Build your Data Skills: Introduction to SQL

Kaeli Samson, MA, MPH
Department of Biostatistics
College of Public Health
University of Nebraska Medical Center

June 25th, 2019

University of Nebraska Medical Center

CSP

New Orleans, Louisiana
February 14–16 2019

Hadley Wickham, PhD
Chief Scientist, RStudio

Top 3 Skills:

1. SQL
2. Github
3. Marketing

Overview

- Brief Intro to SQL
- Terminology
- General syntax/structure
- Description of Dataset
- Basic Queries
- Creating New Variables
- Joins
- Helpful SQL Code

Brief Introduction to SQL

SQL = Structured Query Language

Typically associated with use in database management, but also great for data management, generally!

Brief Background in Database Design

Student	Student Contact	Course	Course Description	Instructor	Instructor Contact
Josie	555-1234	Calculus II	Integration	Julie	555-8888
Ken	555-9845	Calculus II	Integration	Joe	555-2222
Brooke	555-7878	Calculus II	Integration	Julie	555-8888
Addison	555-1111	Calculus II	Integration	Julie	555-8888
Cole	555-6127	Calculus II	Integration	Julie	555-8888
Samantha	555-1534	Calculus II	Integration	Joe	555-2222
Josh	555-5463	Calculus II	Integration	Joe	555-2222
Josie	555-1234	GIS I	Mapping	Paul	555-3333
Ken	555-9845	GIS I	Mapping	Paul	555-3333

Brief Background in Database Design

Student ID	Student Name	Student Contact
1	Josie	555-1234
2	Ken	555-9845
3	Brooke	555-7878
4	Addison	555-1111
5	Cole	555-6127
6	Samantha	555-1534
7	Josh	555-5463

Instructor ID	Instructor Name	Instructor Contact
1	Julie	555-8888
2	Joe	555-2222
3	Paul	555-3333

Course ID	Course Name	Course Description
A	Calculus II	Integration
B	GIS I	Mapping

Enrollment ID	Student ID	Course ID	Instructor ID
1	1	A	1
2	2	A	2
3	1	B	3

Brief Introduction to SQL

SQL can be used to “query” data, but can do more, such as:

- Create new variables
- Join tables together
- Insert observations
- Edit observations
- Delete observations

Terminology

University of Nebraska
Medical Center

S-Q-L vs. Sequel?

Terminology

SAS	SQL
Dataset	Table
Observation	Row
Variable	Column

Note: Since this presentation uses SAS to run SQL, SAS terms will be used interchangeably with SQL terms, although I acknowledge in some fields of study these terms are not considered synonymous.

