PhUSE 2016 Paper CC08

Perish the Sort: Using Indexes and Hash Objects for Efficient Programming

Jason A Smith
Argo Analytics Ltd

Contents

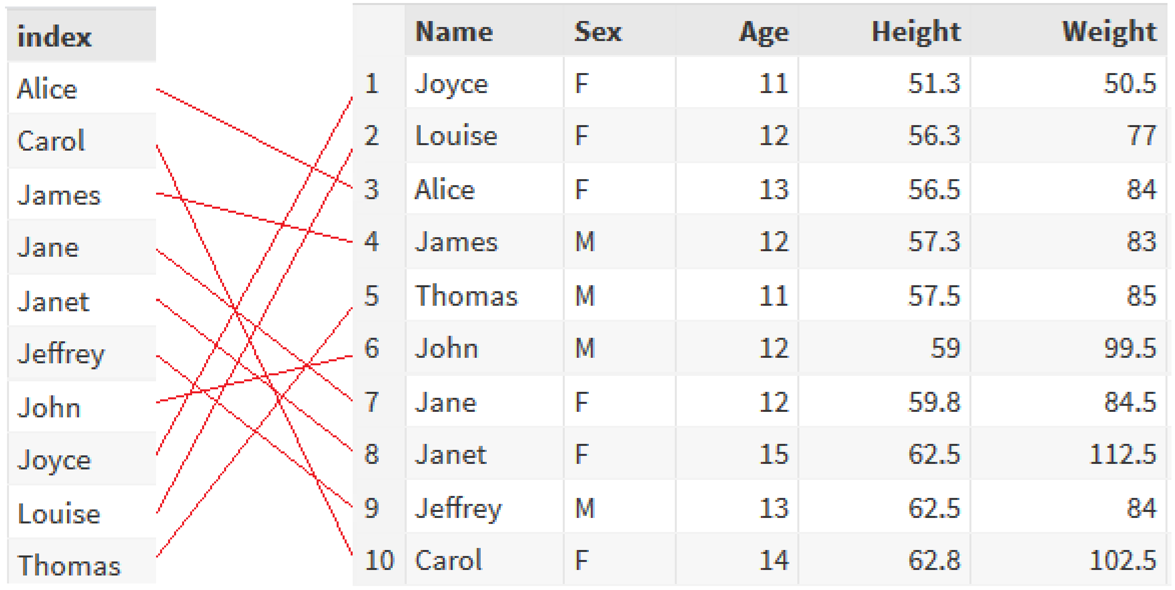
- Indexes to combine or classify data
  - What is an index?
  - Simple index
  - Composite index
  - Multiple and unique indexes
- Hash objects to combine data
  - What is a hash object?
  - Combining unsorted datasets

What is an index?

- an index is a special companion file containing the values and record numbers of the indexed variables:

Classfit Index

Classfit Dataset



index
Alice
Carol
James
Jane
Janet
Jeffrey
John
Joyce
Louise
Thomas

	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Louise	F	12	56.3	77
3	Alice	F	13	56.5	84
4	James	M	12	57.3	83
5	Thomas	M	11	57.5	85
6	John	M	12	59	99.5
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	Carol	F	14	62.8	102.5

Index syntax

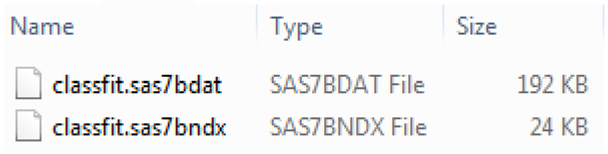
- to create an index on a dataset:

```
data dataset (index=(index-specification-1</unique>
                     index-specification-2</unique>));

    ...your code here

run;
```

- to display index usage information in SAS log:

```
options msglevel=i;
```

- can view in explorer:



Name	Type	Size
classfit.sas7bdat	SAS7BDAT File	192 KB
classfit.sas7bndx	SAS7BNDX File	24 KB

Simple index

- try to merge datasets that are not sorted or ordered:

```
data class;
  merge sashelp.class classfit;
  by name;
run;
```

