• Positioning : EG vs. Base SAS vs. Data Integration Studio
• Compatibility matrix EG / SAS versions
• Features of Enterprise Guide (demo)
  • Splitting the code window
  • Code autocomplete / hints
  • SAS code analyser
  • SAS code versioning (Version 7.x)
  • Conditional branching in a project
  • Exporting code from a project
  • Creating a stored process
• Features of SAS Code
  • Proc GINSIDE
  • In Memory Processing
  • Scaling up and out
  • Programming shortcuts
  • Mix and match tasks and coding (demo : exploding a pie segment)
• Administration and security: demo
## Compatibility Matrix

### SAS / Enterprise Guide Versions

<table>
<thead>
<tr>
<th></th>
<th>4.1</th>
<th>4.2</th>
<th>4.3</th>
<th>5.1</th>
<th>6.1</th>
<th>7.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>9.2</td>
<td>L</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y*</td>
<td>Y</td>
</tr>
<tr>
<td>9.3</td>
<td>N</td>
<td>N</td>
<td>Y*</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>9.4</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y*</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

L: Limited support (Enterprise Guide and SAS on same host, no metadata)

* With appropriate hotfixes applied

Note SAS 9.4 is not supported on Windows XP

The "Split" gadget allows a program window to show two sections in the same code.
Code help is controlled in Programs > Editor Options > Autocomplete
CODE ANALYSER TURNS THIS...
CODE ANALYSER INTO THIS...
CODE VERSIONING
DEMONSTRATION
CODE VERSIONING
- DEMO

NEW IN SAS ENTERPRISE GUIDE 7
CODE VERSIONING

```
proc print data=sashelp.class;
  where sex="M";
run;
```
CODE VERSIONING

History enables comparison of any versions.

Revert enables switching to a previous version of a program.
The blame tab shows who created each line of code and in which version.
CONDITIONAL BRANCHING

THE COMPLETE PROJECT

CUSTOMER_DIM -> Query Builder -> WORK.QUERY_FOR_CUSTOMER_DIM

List Data

SAS Report - List Data

Summary Tables

SAS Report - Summary Tables
Prompts enable a user to supply values for parameters in a task or stored process. Prompts are implemented as macro variables. Selecting "Use prompt throughout project" makes the macro variable global (within a project). Prompts can have:

- Dependencies
- Default values
- Single or Multiple selection
- Mandatory non-blank values
- Static or dynamic pick lists and more
In the query builder, select the columns to include in the result and define filters which refer to your parameter(s)

This query task asks for a value for the prompt &lastname, which has been previously defined using the Enterprise Guide prompt manager.
Right click a branch and select condition > add
Define the condition for this branch to execute

This condition checks the automatic variable &sqlobs, which contains the number of rows created in the query task.
Optionally add an else…action to execute another branch

Alternatively define conditions in each branch
Run the project

Supply values for prompts as needed.
A tick indicates the conditions for a branch were met on this run and the Summary tables branch executed.
When result rows <=50 a listing is produced

<table>
<thead>
<tr>
<th>Row number</th>
<th>Customer First Name</th>
<th>Customer Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Robin</td>
<td>Abbott</td>
</tr>
</tbody>
</table>
When result rows >50 a summary is produced

<table>
<thead>
<tr>
<th>Customer Gender</th>
<th>15-30 years</th>
<th>31-45 years</th>
<th>46-60 years</th>
<th>61-75 years</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>17</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>56</td>
</tr>
<tr>
<td>M</td>
<td>22</td>
<td>19</td>
<td>19</td>
<td>23</td>
<td>83</td>
</tr>
<tr>
<td>Total</td>
<td>39</td>
<td>32</td>
<td>32</td>
<td>36</td>
<td>139</td>
</tr>
</tbody>
</table>
Optionally, add notes to the project and rename nodes to make it easier to follow the logic.
Either export the selected process flow, or the whole project.
Options may alter how much code is displayed in the log, but all code is exported.
STORED PROCESSES CAN BE CREATED FROM CODE OR PROJECTS
FEATURES OF SAS CODE
• SAS/Graph can display graphs for anything where X-Y coordinates are available
• How do you know if a location lies within a geographic boundary?
• SAS has a the GINSIDE procedure to answer this question
• Here's a step-by-step example of usage
PROC GINSIDEB  FIRST CONSIDER THE DATA

