Chapter 4: Combining Tables Vertically using PROC SQL
Outline

- Set Operations
- Combining Columns
- Keywords (CORR and ALL)
- Except
- Intersect
- Union
- Outer Union
### Sample Tables

#### Google Grego

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Columbia, SC</td>
</tr>
<tr>
<td>Laura</td>
<td>Cambridge, MA</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
</tr>
<tr>
<td>Joseph</td>
<td>London, England</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>John</td>
<td>Napierville, IL</td>
</tr>
</tbody>
</table>

#### Bing Grego

<table>
<thead>
<tr>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>Joseph</td>
<td>London, England</td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>VG</td>
<td>Yerevan, Armenia</td>
</tr>
</tbody>
</table>
Combining Tables Vertically

- Standard syntax:
  proc sql; select *
  from google
  (except/intersect/union/outer union)
  (corr/all)
  select * from bing;
Combining TablesVertically

- Multiple SELECT queries processed separately, then combined using set operator
- Set operators select unique rows by default
- Set operators overlay columns by default
Combining Tables Vertically

- Sequential columns in each query should be the same type
- For three or more queries, set operators are evaluated sequentially
- Keywords ALL and CORR modify the set operators
EXCEPT

- Selects unique rows from first table that do not occur in the second table
- Columns are simply overlaid (even if names are different)
- Columns inherit names from the first table
proc sql;
select * from google
except
select * from bing;
quit;

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Columbia, SC</td>
</tr>
<tr>
<td>John</td>
<td>Washington, DC</td>
</tr>
<tr>
<td>Laura</td>
<td>Cambridge, MA</td>
</tr>
</tbody>
</table>
EXCEPT

- In first sweep, the second occurrence of Melissa in google would be removed
- In second sweep, Mark, Joseph, Michael, and the first occurrence of Melissa would be removed
- The second column is called Residence rather than City
- Data is sorted
EXCEPT
EXCEPT ALL

- Use the keyword ALL after EXCEPT so that duplicate rows in the first table that do not occur in the second table are not eliminated.
- The second Melissa does not have a match.
- Data is sorted

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Columbia, SC</td>
</tr>
<tr>
<td>John</td>
<td>Washington, DC</td>
</tr>
<tr>
<td>Laura</td>
<td>Cambridge, MA</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
</tbody>
</table>
EXCEPT ALL

- We would have different results if either John from SC, Laura from MA or John from DC were listed more than once.
- Or….if Melissa from CA was not listed in the second table
EXCEPT CORR

- The keyword CORR displays only columns with the same name in both tables
- Columns are selected first
- Unique rows are then extracted from the first table that do not appear in the second table
- This ordering may generate sparser output than expected
proc sql;
select * from google
except corr
select * from bing;
quit;
EXCEPT CORR and ALL

- If both keywords ALL and CORR are used with EXCEPT
  - Unique and duplicate rows from the first table will be saved, unless they have matches in the second table
  - Only columns with the same name in both tables will be displayed
EXCEPT CORR and ALL

- All occurrences of John are retained
- The unmatched duplicate occurrence of Melissa is retained
- The unmatched occurrence of Laura is retained
- Data is sorted

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>Laura</td>
</tr>
<tr>
<td>Melissa</td>
</tr>
</tbody>
</table>
INTERSECT

- Selects *unique* rows common to both tables
- Column labels are ignored

```sql
proc sql;
select * from google
intersect
select * from bing;
quit;
```
INTERSECT

Intersect on Name and Location

- (Melissa CA) (John SC)
- (Laura MA) (John DC)
- (Joseph UK) (Melissa CA) (Michael FL) (Mark IL)
- (Google) (Both) (Bing) (VG Armenia)
- Melissa only appears once
- The second column is labeled based on the first data set
- Data is sorted

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph</td>
<td>London England</td>
</tr>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
</tr>
</tbody>
</table>
INTERSECT ALL

- Selects *unique and duplicate* rows common to both tables
- Column labels are ignored
- Data is sorted
proc sql;
select * from google
intersect all
select * from bing;
quit;

