



# Tips For Dealing With Large Data Sets

---

Stephen Hanks



## What is Large?

---

- Depends on circumstances
- Batch windows
- Does it take a long time to run



## Benchmarking Vs Tips

- Benchmarking great for regular jobs
- These tips are ideas or suggestions to keep in mind for all jobs



## Use Keep and Drop

```
Editor - Untitled2 *  
Proc summary nway data=tips2.example;  
  class Day_of_week;  
  var gross_profit;  
  output out=work.summary sum=;  
run;
```

```
NOTE: There were 2884224 observations read from the data set TIPS2.EXAMPLE.  
NOTE: The data set WORK.SUMMARY has 7 observations and 4 variables.  
NOTE: PROCEDURE SUMMARY used:  
      real time      42.95 seconds  
      cpu time       5.56 seconds
```

```
Editor - Untitled2 *  
Proc summary nway data=tips2.example(keep=Day_of_week gross_profit);  
  class Day_of_week;  
  var gross_profit;  
  output out=work.summary sum=;  
run;
```

```
NOTE: There were 2884224 observations read from the data set TIPS2.EXAMPLE.  
NOTE: The data set WORK.SUMMARY has 7 observations and 4 variables.  
NOTE: PROCEDURE SUMMARY used:  
      real time      35.21 seconds  
      cpu time       5.05 seconds
```



## Macro Variables Not Functions

```
data fred;
    set cm_data.sales (where=(
        intnx('month',today(),0,'B')
        le trans_date le
        intnx('month',today(),0,'E')));
run;
```



## Macro Variables Not Functions

```
data _null_;
    start=intnx('month',today(),0,'B');
    end=intnx('month',today(),0,'E');
    call symput('start',start);
    call symput('end',end);
run;

data fred2;
    set cm_data.sales
        (where=(&start le trans_date le &end));
run;
```



## Macro Variables Not Functions

---

NOTE: The data set WORK.FRED has 325130 observations and 30 variables.

NOTE: DATA statement used:

real time	4:02.67
user cpu time	3:38.76
system cpu time	17.60 seconds

NOTE: The data set WORK.FRED2 has 325130 observations and 30 variables.

NOTE: DATA statement used:

real time	1.91 seconds
user cpu time	0.97 seconds
system cpu time	0.39 seconds



## Sub Setting if's, Start With Largest

---

```
data test1;
set cm_data.sales;
length date $15;
if year(trans_date)=year(today())-1
  then date='Last Year';
else if year(trans_date)=year(today())
  then date='This Year';
else date='Previous Years';
run;
```



## Sub Setting if's, Start With Largest

---

```
data test2;
set cm_data.sales;
length date $15;
if year(trans_date) lt year(today())-1
    then date='Previous Years';
else if year(trans_date)=year(today())-1
    then date='Last Year';
else date= 'This Year';
run;
```



## Sub Setting if's, Start With Largest

---

```
NOTE: There were 50454448 observations read from the data set
      CM_DATA.SALES.
NOTE: The data set WORK.TEST1 has 50454448 observations and 31
      variables.
NOTE: DATA statement used:
      real time          6:29.08
      user cpu time      5:18.12
      system cpu time    59.21 seconds

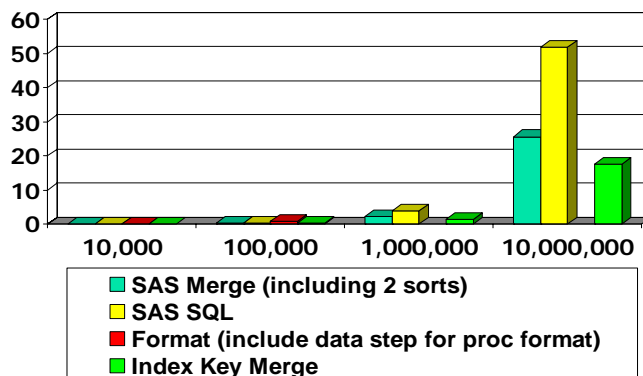
NOTE: There were 50454448 observations read from the data set
      CM_DATA.SALES.
NOTE: The data set WORK.TEST2 has 50454448 observations and 31
      variables.
NOTE: DATA statement used:
      real time          5:21.46
      user cpu time      4:04.50
      system cpu time    55.06 seconds
```

## Keep Proc and Data Sets to a Minimum

- Use keep drop and where statements in procedures.
- When amending code, if possible add changes into existing data steps.

## Joining Large Files, Forget Proc SQL

Size A = Size B





# Indexed Key Merge

	i	x
1	1	0.54028237
2	2	0.217615782
3	3	0.497206389
4	4	0.169914754
5	5	0.809005148
6	6	0.728970216
7	7	0.197403996
8	8	0.519578136
9	9	0.88329901

	i	y
1	1	0.3797289908
2	2	0.7498091505
3	3	0.2843680597
4	5	0.7574904281
5	6	0.1879373589
6	7	0.3894713402
7	8	0.812994189
8	9	0.2720566021
9	10	0.1412464139

# Indexed Key Merge



```
Editor - Untitled2 *
proc datasets lib=work;
  modify b;
  index create i/unique;
quit;

data all;
  set a;
  set b key=i/unique;
  array setmiss (*) y;
  if _iorc_ ne 0 then do;
    _error_=0;
    do i=1 to dim(setmiss);
      setmiss(i)=.;
    end;
  end;
run;
```



## Indexing

---

- Can be very useful but needs thought on when and where to index.
- Can create indexes 4 ways
  - Data step `data fred(index=a);`
  - Proc Datasets `index create a;`
  - Proc SQL `CREATE INDEX a ON fred (a);`
  - SCL Code `rc=icreate(tableid,'a',);`



## Using and Not Using Indexes

---

- Sometimes we want to force SAS to use or not use indexes
- Data Step

```
set store.touchpoint(idxwhere=yes);
```

- Proc SQL

```
proc sql;  
  create table join as  
  select a.* , b.c  
  from table1(noindex=yes) as a , table2 as b  
  where a.a = b.a ;  
quit;
```



## SPDS and SPDE

---

- What is SPDS or SPDE
- Splits data files up into many parts
- New file for each index
- Parallel processing
- No need for sorting



## Question?

---