• This example uses SAS/Graph data set mapsas.counties
• Shipped with SAS, also freely downloadable from the SAS/GRAPH product documentation page on support.sas.com
• Here's the first few observations

<table>
<thead>
<tr>
<th>Obs</th>
<th>COUNTY</th>
<th>STATE</th>
<th>SEGMENT</th>
<th>DENSITY</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.51449</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.51343</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.51344</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.51239</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.51191</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.50819</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.50818</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.50818</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.50816</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1.50846</td>
</tr>
</tbody>
</table>
PROC GINSIDE NOW CREATE THE CUSTOMER LOCATIONS DATA SET

data customers;
  length city $20;
  input lastname$ zip x y city $;
  datalines;
  Smith    27611 1.374164 0.623436 Raleigh
  Jones    27560 1.375948 0.625278 Morrisville
  Doe      27513 1.375279 0.624922 Cary
  Patel    27520 1.369120 0.621970 Clayton
  White    27705 1.377910 0.628629 Durham
  Short    27587 1.370373 0.627680 WakeForest
  Phillips 27591 1.368124 0.624705 Wendell
  Jackson  27597 1.367264 0.625629 Zebulon
  ;
  run;

### PROC GINSIDE  CUSTOMERS CURRENTLY LOOKS LIKE THIS…

<table>
<thead>
<tr>
<th>Obs</th>
<th>city</th>
<th>lastname</th>
<th>zip</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
<td>1.37416</td>
<td>0.62344</td>
</tr>
<tr>
<td>2</td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
<td>1.37595</td>
<td>0.62528</td>
</tr>
<tr>
<td>3</td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
<td>1.37528</td>
<td>0.62492</td>
</tr>
<tr>
<td>4</td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
<td>1.36912</td>
<td>0.62197</td>
</tr>
<tr>
<td>5</td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
<td>1.37791</td>
<td>0.62863</td>
</tr>
<tr>
<td>6</td>
<td>WakeForest</td>
<td>Short</td>
<td>27587</td>
<td>1.37037</td>
<td>0.62768</td>
</tr>
<tr>
<td>7</td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
<td>1.36812</td>
<td>0.62471</td>
</tr>
<tr>
<td>8</td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
<td>1.36726</td>
<td>0.62563</td>
</tr>
</tbody>
</table>
PROC GINSIDE SUBSET MAPSSAS.COUNTIES TO JUST WAKE COUNTY NORTH CAROLINA

• data mymap;
  • set mapssas.counties(where=(fipstate(state)='NC' and county=183));
• run;
PROC GINSIDE CONCATENATE THE TWO DATASETS, PROJECT THE COORDINATES AS A MAP THEN SPLIT THEM AGAIN

- data combined;
- set customers mymap;
- run;
- proc gproject =combined out=combined dupok;
- id state county;
- run;
- data mymap customers;
- set combined;
- if missing(zip) = 1 then output mymap;
- else output customers;
- run;
CUSTOMERS NOW LOOKS LIKE THIS, AS THE COMBINED DATASET HAD ADDITIONAL COLUMNS

<table>
<thead>
<tr>
<th>Obs</th>
<th>x</th>
<th>y</th>
<th>DENSITY</th>
<th>city</th>
<th>lastname</th>
<th>zip</th>
<th>COUNTY</th>
<th>STATE</th>
<th>SEGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.00154</td>
<td>-0.00134</td>
<td>.</td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>-0.00299</td>
<td>0.000507</td>
<td>.</td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>-0.00244</td>
<td>0.000149</td>
<td>.</td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>0.002556</td>
<td>-0.0028</td>
<td>.</td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>-0.00457</td>
<td>0.003862</td>
<td>.</td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>0.001531</td>
<td>0.002906</td>
<td>.</td>
<td>WakeForest</td>
<td>Short</td>
<td>27587</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>0.003359</td>
<td>-6.6E-05</td>
<td>.</td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>0.004054</td>
<td>0.00086</td>
<td>.</td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
**PROC GINSIDE**

RUN PROC GINSIDE TO SEE IF CUSTOMERS ARE INSIDE THE SELECTED AREA

```sas
proc ginside map=mymap data=customers out=customers;
   id state county;
run;
```

