

# SAS proc sql interface with the macro language

Elegance and power

# Proc sql can create macro variables

```
proc sql;  
  select count( distinct svctype )  
  into :n_svc  
  from mydata  
  where seg_cd in ( 'WW','CC' );  
quit;
```

# Especially useful are lists

```
proc sql;  
  select distinct sic2  
    into :sic_list separated by ' '  
  from mydata  
  where regn = 'NE'  
  order by sic2;  
quit;  
%put sic_list="&sic_list";
```

# Lists, Arrays and Vectors

- For convenience, we can think of a list as a row vector and an array as a column vector
- Simple list example:  

```
%let seglist = WW CC FF SS;
```
- Simple array example:  

```
%let seg1 = WW;  
%let seg2 = CC;  
%let seg3 = FF;  
%let seg4 = SS;
```

# Using proc sql to make an array

```
proc sql;  
  select distinct seg_cd  
  into :seg1 - :seg10  
  from mydata  
  order by seg_cd;  
quit;
```

# A recent real world project

- 50+ datasets in 5 directories
- Built a dataset listing all datasets with other meta data
- Receive requests from a web interface for one or more time series
- Must identify the dataset containing each series

- Requests contain which opco and which class variables (in a list) for each time series
- These 2 items are sufficient to uniquely identify the dataset containing the requested data

# Identifying the datasets - Proc sql

```
proc sql noprint;
  %do i=1 %to &n_ts;
    select memname, path
      into :mem&i separated by ' ',
      :path&i separated by ' '
    from meta.meta_profiles
    where opco="&&opco&i"
          and class_list = "&&clist&i";
    %put &i mem=&&mem&i path=&&path&i;
  %end;
quit;
```

- This give me lists (arrays) of dataset names and the directory paths for each time series requested
- Data can now be extracted and reported