General Structure and Syntax of SQL

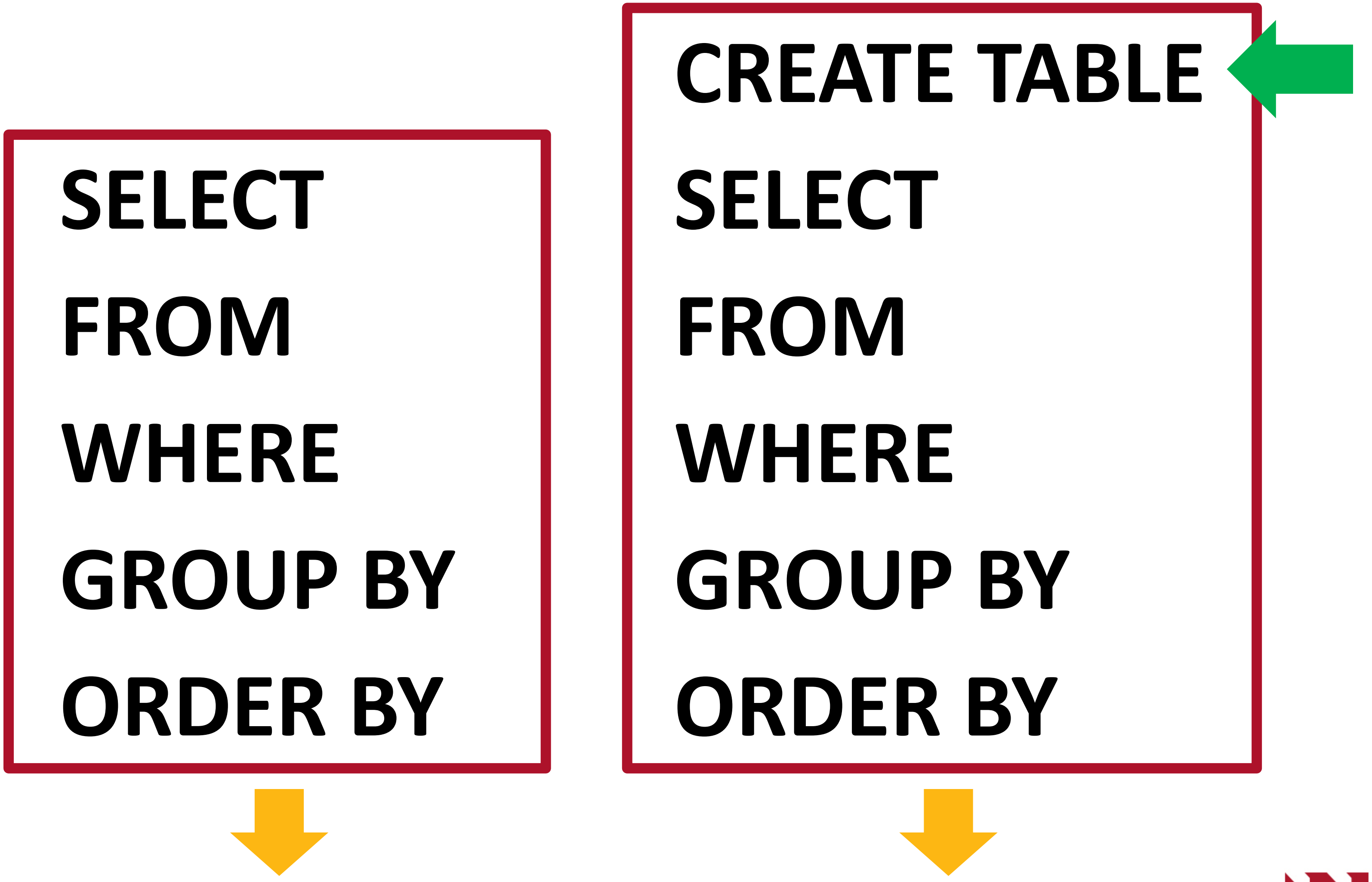
University of Nebraska Medical Center

Common SQL Clauses

SELECT	Choose variables/columns for your table
FROM	Indicate source(s) of data (i.e. datasets)
WHERE	Subsetting criteria for rows
GROUP BY	Grouping desired for summary variables
ORDER BY	Sort order for rows

Clauses must be in this specific order!

Common SQL Clauses



SELECT

FROM

WHERE

GROUP BY

ORDER BY

Prints query result (i.e. table) to output

CREATE TABLE

SELECT

FROM

WHERE

GROUP BY

ORDER BY

Saves query result (i.e. table) as SAS dataset

SQL in SAS

```
PROC SQL ;
        SELECT
        FROM
        WHERE
        GROUP BY
        ORDER BY ;
QUIT ;
```

Specific to SAS

SQL in SAS

Note: While SQL is an ANSI standard language, each software that runs it, including SAS, may have their own options that are specific to that software. As such, some of the code in this presentation may not work outside SAS, but the general principles will still apply.