Proc GINSIDE adds the ID variables for the coordinates which are inside the boundary,
### PROC GINSIDE

Customers now looks like this

<table>
<thead>
<tr>
<th>Obs</th>
<th>x</th>
<th>y</th>
<th>SEGMENT</th>
<th>STATE</th>
<th>COUNTY</th>
<th>DENSITY</th>
<th>city</th>
<th>lastname</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.00154</td>
<td>-0.00134</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
</tr>
<tr>
<td>2</td>
<td>-0.00299</td>
<td>0.000507</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
</tr>
<tr>
<td>3</td>
<td>-0.00244</td>
<td>0.000149</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
</tr>
<tr>
<td>4</td>
<td>0.001531</td>
<td>0.002906</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>WakeForest</td>
<td>Short</td>
<td>27587</td>
</tr>
<tr>
<td>5</td>
<td>0.003359</td>
<td>-6.6E-05</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
</tr>
<tr>
<td>6</td>
<td>0.004054</td>
<td>0.00086</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
</tr>
<tr>
<td>7</td>
<td>0.002556</td>
<td>-0.0028</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
</tr>
<tr>
<td>8</td>
<td>-0.00457</td>
<td>0.003862</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
</tr>
</tbody>
</table>
PROC GINSIDE

NOW ADD SOME ADDITIONAL COLUMNS TO ANNOTATE THE GRAPH..

data customers;
  set customers;
  length color style $8;
  retain xsys ysys '2' hsys '3' when 'a' position '5' size 5;
  style='marker';
  if missing(county) then do;
    color='red';
    text='X';
  end;
  else do;
    color='green';
    text='U'; /* Marker font for a square */
  end;
  output;
  style='markere'; /* outline the symbols */
  color='gray33';
  output;
run;
### PROC GINSIDE

**CUSTOMERS NOW LOOKS LIKE THIS...**

<table>
<thead>
<tr>
<th>Obs</th>
<th>x</th>
<th>y</th>
<th>SEGMENT</th>
<th>STATE</th>
<th>COUNT</th>
<th>DENSITY</th>
<th>city</th>
<th>lastname</th>
<th>zip</th>
<th>color</th>
<th>style</th>
<th>xsys</th>
<th>ysys</th>
<th>hsys</th>
<th>when</th>
<th>position</th>
<th>size</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.00154</td>
<td>-0.00134</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-0.00154</td>
<td>-0.00134</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.00299</td>
<td>0.000507</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.00299</td>
<td>0.000507</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.00244</td>
<td>0.000149</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.00244</td>
<td>0.000149</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.001531</td>
<td>0.002906</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>WakeForest</td>
<td>Shortest</td>
<td>27587</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.001531</td>
<td>0.002906</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>WakeForest</td>
<td>Shortest</td>
<td>27587</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.003359</td>
<td>-6.6E-05</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.003359</td>
<td>-6.6E-05</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.004054</td>
<td>0.00086</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
<td>green</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.004054</td>
<td>0.00086</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5U</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.002556</td>
<td>-0.0028</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
<td>red</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5X</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.002556</td>
<td>-0.0028</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5X</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-0.00457</td>
<td>0.003862</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
<td>red</td>
<td>marker</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5X</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>-0.00457</td>
<td>0.003862</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td>.</td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
<td>gray33</td>
<td>markere</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>a</td>
<td>5</td>
<td>5X</td>
<td></td>
</tr>
</tbody>
</table>
PROC GMAP DATA=mymap MAP=mymap ANNO=customers;
  ID state county;
  CHORO county / STAT=SUM COUTLINE=BLACK NOLEGEND;
RUN;
QUIT;
PROC GINSIDE  THE FINAL RESULT..
If the map wasn't subset to just Wake County, proc GINSIDE would add the ID variables for all points which fall inside any boundary:

<table>
<thead>
<tr>
<th>Obs</th>
<th>x</th>
<th>y</th>
<th>SEGMENT</th>
<th>STATE</th>
<th>COUNTY</th>
<th>DENSITY</th>
<th>city</th>
<th>lastname</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.016334</td>
<td>0.008947</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>Raleigh</td>
<td>Smith</td>
<td>27611</td>
</tr>
<tr>
<td>2</td>
<td>0.014867</td>
<td>0.010772</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>Morrisville</td>
<td>Jones</td>
<td>27560</td>
</tr>
<tr>
<td>3</td>
<td>0.015413</td>
<td>0.010422</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>Cary</td>
<td>Doe</td>
<td>27513</td>
</tr>
<tr>
<td>4</td>
<td>0.019353</td>
<td>0.013229</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>WakeForest</td>
<td>Short</td>
<td>27587</td>
</tr>
<tr>
<td>5</td>
<td>0.021218</td>
<td>0.010281</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>Wendell</td>
<td>Phillips</td>
<td>27591</td>
</tr>
<tr>
<td>6</td>
<td>0.021901</td>
<td>0.011215</td>
<td>.</td>
<td>37</td>
<td>183</td>
<td></td>
<td>Zebulon</td>
<td>Jackson</td>
<td>27597</td>
</tr>
<tr>
<td>7</td>
<td>0.02045</td>
<td>0.007534</td>
<td>.</td>
<td>37</td>
<td>101</td>
<td></td>
<td>Clayton</td>
<td>Patel</td>
<td>27520</td>
</tr>
<tr>
<td>8</td>
<td>0.013244</td>
<td>0.014107</td>
<td>.</td>
<td>37</td>
<td>63</td>
<td></td>
<td>Durham</td>
<td>White</td>
<td>27705</td>
</tr>
</tbody>
</table>
IN - MEMORY PROCESSING
IN MEMORY PROCESSING

• I/O (disk) is often the biggest constraint on SAS systems.
• Memory is often abundant and cheap.

In this code `orion.customerdim` is read from disk four times

```sas
proc freq data=orion.customerdim;
  tables CustomerCountry CustomerType;
run;
proc print data=orion.customerdim noobs;
  where CustomerType='Orion Club Gold members high activity';
  var CustomerID CustomerName CustomerAgeGroup;
run;
proc means data=orion.customerdim mean median max min;
  var CustomerAge;
  class CustomerGroup;
run;
proc tabulate data=orion.customerdim format=8.;
  class CustomerAgeGroup CustomerType;
  table CustomerType All=Total,
       CustomerAgeGroup*n=' ' All=Total*n=' '/rts=45;
run;
```
The SASFILE global statement loads an entire SAS dataset into memory for subsequent DATA and PROC steps. Loading only performs one I/O action.

**OPEN**
Opens the file and allocates the buffers, but defers reading the data into memory until a procedure or a statement that references the file is executed.

**LOAD**
Opens the file, allocates the buffers, and reads the data into memory.

**CLOSE**
Releases the buffers and closes the file.
Before using the dataset use the LOAD (or OPEN) option. Ensure you use the CLOSE option once you have finished with the table to free up system resource.

```sas
sasfile orion.customerdim load;
proc freq data=orion.customerdim tables CustomerType; run;
proc print data=orion.customerdim noobs;
   where CustomerType='Orion Club Gold members high active';
   var CustomerID CustomerName CustomerAgeGroup;
run;
proc means data=orion.customerdim mean median max min;
   var CustomerAge;
   class CustomerGroup;
run;
proc tabulate data=orion.customerdim format=8.;
   class CustomerAgeGroup CustomerType;
   table CustomerType All=Total
      CustomerID= ' /rts=45;
run;
sasfile orion.customerdim close;
```
CAUTION:

• If your dataset is larger than the amount of memory available this can degrade performance.

• The operating system may already cache file reads

RECOMMENDATIONS:

• Run your code with and without the SASFILE option to compare the results.

• SASFILE will be a greater benefit during peak loads on your system.
1. Terminology
2. Performance enhancements in Base SAS9
3. SAS/Connect : Scaling up and out
4. Storage and Scalability
TERMINOLOGY

- Scaling UP
  - Coping with bigger tables
- Scaling OUT
  - Coping with more users
- Threading
  - Splitting a task into multiple smaller work units, each of which can be run on a different processor
TERMINOLOGY

• Striping
  • Splitting a table into many smaller chunks, and distributing them across multiple disks

• Pipelining
  • Handing partial results from one task to another, enabling a second task to begin before the first one has completed.
PERFORMANCE ENHANCEMENTS IN BASE SAS

- Parallel Processing means processing is handled by multiple CPUs simultaneously.
- SAS performance is improved through
  - Threaded I/O
  - Threaded application processing
• Processes suitable for threading include:
  • Sorting
  • Grouping
  • Summarising