- data step runs with errors:

```
ERROR: BY variables are not properly sorted on data set WORK.CLASSFIT.
Name=Louise Sex=F Age=12 Height=56.3 Weight=77 predict=76.488485693
lowermean=67.960050237 uppermean=85.016921149 lower=51.314521735
upper=101.66244965 FIRST.Name=1 LAST.Name=1 _ERROR_=1 _N_=13
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 14 observations read from the data set SASHELP.CLASS.
NOTE: There were 3 observations read from the data set WORK.CLASSFIT.
WARNING: The data set WORK.CLASS may be incomplete.  When this step was stopped
         there were 12 observations and 10 variables.
WARNING: Data set WORK.CLASS was not replaced because this step was stopped.
```

Simple index

- define simple index NAME:

```
data classfit (index=(name));
  set sashelp.classfit;
run;
```

- log confirms that the index has been defined:

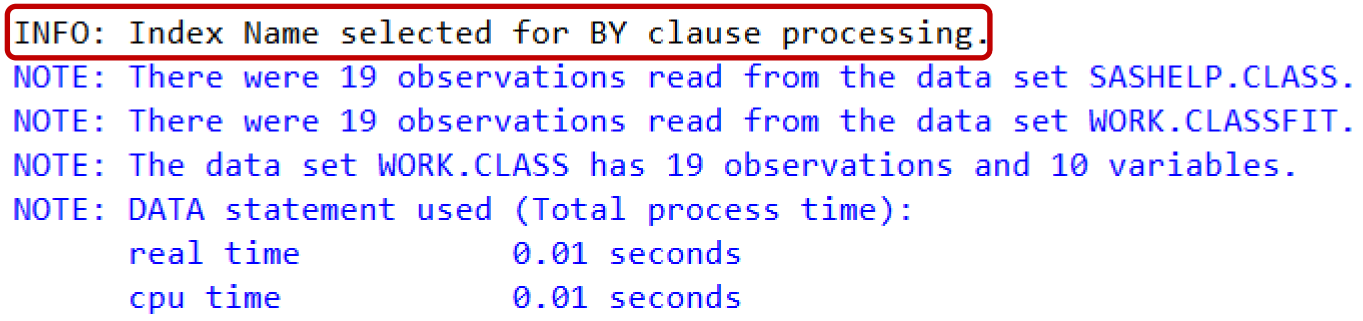
```
NOTE: There were 19 observations read from the data set SASHELP.CLASSFIT.
NOTE: The data set WORK.CLASSFIT has 19 observations and 10 variables.
NOTE: Simple index name has been defined.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Simple index

- data can now be used with a BY statement without the need for the dataset to be sorted:

```
data class;
  merge sashelp.class classfit;
  by name;
run;
```

- log confirms that the index has been used, no SAS errors:



```
INFO: Index Name selected for BY clause processing.
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: There were 19 observations read from the data set WORK.CLASSFIT.
NOTE: The data set WORK.CLASS has 19 observations and 10 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Composite index

- define composite index on variables Sex and Age:

```
data class (index=(sexage=(sex age)));
   set sashelp.class;
run;
```

- log confirms that the composite index has been defined:

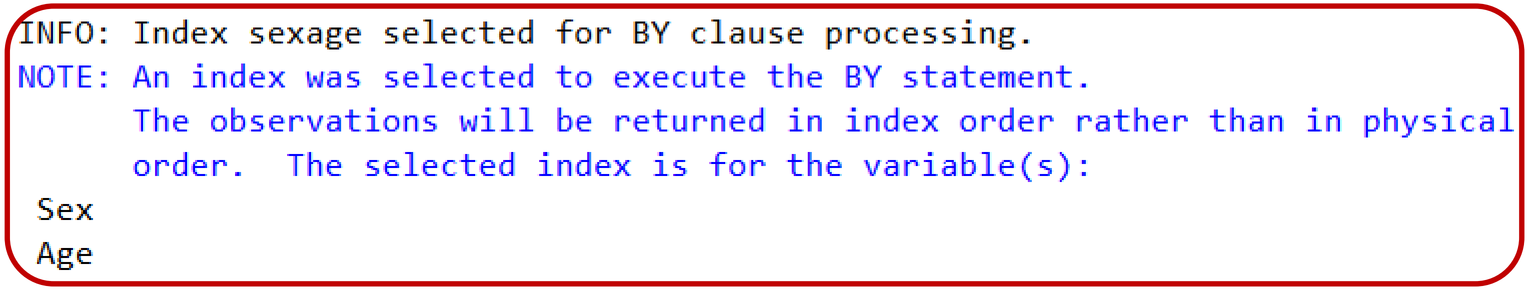
```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: Composite index sexage has been defined.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Composite index

- data can now be used with a BY statement without the need for the dataset to be sorted:

```
proc means data=class;
  by sex age;
  var height weight;
run;
```

- log confirms that the index has been used:



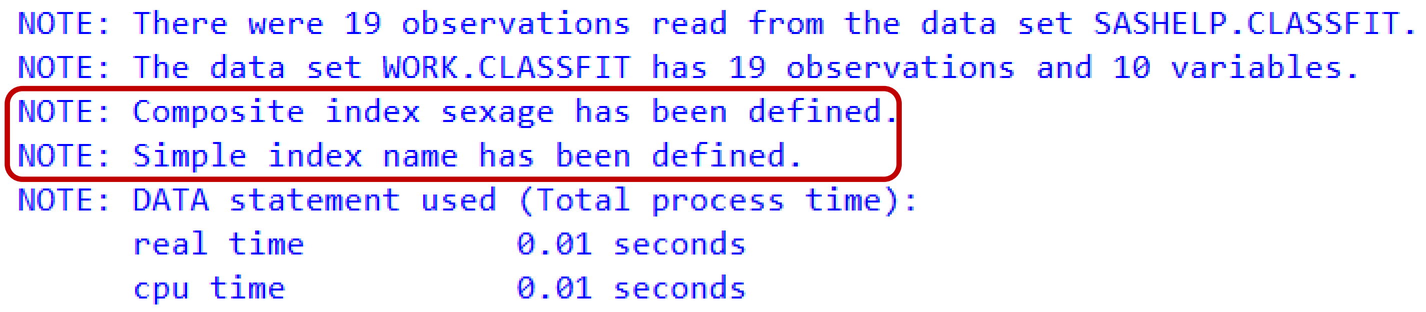
```
INFO: Index sexage selected for BY clause processing.
NOTE: An index was selected to execute the BY statement.
      The observations will be returned in index order rather than in physical
      order.  The selected index is for the variable(s):
 Sex
 Age
```

Multiple indexes

- define simple index NAME along with a composite index on variables SEX and AGE:

```
data classfit (index=(name
                      sexage=(sex age)));
  set sashelp.classfit;
run;
```

- log confirms that both indexes have been defined:



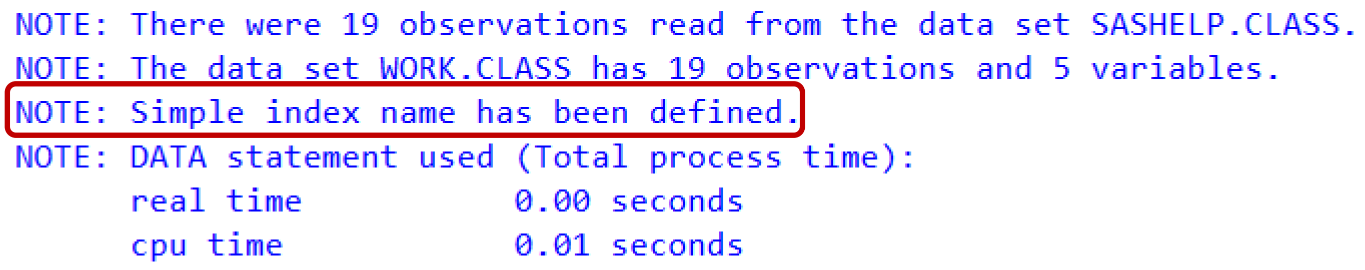
```
NOTE: There were 19 observations read from the data set SASHELP.CLASSFIT.
NOTE: The data set WORK.CLASSFIT has 19 observations and 10 variables.
NOTE: Composite index sexage has been defined.
NOTE: Simple index name has been defined.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Unique index

- a unique index can be used to ensure that the key variable(s) are unique for each row:

```
data class (index=(name/unique));
  set sashelp.class;
run;
```

- the index creation is successful, confirming that NAME is unique:



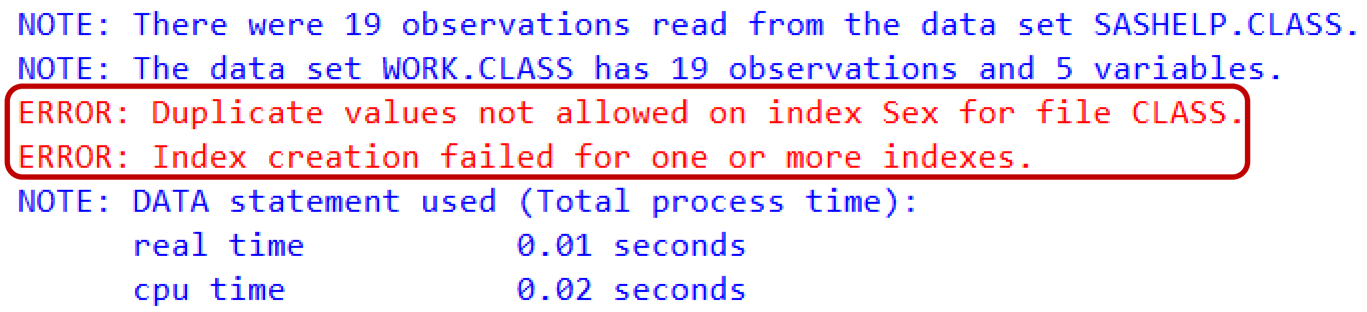
```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: Simple index name has been defined.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds
```

Unique index

- SAS will reject the index and give an error if any duplicate keys exist:

```
data class (index=(sex/unique));
  set sashelp.class;
run;
```

- log shows that the index creation has failed:

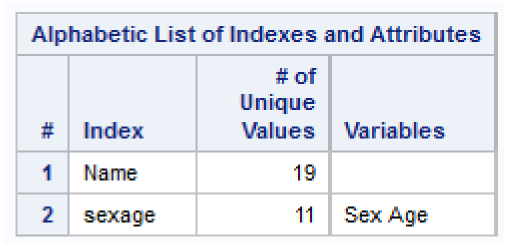


```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
ERROR: Duplicate values not allowed on index Sex for file CLASS.
ERROR: Index creation failed for one or more indexes.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.02 seconds
```

Index management

- indexes can be easily viewed using either PROC CONTENTS:

```
proc contents data=classfit;
run;
```



Alphabetic List of Indexes and Attributes

#	Index	# of Unique Values	Variables
1	Name	19	
2	sexage	11	Sex Age

Index management

- or PROC SQL:

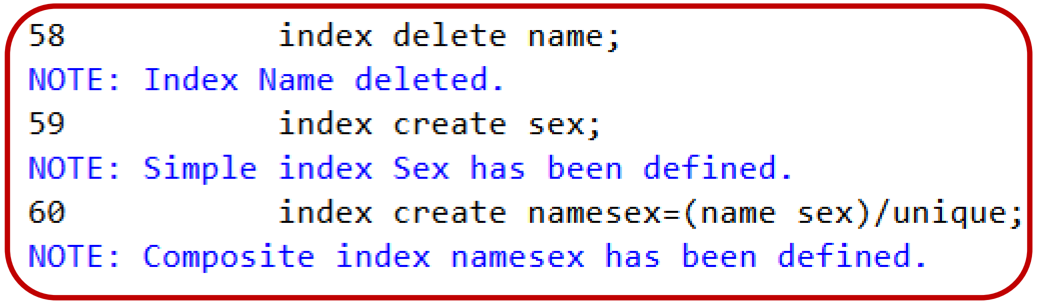
```
proc sql;
  describe table classfit;
quit;
```

```
NOTE: SQL table WORK.CLASSFIT was created like:

create table WORK.CLASSFIT( bufsize=65536 )
  (
   Name char(8),
   Sex char(1),
   Age num,
   Height num
  );
create index Name on WORK.CLASSFIT(Name);
create index sexage on WORK.CLASSFIT(Sex,Age);
```

Index management

- indexes can be easily added to (or deleted from) existing datasets using either PROC DATASETS:

```
proc datasets nolist;
  modify classfit;
  index delete name;
  index create sex;
  index create namesex=(name sex)/unique;
quit;
```

