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

Name	Residence
John	Columbia, SC
Laura	Cambridge, MA
Melissa	Los Angeles, CA
Michael	Orlando, FL
Joseph	London, England
Melissa	Los Angeles, CA
Mark	Napierville, IL
John	Washington, DC

Bing Grego

Name	City
Mark	Napierville, IL
Joseph	London, England
Michael	Orlando, FL
Melissa	Los Angeles, CA
VG	Yerevan, Armenia

Combining Tables Vertically

- Standard syntax:

```
proc sql;  select *
from google
(except/intersect/union/outer union)
(corr/all)
select * from bing;
```

Combining Tables Vertically

- 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

EXCEPT

```
proc sql;
select * from google
except
select * from bing;
quit;
```

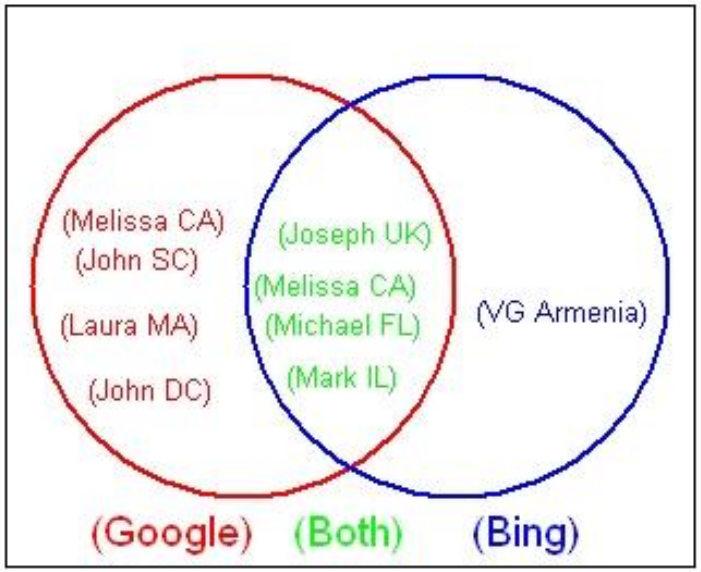
Name	Residence
John	Columbia, SC
John	Washington, DC
Laura	Cambridge, MA

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

Intersect on Name and Location



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

Name	Residence
John	Columbia, SC
John	Washington, DC
Laura	Cambridge, MA
Melissa	Los Angeles, CA

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

EXCEPT CORR

```
proc sql;
select * from google
except corr
select * from bing;
quit;
```