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph</td>
<td>London England</td>
</tr>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
</tr>
</tbody>
</table>
INTERSECT CORR

- Selects *unique* rows common to both tables for matching column names

```sql
proc sql;
select * from google
intersect corr
select * from bing;
quit;
```
INTERSECT CORR and ALL

proc sql;
select * from google
intersect all corr
select * from bing;
run;

- Unique and duplicate rows that appear in both tables will be saved
- Only columns with the same name in both tables will be displayed
UNION

- Selects *unique* rows in either table
- Column labels are ignored
- Rows are sorted

```sql
proc sql;
select * from google
union
select * from bing;
quit;
```
UNION

Union on Name and Location

(Melissa CA)
(John SC)
(Laura MA)
(John DC)

(Melissa CA)
(Michael FL)
(Mark IL)

(Joseph UK)

(Google)

(Unique Union)

© Spring 2012 Imelda Go, John Grego, Jennifer Lasecki and the University of South Carolina
<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Columbia, SC</td>
</tr>
<tr>
<td>John</td>
<td>Washington, DC</td>
</tr>
<tr>
<td>Joseph</td>
<td>London, England</td>
</tr>
<tr>
<td>Laura</td>
<td>Cambridge, MA</td>
</tr>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
</tr>
<tr>
<td>VG</td>
<td>Armenia</td>
</tr>
</tbody>
</table>
UNION and ALL

- Selects *all* rows in either table
- Column labels are ignored
- Not sorted

```sql
proc sql;
select * from google
union all
select * from bing;
quilt;
```
UNION and CORR

- Selects all unique rows in either table based on matching column names

```sql
proc sql;
select * from google
union corr
select * from bing;
run;
```
## UNION and CORR

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
</tr>
<tr>
<td>Joseph</td>
</tr>
<tr>
<td>Laura</td>
</tr>
<tr>
<td>Mark</td>
</tr>
<tr>
<td>Melissa</td>
</tr>
<tr>
<td>Michael</td>
</tr>
<tr>
<td>VG</td>
</tr>
</tbody>
</table>
UNION CORR and ALL

- Unique and duplicate rows that appear in either table will be saved
- Only columns with the same name in both tables will be displayed
- Output is unsorted

```
proc sql;
select * from google
union all corr
select * from bing;
quit;
```
OUTER UNION

- Selects all rows in both tables, but does not overlay any of the information
- Output will have $r_1 + r_2$ rows and $c_1 + c_2$ columns

```sql
proc sql;
select * from google
outer union
select * from bing;
quit;
```
OUTER UNION and CORR

- Using CORR overlays columns with the same name
- More useful than OUTER UNION, but still does not *merge* data

```sql
proc sql;
select * from google
outer union corr
select * from bing;
quit;
```
## OUTER UNION with CORR

<table>
<thead>
<tr>
<th>Name</th>
<th>Residence</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Columbia, SC</td>
<td></td>
</tr>
<tr>
<td>Laura</td>
<td>Cambridge, MA</td>
<td></td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
<td></td>
</tr>
<tr>
<td>Michael</td>
<td>Orlando, FL</td>
<td></td>
</tr>
<tr>
<td>Joseph</td>
<td>London, England</td>
<td></td>
</tr>
<tr>
<td>Melissa</td>
<td>Los Angeles, CA</td>
<td></td>
</tr>
<tr>
<td>Mark</td>
<td>Napierville, IL</td>
<td></td>
</tr>
<tr>
<td>John</td>
<td>Washington, DC</td>
<td></td>
</tr>
<tr>
<td>Mark</td>
<td></td>
<td>Napierville, IL</td>
</tr>
<tr>
<td>Joseph</td>
<td></td>
<td>London, England</td>
</tr>
<tr>
<td>Melissa</td>
<td></td>
<td>Orlando, FL</td>
</tr>
<tr>
<td>Michael</td>
<td></td>
<td>Los Angeles, CA</td>
</tr>
<tr>
<td>VG</td>
<td></td>
<td>Yerevan, Armenia</td>
</tr>
</tbody>
</table>