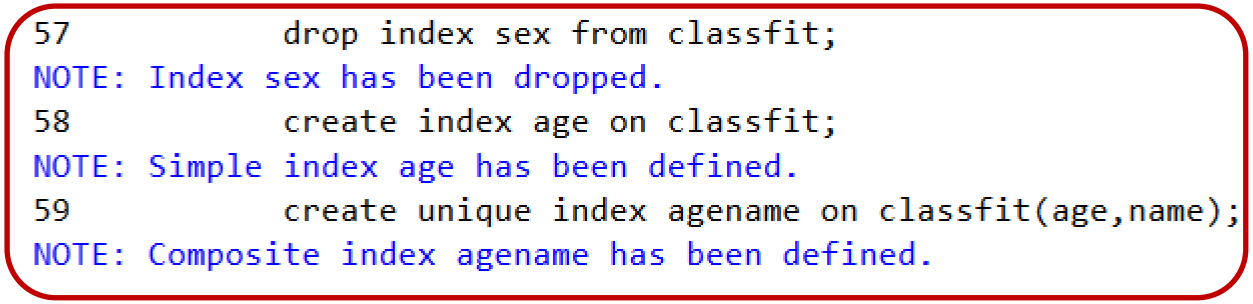


```
58           index delete name;
NOTE: Index Name deleted.
59           index create sex;
NOTE: Simple index Sex has been defined.
60           index create namesex=(name sex)/unique;
NOTE: Composite index namesex has been defined.
```

Index management

- indexes can be easily added to (or deleted from) existing datasets using either PROC DATASETS or PROC SQL:

```
proc sql;
  drop index sex from classfit;
  create index age on classfit;
  create unique index agename on classfit(age,name);
 quit;
```

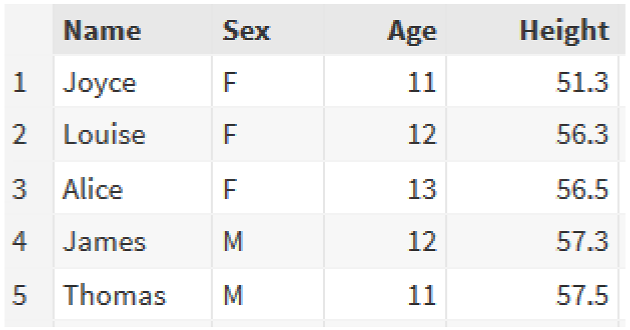


```
          drop index sex from classfit;
NOTE: Index sex has been dropped.
          create index age on classfit;
NOTE: Simple index age has been defined.
          create unique index agename on classfit(age,name);
NOTE: Composite index agename has been defined.
```

Index example

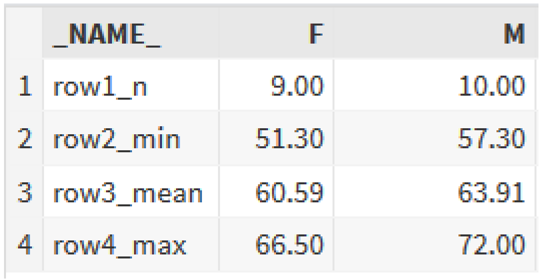
```
data class1;
  merge sashelp.class classfit;
  by name;
run;

proc sort data=class1;
  by sex;
run;

proc means data=class1 noprint;
  by sex;
  var height;
  output out=class2 (drop=_:) n=row1_n min=row2_min mean=row3_mean max=row4_max;
run;

proc transpose data=class2 out=class3;
  by sex;
  var row:;
run;

proc sort data=class3;
  by _name_;
run;

proc transpose data=class3 out=height_summary;
  by _name_;
  var col1;
  format col1 8.2;
  id sex;
run;
```

CLASSFIT



	Name	Sex	Age	Height
1	Joyce	F	11	51.3
2	Louise	F	12	56.3
3	Alice	F	13	56.5
4	James	M	12	57.3
5	Thomas	M	11	57.5

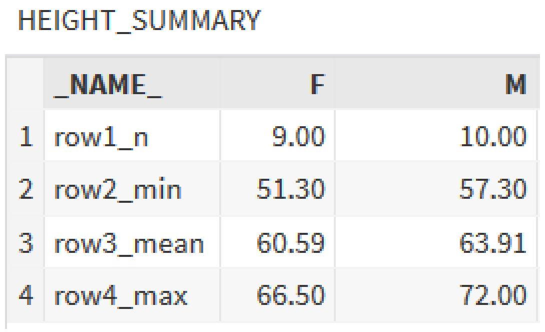
HEIGHT_SUMMARY



	NAME	F	M
1	row1_n	9.00	10.00
2	row2_min	51.30	57.30
3	row3_mean	60.59	63.91
4	row4_max	66.50	72.00