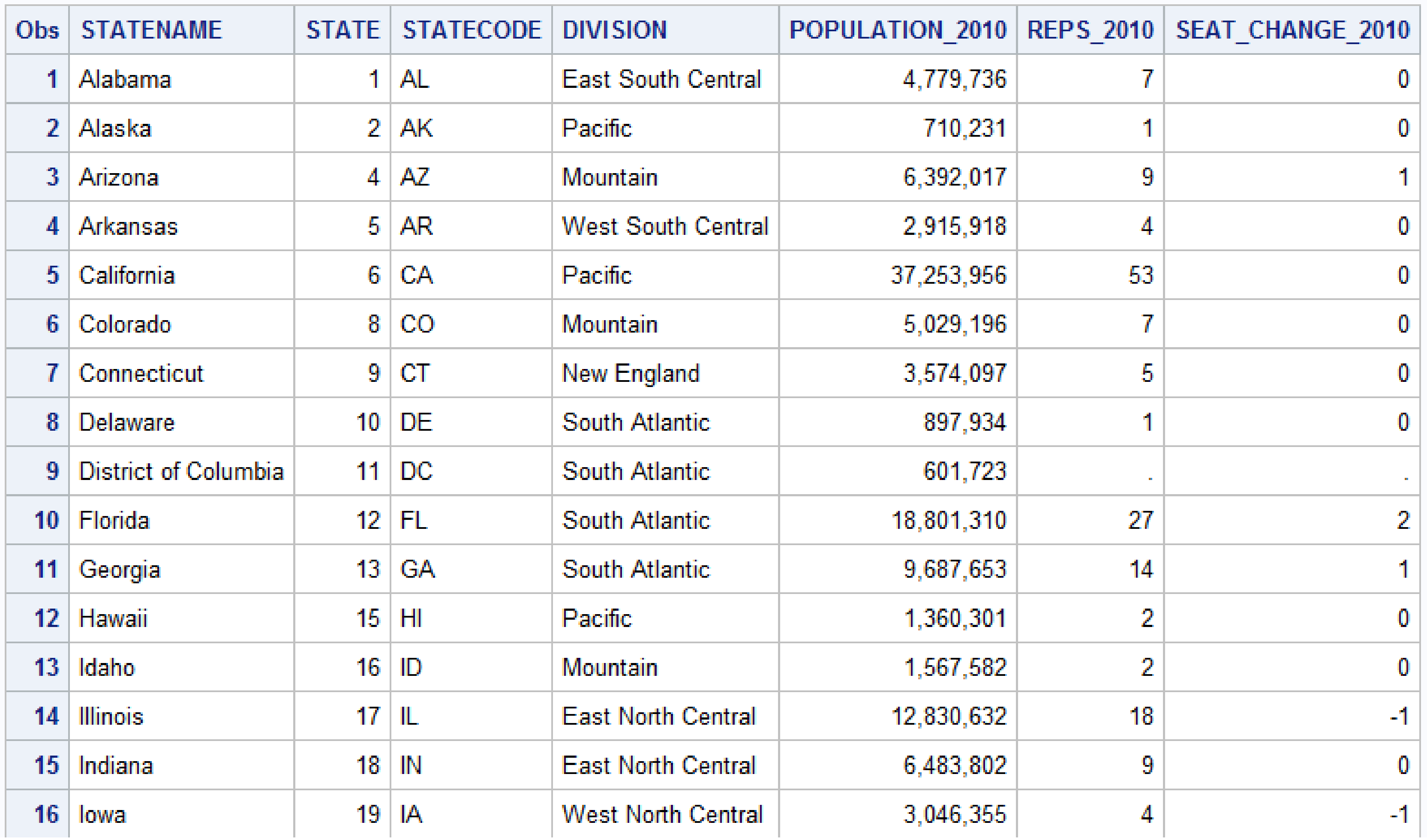
Structure of Example Dataset

University of Nebraska Medical Center

Data: sashelp.us_data

- **State Identifiers** (x3)
  - Name
  - Abbreviation
  - FIPS Code
- **State Information**
  - Division
  - Population
  - Number of representatives
  - Change in number of seats

sashelp.us_data (abbr.)