Name
John
Laura

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

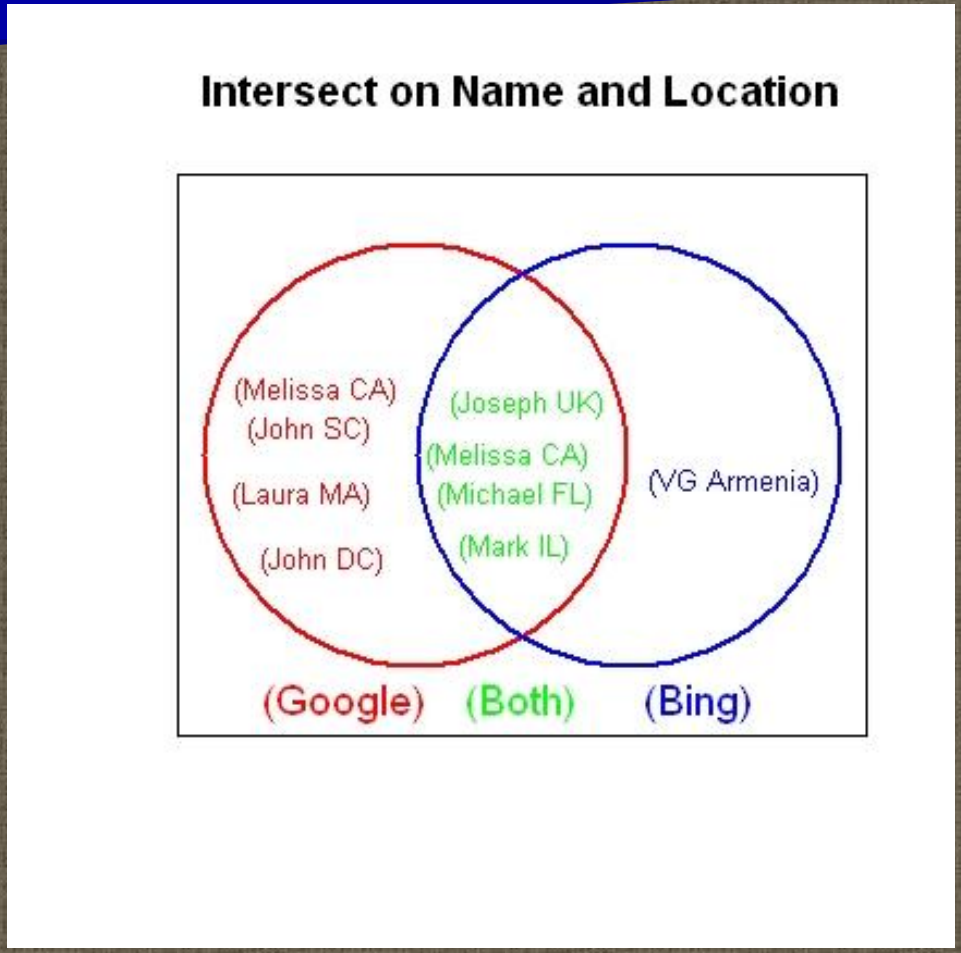
Name
John
John
Laura
Melissa

INTERSECT

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

```
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)
(VG Armenia)
(Google)
(Both)
(Bing)

INTERSECT

- Melissa only appears once
- The second column is labeled based on the first data set
- Data is sorted

Name	Residence
Joseph	London England
Mark	Napierville, IL
Melissa	Los Angeles, CA
Michael	Orlando, FL

INTERSECT ALL

- Selects *unique and duplicate* rows common to both tables
- Column labels are ignored
- Data is sorted

INTERSECT ALL

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

Name	Residence
Joseph	London England
Mark	Napierville, IL
Melissa	Los Angeles, CA
Michael	Orlando, FL

INTERSECT CORR

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

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

Name
Joseph
Mark
Melissa
Michael

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

Name
Joseph
Mark
Melissa
Michael

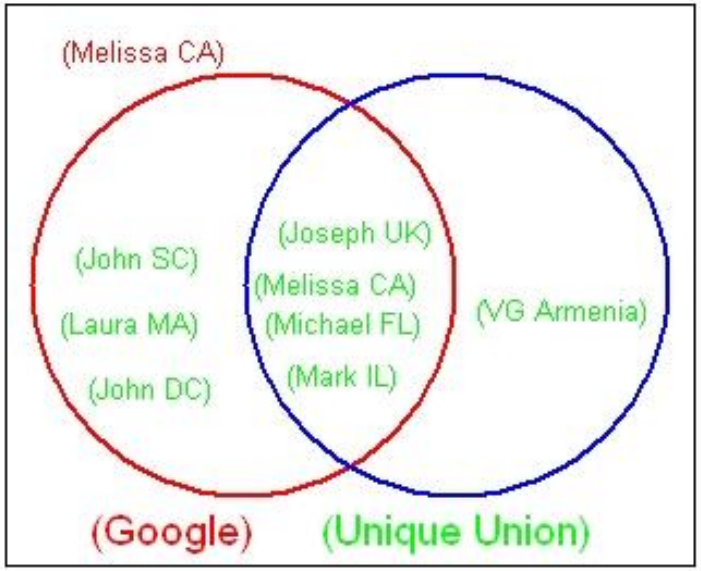
UNION

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

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

UNION

Union on Name and Location



UNION

Name	Residence
John	Columbia, SC
John	Washington, DC
Joseph	London, England
Laura	Cambridge, MA
Mark	Napierville, IL
Melissa	Los Angeles, CA
Michael	Orlando, FL
VG	Armenia

UNION and ALL

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

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

UNION and CORR

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

```
proc sql;
select * from google
union corr
select * from bing;
run;
```

UNION and CORR

Name
John
Joseph
Laura
Mark
Melissa
Michael
VG

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

```
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

```
proc sql;
select * from google
outer union corr
select * from bing;
quit;
```

OUTER UNION with CORR

Name	Residence	City
John	Columbia, SC	
Laura	Cambridge, MA	
Melissa	Los Angeles, CA	
Michael	Orlando, FL	
Joseph	London, England	
Melissa	Los Angeles, CA	
Mark	Napierville, IL	
John	Washington, DC	
Mark		Napierville, IL
Joseph		London, England
Melissa		Orlando, FL
Michael		Los Angeles, CA
VG		Yerevan, Armenia