib &H_dsn;
```

/\* &Attr\_mv Attributes \*\* &Key\_mv Key variables \*\*\*\* &Data\_mv Data variables \*\*\* &Miss\_mv Call missing variables \*\*\*\*\*//\*Attrib macro variable\*/

```
data H_mac_v(keep=name Key_Item attr varnum R_name re_name);
set work.&out_dsn1;
    if type = 2 then A_type="$";
    else if type = 1 then A_type="";
    Attr=Catx("",name,"length=",A_type,length);
    Key_Item=quote(compress(name));
    R_name=cats("R_",name);
    Re_name=catx(" ",name,"=",R_name);
    where varnum in (2 3 4 5 6);
```

```
proc sql noprint;
    select attr, name, Re_name
       into :Attr_mv separated by "",
         :Keep1_mv separated by "",
           :Re name separated by ""
       from H mac v;
       *where varnum in (1 2 3 4 5 6);
       where varnum between 2 and 6; /*select varnum for attrib*/
quit;
%put &Attr_mv;
%put &Keep1_mv;
%put &Re_name;
64
            Sput &Attr mv;
policy id length= $ 10 First name length= $ 15 Last Name length= $ 15 city length= $ 15 Country length= $ 15
65
            %put &Keep1 mv;
policy id First name Last Name city Country
66
            %put &Re name;
policy id = R policy id First name = R First name Last Name = R Last Name city = R city Country = R Country
```

```
/*******KEY Macro variable*********/

proc sql noprint;

select Key_Item

into :Key_mv separated by ","

from H_mac_v

where varnum in (2); /*select varnum for key item*/
```

quit;

%put &Key\_mv;

71 %put &Key\_mv;
"policy id"

/\*\*\*\*\*\*\*\*\*\*\*\*Data Macro variable and Call missing \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

proc sql noprint;

select Key\_Item as D\_Item,name
into :Data\_mv separated by ",",
:Miss\_mv separated by ","
from H\_mac\_v
/\*where varnum in (1); /\*select varnum for data item(s) and call missing\*/
where varnum between 2 and 7;\*/

quit;

%put &Data\_mv;%put &Data\_mv;%put &Miss\_mv;%put &Miss\_mv;86%put &Miss\_mv;policy\_id,First\_name,Last\_Name,city,Country

data work.examp&H\_dsn;

\*if \_n\_=0 then set work.&H\_dsn(keep=&keep1\_mv); attrib &Attr\_mv;

if \_n\_=1 then do;

dcl hash exp(dataset:"work.&H\_dsn"); exp.definekey (&Key\_mv); exp.definedata (&Data\_mv); exp.definedone(); call missing(&miss\_mv); end; set &dsn\_cmp; rc=exp.find(key: policy\_id); if rc=0 then exp.output(dataset:'DumpH'); if rc=0 then output work.examp&H\_dsn;

#### REFERENCES

Bloom, Janice and Secosky, Jason, <u>Getting Started with the DATA Step Hash Iterator</u>, Paper 271-2007, SAS Global Forum 2007.
Burlew, Michele M. <u>SAS® Hash Object Programming Made Easy</u>, Cary, NC: SAS Institute, 2012.
Carpenter, Art, <u>Carpenter's Guide to Innovative SAS Techniques</u>, Cary, N: SAS Institute, Inc., 2012.
Dorfman, Paul and Shajenko, Lessia S., <u>Hash Crash and Beyond</u> Paper 037-2008 SAS Global Forum 2008.
Dorfman, Paul and Vyverman, Koen, <u>The SAS Hash Object in Action</u>, Paper 153-2009 SAS Global Forum 2009.
Dorfman, Paul and Vyverman, Koen, <u>Data Step Hash Objects as Programming Tools</u>, Paper 241-31, SUGI 31.
Dorfman, Paul, Vymerman, Koen, and Dorfman, Victor, <u>Black Belt Hashigana</u>, Paper 023-2010, SAS Global Forum 2010.
Hinson, Joseph <u>The Hash-of-Hashes as a "Russian Doll" Structure: An Example with XML Creation</u>, Paper 021-2013, SAS Global Forum 2013,
Loren, Judy <u>How Do I Love Hash Tables? Let Me Count the Ways!</u>, Paper 029-2008 SAS Global Forum 2008.
Ray, Robert and Secosky, Jason, <u>Better Hashing in SAS® 9.2</u>, Paper 306 2008.
SAS Community <u>Hash Object Resources</u>
SAS Institute Inc., <u>SAS® 9.3 Component Objects Reference</u>, 2011.
Secosky, Jason and Janice Bloom, <u>Getting Started with the DATA Step Hash Object</u>, Paper 271-2007, SAS Global Forum 2007.

## Endnotes

<sup>1</sup> The Hash Object and the Hash Iterator Object first appeared in SAS 9.0. In later versions of SAS, the Java, Logger, and Appender Objects were added and the hash and hash iterator objects were greatly improved.

<sup>2</sup> Romeo and Juliet, Act II, Scene 2, line 43.

data match ;