• Not every procedure is a candidate for threading e.g. Proc Print
THREADED PROCEDURES

- PROC REG
- PROC GLM
- PROC LOESS
- PROC DMREG
- PROC DMINE
- PROC SORT
- PROC SUMMARY / MEANS
- PROC SQL
- PROC REPORT
- PROC TABULATE
ENABLING THREADING

• Thread enabled procedures use threading by default.
• Threading can be enabled or disabled throughout the SAS System using the THREADS system option
• General form of the THREADS system option

OPTIONS THREADS | NOTHREADS
• Threading can be set at step level

**PROC MEANS** DATA=\textit{SAS-data-set} THREADS | NOTHREADS;

**PROC SORT** DATA=\textit{SAS-data-set} THREADS | NOTHREADS;

**PROC SQL** THREADS | NOTHREADS;
ENABLING THREADING

• The number of CPUs to use for processing is controlled with the CPUCOUNT system option
• General form of the CPUCOUNT system option:

```
OPTION CPUCOUNT=1-1024 | ACTUAL;
```

• 1-1024 is the number of CPUs that SAS will assume are available for use by thread-enabled applications
• ACTUAL is the number of CPUs that SAS detects are available for a specific session
• Basic requirements
• Connect in SAS 6
• New features in SAS8
• New features in SAS 9
Network connection
e.g. TCP/IP
SAS/CONNECT REQUIRED SETTINGS

• How the SAS sessions will talk
• What machine to talk to
• Optionally, a script file can be used to automate all or part of the signon
• The instruction to connect
%let server3=127.0.0.1;
options comamid=tcp remote=server3;
filename rlink "C:sas\connect\scripts\tcpunix.scr";
signon;
rsSubmit;
  proc print data=sashelp.class;
  run;
endrsSubmit;
options comamid=tcp remote=server3 autosignon;

/*submit several jobs simultaneously*/
rssubmit server1 wait=no macvar=SVR1_status;
   proc print data=sashelp.class;
        run;
   endrssubmit;

rssubmit server2 wait=no macvar=SVR2_status;
   proc print data=sashelp.class;
        run;
   endrssubmit;

/*set a synchronisation point*/
waitfor _all_ server1 server2;
SAS/CONNECT IN SAS 8
JOBS WITH NO DEPENDENCIES
## SAS/CONNECT IN SAS 8
### JOBS WITH NO DEPENDENCIES

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Job 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server 2</td>
<td>Job 2</td>
</tr>
<tr>
<td>Server 3</td>
<td>Job 3</td>
</tr>
<tr>
<td>Client</td>
<td>Job 4</td>
</tr>
</tbody>
</table>

**Time**
SAS/CONNECT IN SAS 8
JOBS WITH DEPENDENCIES

Server 1: Job1
Server 2: Job2
Server 3: Job3
Client: Job4

Time
rsubmit server1 wait=no;
libname mypipe1 sasesock "pipe1" timeout=20;
data mypipe1.demodataset;
  do i = 1 to 10000;
    y=ceil(ranuni(0)*1000); output;
  end;
run;
endrsubmit;

rsubmit server2 wait=no;
libname mypipe1 sasesock "pipe1" timeout=20;
libname mypipe2 sasesock "pipe2" timeout=20;
proc sort data=mypipe1.demodataset
  out=mypipe2.demosorted;
  by y;
run;
endrsubmit;

/*more code...*/
SAS/CONNECT IN SAS 8
JOBS WITH DEPENDENCIES

DATA step

SORT step

SUMMARY step

Disk
Pipeline parallelism ("pipelining")
SAS/CONNECT IN SAS 9
DEPENDENT JOBS

Client
Server 1
Server 2
Server 3

Job1
Job2
Job3
Job4

Time
• Part of Base SAS in Version 9
• Shares the striping and automatic parallel sort features of the SAS Scalable Performance Data Server®
• SPD Server adds client-server operation, advanced security, automatic query optimisation, dynamic clusters and row-level locking.
• Base SAS
LIBNAME demo1 "C:\folder1";

• SPD Engine (basic form)
LIBNAME demo1 SPDE "C:\folder1";