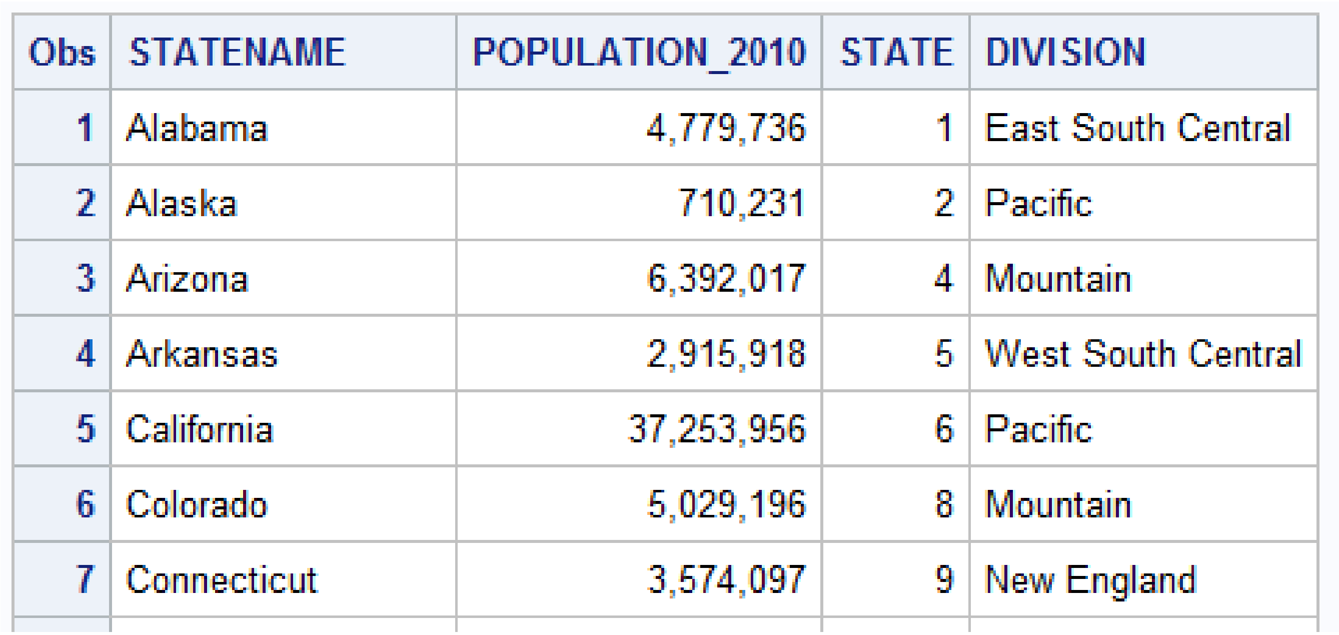


Obs	STATENAME	STATE	STATECODE	DIVISION	POPULATION_2010	REPS_2010	SEAT_CHANGE_2010
1	Alabama	1	AL	East South Central	4,779,736	7	0
2	Alaska	2	AK	Pacific	710,231	1	0
3	Arizona	4	AZ	Mountain	6,392,017	9	1
4	Arkansas	5	AR	West South Central	2,915,918	4	0
5	California	6	CA	Pacific	37,253,956	53	0
6	Colorado	8	CO	Mountain	5,029,196	7	0
7	Connecticut	9	CT	New England	3,574,097	5	0
8	Delaware	10	DE	South Atlantic	897,934	1	0
9	District of Columbia	11	DC	South Atlantic	601,723	.	.
10	Florida	12	FL	South Atlantic	18,801,310	27	2
11	Georgia	13	GA	South Atlantic	9,687,653	14	1
12	Hawaii	15	HI	Pacific	1,360,301	2	0
13	Idaho	16	ID	Mountain	1,567,582	2	0
14	Illinois	17	IL	East North Central	12,830,632	18	-1
15	Indiana	18	IN	East North Central	6,483,802	9	0
16	Iowa	19	IA	West North Central	3,046,355	4	-1

New Dataset

```
data us_pop;
   set sashelp.us_data
      (rename=(population_2010=pop_2010));
   keep statename state division pop_2010;
run;
```

us_pop



Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

Basic Queries

University of Nebraska
Medical Center

Basic Structure of SQL Code

Printing all variables and observations in a dataset

SQL

```
proc sql;
   select var1, var2, var3, var4
   from dataset;
quit;
```

Basic Structure of SQL Code

Printing all variables and observations in a dataset

dataset

Traditional SAS Code:

```
proc print data=us_pop;
   var statename pop_2010 state division;
run;
```

variables

variables

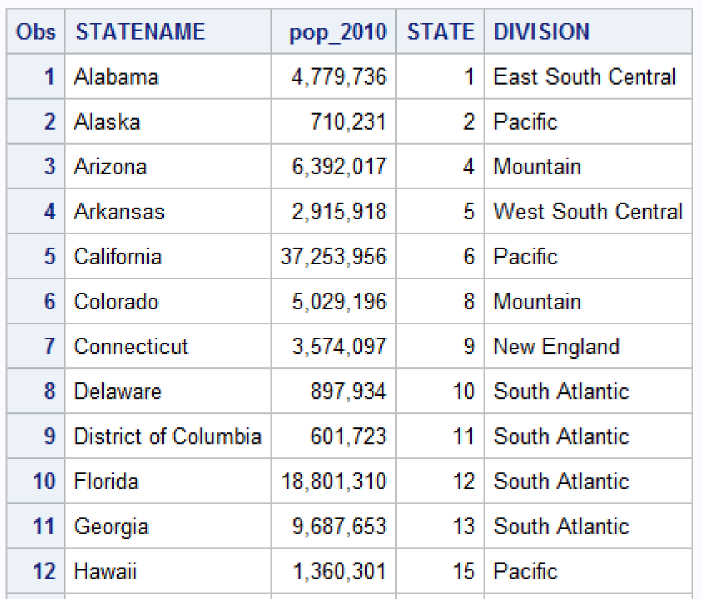
```
proc sql;
   select statename, pop_2010, state, division
   from us_pop;
quit;
```