Index example

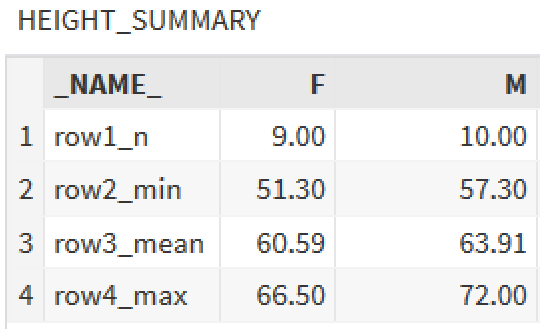
```
data class1 (index=(sex));
  merge sashelp.class classfit;
  by name;
run;

proc means data=class1 noprint;
  by sex;
  var height;
  output out=class2 (drop=_:) n=row1_n min=row2_min mean=row3_mean max=row4_max;
run;

proc transpose data=class2 out=class3;
  by sex;
  var row:;
run;

proc sort data=class3;
  by _name_;
run;

proc transpose data=class3 out=height_summary;
  by _name_;
  var col1;
  format col1 8.2;
  id sex;
run;
```


HEIGHT_SUMMARY

	NAME	F	M
1	row1_n	9.00	10.00
2	row2_min	51.30	57.30
3	row3_mean	60.59	63.91
4	row4_max	66.50	72.00

Index example

```
data class1 (index=(sex));
  merge sashelp.class classfit;
  by name;
run;

proc means data=class1 noprint;
  by sex;
  var height;
  output out=class2 (drop=_:) n=row1_n min=row2_min mean=row3_mean max=row4_max;
run;

proc transpose data=class2 out=class3 (index=(_name_));
  by sex;
  var row:;
run;

proc transpose data=class3 out=height_summary;
  by _name_;
  var col1;
  format col1 8.2;
  id sex;
run;
```



HEIGHT_SUMMARY

	NAME	F	M
1	row1_n	9.00	10.00
2	row2_min	51.30	57.30
3	row3_mean	60.59	63.91
4	row4_max	66.50	72.00

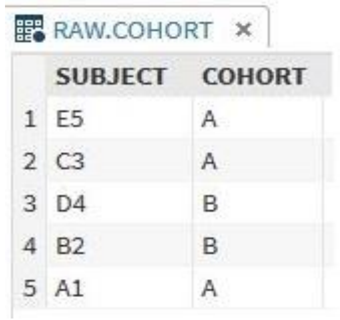
What is a Hash Object?

- hash objects are a type of data structure that allows SAS to efficiently search for data
- stored in memory and only exists during the execution of the data step
- can be used to combine two or more datasets
- no need for either dataset to be sorted or ordered and the order of the original dataset is unchanged

Combining unsorted datasets

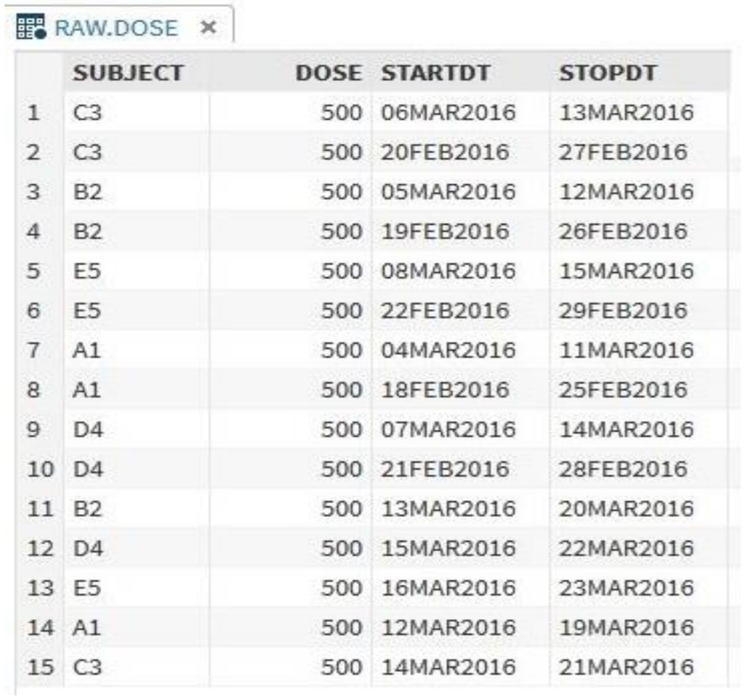
- need to combine RAW.DOSE and RAW.COHORT datasets, only keeping subjects in Cohort A

- neither dataset is sorted, and we want to retain the original order of RAW.DOSE