• SPD Engine (more complete form)
LIBNAME demo1 SPDE "C:\folder1" /* primary path */
   metapath="D:\disk01" /* metadata path */
   datapath="E:\disk02" /* 1st data path */
      "F:\disk03" /* 2nd data path */
      "G:\disk04" /* 3rd data path */
      "H:\disk05") /* 4th data path */
   indexpath="I:\disk06"; /* index path */
**STORAGE AND SCALABILITY**

**BASE Library**
- 20 MB/second

**SPDE Library**
- > 60 MB/second

- Disk 1: Metadata
- Disk 2: Data, Data
- Disk 3: Data, Data
- Disk 4: Data, Data
- Disk 5: Data
- Disk 6: Index, Index

- Descriptor
- Data
- Index
## TOP 10 KEYBOARD SHORTCUTS

<table>
<thead>
<tr>
<th>Key combination</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift + Ctrl+U</td>
<td>Upper case selected text</td>
</tr>
<tr>
<td>Shift + Ctrl+L</td>
<td>Lower case selected text</td>
</tr>
<tr>
<td>Ctrl + Tab</td>
<td>Indent selected text 1 tab space</td>
</tr>
<tr>
<td>Shift + Ctrl + Tab</td>
<td>Unindent selected text 1 tab space</td>
</tr>
<tr>
<td>Ctrl + i</td>
<td>Auto - layout of code</td>
</tr>
<tr>
<td>Ctrl + /</td>
<td>Comment out selected text</td>
</tr>
<tr>
<td>Shift + Ctrl + /</td>
<td>Uncomment selected text</td>
</tr>
<tr>
<td>Alt + Click/drag</td>
<td>Select a rectangular block of text (also works in MS Word)</td>
</tr>
<tr>
<td>Shift + Ctrl + A</td>
<td>Add a new shortcut</td>
</tr>
<tr>
<td>Click &amp; Drag over an</td>
<td>Shows matching bracket / matching text</td>
</tr>
</tbody>
</table>
SHORTCUTS FOR SELECTING VARS IN CODE
proc sql noexec feedback;
select *
from sashelp.class;
quit;
OTHER KEYBOARD SHORTCUTS
DEMONSTRATION
For a data set with this structure left to right:

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>AGE</th>
<th>HEIGHT</th>
<th>VAR1</th>
<th>VAR2</th>
<th>VAR3</th>
<th>VAR4</th>
<th>SEX</th>
<th>VAR5</th>
</tr>
</thead>
<tbody>
<tr>
<td>($)</td>
<td>($)</td>
<td>(N)</td>
<td>(N)</td>
<td>(N)</td>
<td>(N)</td>
<td>(N)</td>
<td>(N)</td>
<td>($)</td>
<td>(N)</td>
</tr>
</tbody>
</table>

**DATA STEP COLUMN SELECTION**

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Selects</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>character</em></td>
<td>All character variables</td>
</tr>
<tr>
<td><em>numeric</em></td>
<td>All numeric variables</td>
</tr>
<tr>
<td><em>all</em></td>
<td>All variables</td>
</tr>
<tr>
<td>var1-var5</td>
<td>var1, var2, var3, var4, var5</td>
</tr>
<tr>
<td>var1--var5</td>
<td>var1, var2, var3, var4, sex, var5</td>
</tr>
<tr>
<td>Sum (Age, var1-var5);</td>
<td>Age, plus the result of subtracting var5 from var1</td>
</tr>
<tr>
<td>Sum (Age, of var1-var5);</td>
<td>Age, var1, var2, var3, var4, var5</td>
</tr>
<tr>
<td>Sum (Age, of var1--var5)</td>
<td>Age, var1, var2, var3, var4, sex, var5</td>
</tr>
</tbody>
</table>

But generates NOTE: Invalid numeric data, SEX='F' and reads SEX as a missing value, as it is a character variable)

Keep v: ;

All variables beginning with the letter 'v' var1, var2, var3, var4, var5

Note: this wildcard syntax doesn't work with functions in older versions of SAS.

sum(of var:)

all the variables starting with V, i.e. var1, var2, var3, var4, var5

id-character-sex

All the character variables between ID and SEX (inclusive) id name sex

height-numeric-var5

All the numeric variables between AGE and VAR5 (inclusive) Height VAR1 VAR2 VAR3 VAR4 VAR5
MIX AND MATCH: EXPLODING A PIE SEGMENT