dataset

Basic Structure of SQL Code

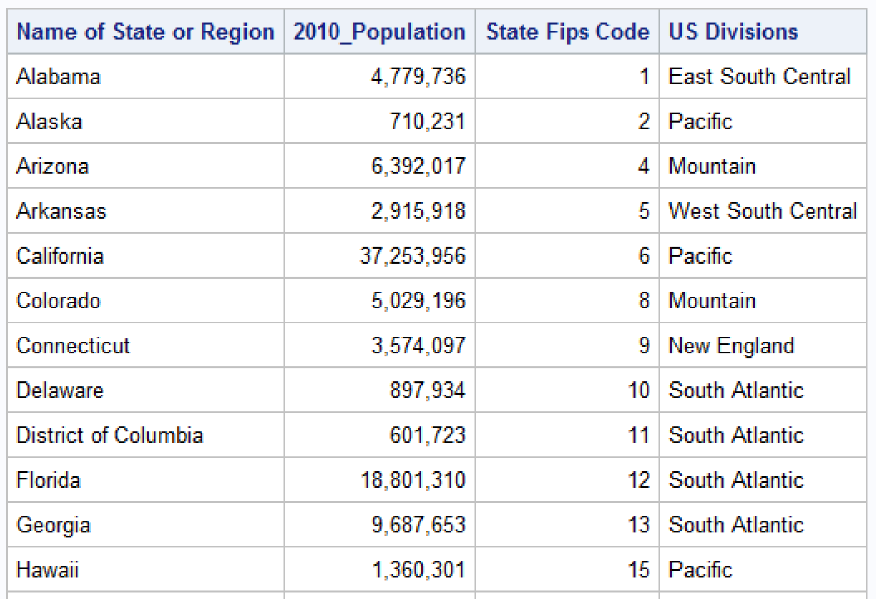
Printing all variables and observations in a dataset

Proc Print



Obs	STATENAME	pop_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England
8	Delaware	897,934	10	South Atlantic
9	District of Columbia	601,723	11	South Atlantic
10	Florida	18,801,310	12	South Atlantic
11	Georgia	9,687,653	13	South Atlantic
12	Hawaii	1,360,301	15	Pacific

Proc SQL



Name of State or Region	2010_Population	State Fips Code	US Divisions
Alabama	4,779,736	1	East South Central
Alaska	710,231	2	Pacific
Arizona	6,392,017	4	Mountain
Arkansas	2,915,918	5	West South Central
California	37,253,956	6	Pacific
Colorado	5,029,196	8	Mountain
Connecticut	3,574,097	9	New England
Delaware	897,934	10	South Atlantic
District of Columbia	601,723	11	South Atlantic
Florida	18,801,310	12	South Atlantic
Georgia	9,687,653	13	South Atlantic
Hawaii	1,360,301	15	Pacific