RAW.COHORT

	SUBJECT	COHORT
1	E5	A
2	C3	A
3	D4	B
4	B2	B
5	A1	A



RAW.DOSE

	SUBJECT	DOSE	STARTDT	STOPDT
1	C3	500	06MAR2016	13MAR2016
2	C3	500	20FEB2016	27FEB2016
3	B2	500	05MAR2016	12MAR2016
4	B2	500	19FEB2016	26FEB2016
5	E5	500	08MAR2016	15MAR2016
6	E5	500	22FEB2016	29FEB2016
7	A1	500	04MAR2016	11MAR2016
8	A1	500	18FEB2016	25FEB2016
9	D4	500	07MAR2016	14MAR2016
10	D4	500	21FEB2016	28FEB2016
11	B2	500	13MAR2016	20MAR2016
12	D4	500	15MAR2016	22MAR2016
13	E5	500	16MAR2016	23MAR2016
14	A1	500	12MAR2016	19MAR2016
15	C3	500	14MAR2016	21MAR2016

Hash Object example

```
/* set N to current order of dataset */
data dose1;
  set raw.dose;
  n=_n_;
run;

/* sort DOSE by Subject */
proc sort data=dose1;
  by subject;
run;

/* sort COHORT by subject */
proc sort data=raw.cohort out=cohort;
  by subject;
run;

/* merge DOSE with COHORT, only keep Cohort A subjects */
data dose2;
  merge dose1 (in=a) cohort (where=(cohort='A') in=b);
  by subject;
  if a & b;
run;

/* sort DOSE back to original order */
proc sort data=dose2 out=cut.dose (drop=n);
  by n;
run;
```

Hash Object example

```
/* combine DOSE with COHORT, only keep Cohort A subjects */
data cut.dose;
  length SUBJECT $2 COHORT $1;

  if _n_=1 then do;
    declare hash h(dataset:"raw.cohort(where=(cohort='A'))"); ①
    h.defineKey("subject"); ②
    h.defineData("cohort"); ③
    h.defineDone();
    call missing (subject,cohort);
  end;

  set raw.dose; ④
  rc = h.find(); ⑤
  if rc = 0 then output; ⑥
  drop rc;
run;
```

① read the RAW.COHORT dataset into the hash object

② define SUBJECT as the key variable (equivalent to the BY variable in a merge)

③ define any data item variables that are to be added to the new dataset

④ read in the RAW.DOSE dataset

⑤ h.find() is the method used to retrieve the data from the hash object

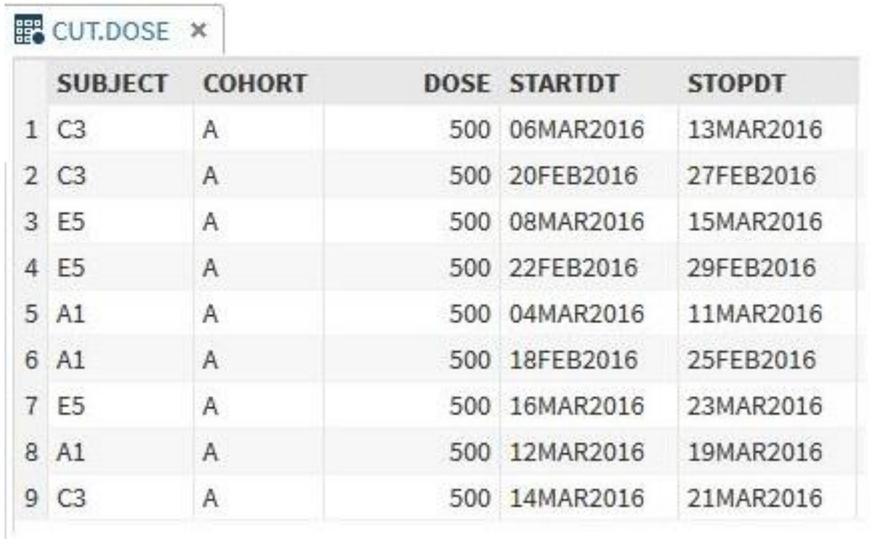
⑥ a return code of zero indicates that the find was successful

Hash Object example

- SAS log confirms that 9 observations have been output:

```
NOTE: There were 3 observations read from the data set RAW.COHORT.
      WHERE cohort='A';
NOTE: There were 15 observations read from the data set RAW.DOSE.
NOTE: The data set CUT.DOSE has 9 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.04 seconds
```