```
3 set small point = _n_ ; * get key/data attributes for parameter type matching ;
* set small (obs = 1) ; * this will work, too :-)! ;
* if 0 then set small ; * and so will this :-)! ;
* set small (obs = 0) ; * but for some reason, this will not :-( ;
```

Paul Dorfman Hash Crash and Beyond p. 2

Contact information:

Philip Burtch burtch@mac.com or philip.burtch@usaa.com



Data work.cust; infile datalines dsd; input Cust\_id: \$10. policy\_id: \$10. First\_name: \$15. Last\_Name: \$15. city: \$15. Country : \$15.;

datalines;

pe12489,81880,Seth,Pecksniff,London,Eng sh15878,11489,,Shakespeare,London,Eng sh15878,98355,Shakespeare,,London,Eng ma48798,22389,Thomas,Mann,Princeton,USA ch12489,81889,Martin,Chuzlewit,London,Eng do78521,12587,Amy,Dorrit,London,Eng to12587,36901,Leo,Tolstoy,Yasnaya Polyana,Russia ga78521,38525,Elizabeth,Gaskell,Knutsford,Eng ni33691,66381,Nicholas,Nickleby,London,Eng cl11483,22871,Arthur,Clennam,London,Eng go78224,36999,Oliver,Goldsmith,London,Eng ti78902,65547,Montague,Tigg,London,Eng ti78902,58749,Tigg,Montague,London,Eng ta48963,22158,Mark,Tapley,London,Eng ch78984,44879,Chevy,Slyme,London,Eng ja48548,22487,John,Jarndyce,Bleak House,Eng gu77892,14856,William,Guppy,London,Eng

run;

#### Data

Data work.Pol: infile datalines dsd; input Policy id:\$10. vin: \$17. Year: \$4. Make: \$17. Abbr Vin \$2.; datalines: 11489,1FA0000003000001,2003,Ford,F3 22389,1G10000005000002,2005,Chevrolet,G5 81889,JHG0000003000003,2003,Honda,H3 12587,1FA0000005000004,2005,Ford,F5 36901,1GC0000003000005,2005,Chevrolet,G5 38525,JHG0000003000006,2003,Honda,H3 66381,1FB0000005000007,2005,Ford,F5 22871,1G1000003000008,2003,Chevrolet,G3 36999,JHG0000005000009,2005,Honda,H5 81889,1FA00000030000010,2003,Ford,F3 12587,1G10000050000011,2005,Chevrolet,G5 36901,JHG00000030000012,2003,Honda,H3 65547.1FA00000030000013.2003.Ford.F3 65547,1G100000050000014,2005,Chevrolet,G5 58749,JHG00000030000015,2003,Honda,H3 58749,1FA00000050000016,2005,Ford,F5 22158,1GC00000030000017,2005,Chevrolet,G3 98355,JHG00000030000018,2003,Honda,H3 22158.JHG00000030000019.2003.Honda.H3 44879,1FB0000005000020,2005,Ford,F5 44879,1G1000003000021,2003,Chevrolet,G3 22487,JHG00000050000022,2005,Honda,H5 14856.1FA0000003000023.2003.Ford.F3 65547,1G100000050000024,2005,Chevrolet,G5 65547,JHG0000003000025,2003,Honda,H3

data work.claim; infile datalines dsd; input Policy\_id:\$10. claim\_no: \$10. Vin: \$17.; datalines; 81889,7141,JHG00000030000003 12587,7249,1FA00000050000004 22871,7681,1G100000030000008 98355,7897,JHG00000030000018 36901,8113,JHG0000003000005 ; run;