DEMONSTRATION
**MIX AND MATCH**

- Use SAS Enterprise Guide as an aid in rapid application development
- For any task that is unfamiliar / used infrequently
  - Use the tasks and wizards to quickly get close to the desired result
  - Explore options without having to look up syntax
  - Then modify / extend the generated code as needed
PIE CHART TASK

MANY OPTIONS - BUT NOT ONE TO EXPLODE A SEGMENT
INITIAL RESULT

The pie chart shows the distribution of data with the following divisions:

- Red: 5
- Blue: 2
- Green: 1
- Yellow: 1
- Brown: 4
- Light purple: 15
- Purple: 4

The chart illustrates the relative proportions of each category.
The explode= option specifies values we want to pull out from the pie.
THE FINAL RESULT
TWO WAYS TO SCHEDULE FROM ENTERPRISE GUIDE

- Schedule within Enterprise Guide
  - Limited capabilities
  - No central administration
- Export project code and schedule with LSF Scheduler
  - Fully functional, supports complex flows and dependencies
  - Centralised administration
SCHEDULING

USING ENTERPRISE GUIDE SCHEDULING

Enterprise Guide

File > Schedule project: Complete answers in wizard

VB script is created

SAS Management Console
**SCHEDULING USING SAS MANAGEMENT CONSOLE SCHEDULING**

Enterprise Guide

1. Export all code in project or all code in flow
2. .sas file is created

SAS Management Console

1. Choose Deploy data Step Program
2. Choose 'New Flow'
3. Add .sas files
4. Edit Flow
5. Schedule Flow
SCHEDULE WITHIN ENTERPRISE GUIDE
• Set the scheduling options for when this project will run
• The client PC must be connected to the network to initiate the project
The project is scheduled as a VB script
1. Create scripts (code) and associated metadata
2. Create flows
3. Schedule Flows
SCHEDULING IN SAS  SCHEDULERS

- Listen for triggers
  - Time events e.g. "At 22:30 on Thursday"
  - File events e.g. "When a file called newdata.xlsx is created"
  - Other events e.g. "When a job completes successfully / when a job fails"
- Issue commands to launch jobs
  - e.g. `sas -config sasv9.cfg -sysin myprogram.sas`
- Monitor and record status of launched jobs
  - e.g. run time, exit codes for success / failure
SCHEDULING IN SAS  SAS SUPPORTED SCHEDULERS

- LSF from platform computing, now part of IBM
  - Commercial scheduler
- Operating system schedulers e.g. AT (Windows), CRON (Unix)
  - Less functionality than LSF (e.g. no complex dependencies)
SCHEDULING IN SAS REQUIREMENTS

- A file containing the code to be run (normally with a *.sas extension)
- Metadata describing the file
## Scheduling in SAS: Creating the Items

<table>
<thead>
<tr>
<th>SAS Client</th>
<th>*.sas file</th>
<th>Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS Data Integration Studio</td>
<td>&quot;Deploy job for scheduling…” wizard creates both items simultaneously</td>
<td></td>
</tr>
<tr>
<td>SAS Enterprise Guide</td>
<td>Save program file or Export project as code</td>
<td>Created in SAS Management Console scheduling plugin using &quot;Deploy Data Step Program&quot; option</td>
</tr>
<tr>
<td>SAS Display Manager</td>
<td>Save program file</td>
<td></td>
</tr>
<tr>
<td>Other text editor</td>
<td>Save script as a *.sas file</td>
<td></td>
</tr>
</tbody>
</table>
SCHEDULING IN SAS  CREATING FLOWS

- Flows are a collection of
  - Jobs (*.sas files with associated metadata)
  - Triggers
  - AND / OR gates
  - Dependencies
- Built using SAS Management Console Scheduling plugin
  - New > Flow
  - Edit > Flow
Once the flow is defined, it must be handed on to the scheduler
In SAS Management Console scheduling plugin
  1. Select the flow
  2. Right click, choose Schedule Flow....
• SAS ships LSF tools to view and monitor deployed / running flows
• The SAS licensed versions of the LSF tools have limited functionality
  • No ability to create / edit flows (use SAS Management Console for this)
  • Any non-SAS jobs will be terminated after 30 seconds
• Option of buying a full licence from IBM
THANKYOU - QUESTIONS?