- contains only the Cohort A subjects, original order is retained

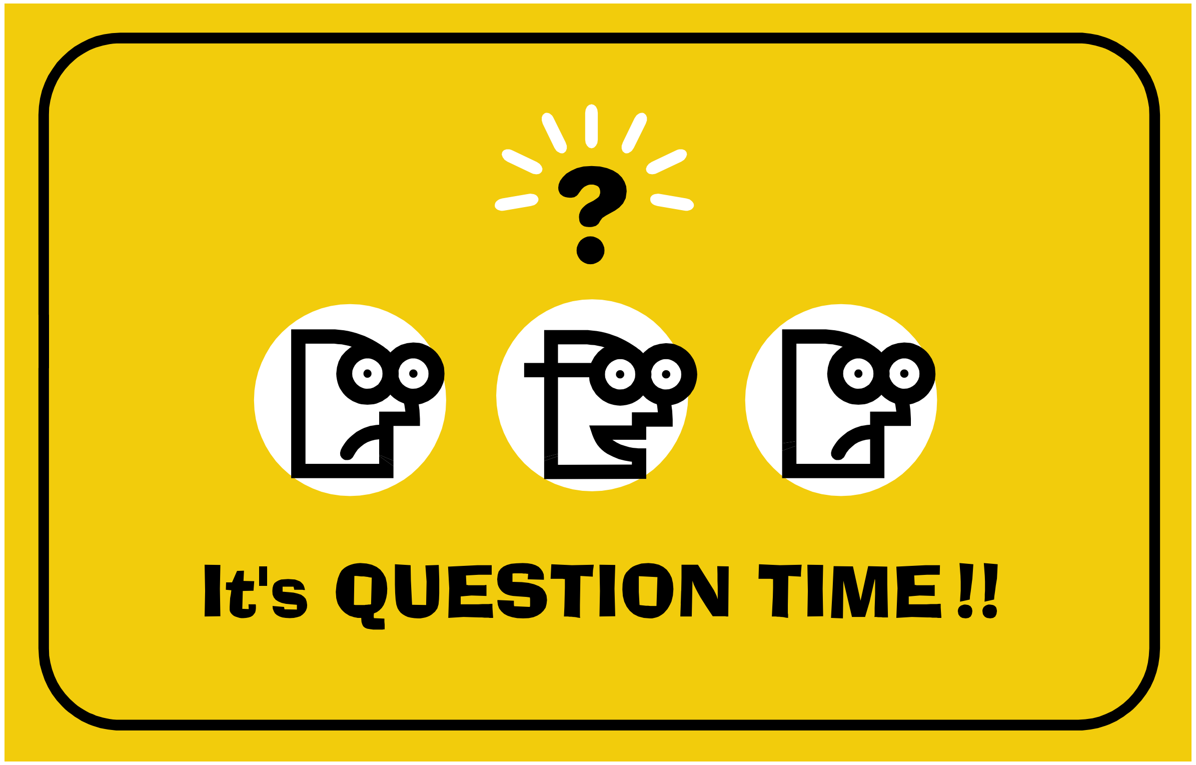

CUT.DOSE

	SUBJECT	COHORT	DOSE	STARTDT	STOPDT
1	C3	A	500	06MAR2016	13MAR2016
2	C3	A	500	20FEB2016	27FEB2016
3	E5	A	500	08MAR2016	15MAR2016
4	E5	A	500	22FEB2016	29FEB2016
5	A1	A	500	04MAR2016	11MAR2016
6	A1	A	500	18FEB2016	25FEB2016
7	E5	A	500	16MAR2016	23MAR2016
8	A1	A	500	12MAR2016	19MAR2016
9	C3	A	500	14MAR2016	21MAR2016

Conclusion

- indexes can simplify coding by replacing the sort procedure with an index option before combining or classifying data
- hash objects can be used to combine two or more unsorted datasets in a single data step
- these techniques can be used to reduce reliance on the SORT procedure resulting in shorter code, quicker and neater programming as well as improved execution time

Questions?



Contact

Jason A Smith

Argo Analytics Ltd

32 Woodlark Road, Cambridge CB3 0HS, UK

Phone: +44 7792 046599

Email: jason@argoanalytics.co.uk

LinkedIn: http://www.linkedin.com/in/jason-a-smith

Web: www.argoanalytics.co.uk