Basic Structure of SQL Code

Printing all variables and observations in a dataset

select all variables

```
proc sql;
  select *
  from dataset;
quit;
```

Basic Structure of SQL Code

Printing all observations in a dataset

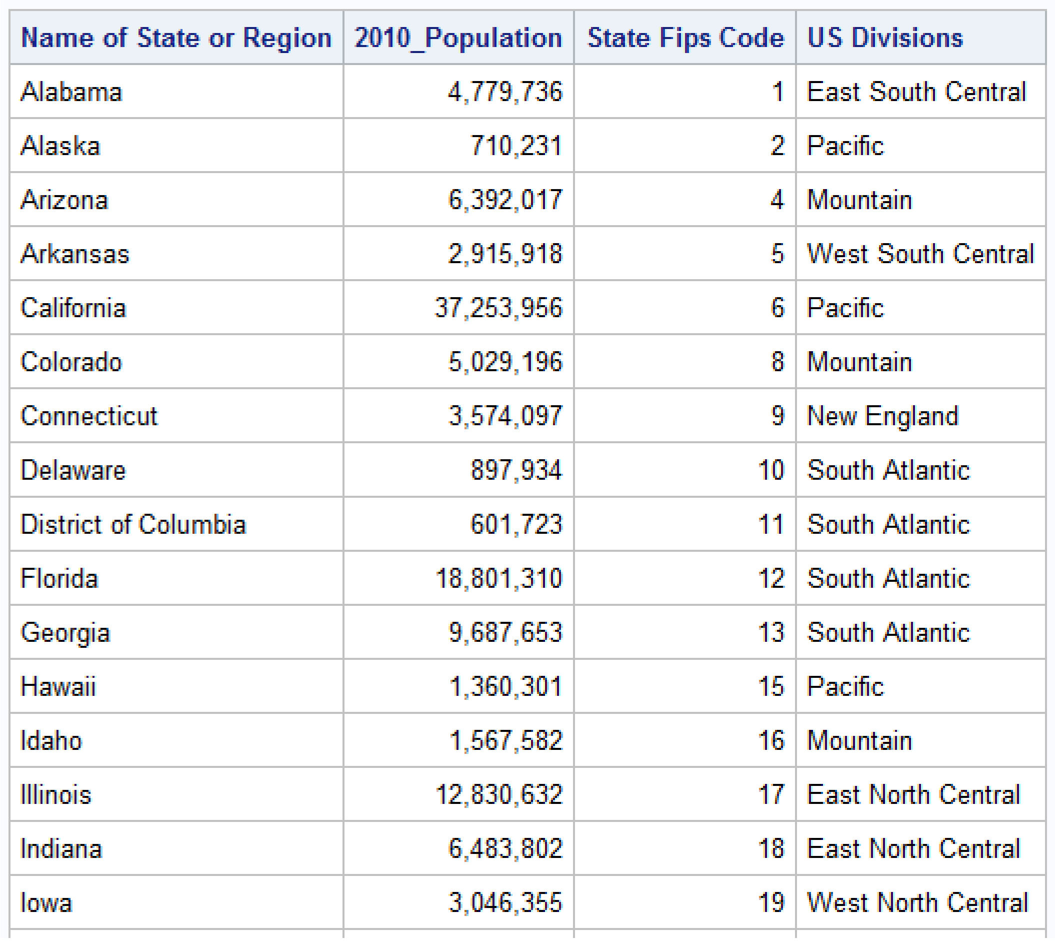
Traditional SAS Code:

```
      proc print data=us_pop;
      run;
```

```
      proc sql;
         select *
         from us_pop;
      quit;
```

Basic Structure of SQL Code

Printing all observations in a dataset



Name of State or Region	2010_Population	State Fips Code	US Divisions
Alabama	4,779,736	1	East South Central
Alaska	710,231	2	Pacific
Arizona	6,392,017	4	Mountain
Arkansas	2,915,918	5	West South Central
California	37,253,956	6	Pacific
Colorado	5,029,196	8	Mountain
Connecticut	3,574,097	9	New England
Delaware	897,934	10	South Atlantic
District of Columbia	601,723	11	South Atlantic
Florida	18,801,310	12	South Atlantic
Georgia	9,687,653	13	South Atlantic
Hawaii	1,360,301	15	Pacific
Idaho	1,567,582	16	Mountain
Illinois	12,830,632	17	East North Central
Indiana	6,483,802	18	East North Central
Iowa	3,046,355	19	West North Central

Basic Structure of SQL Code

Printing unique observations in a dataset

```
proc sql;
  select division
  from us_pop;
quit;
```

only select unique observations

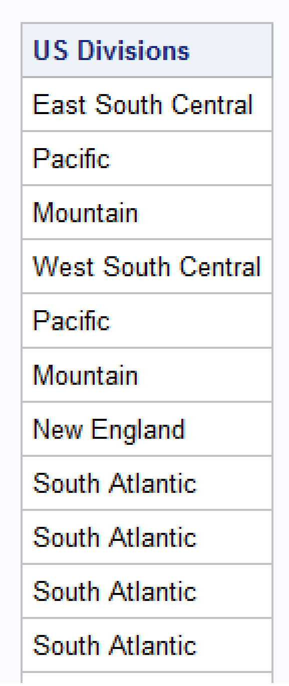
```
proc sql;
  select distinct division
  from us_pop;
quit;
```

Basic Structure of SQL Code

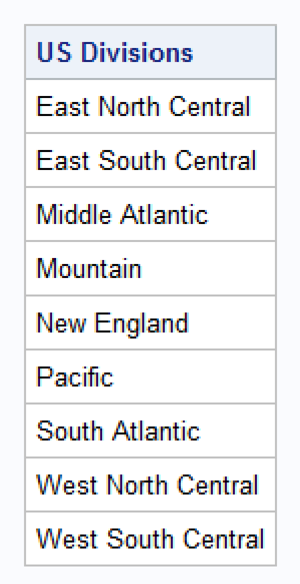
Printing unique observations in a dataset

```
proc sql;
    select division
    from us_pop;
quit;
```

```
proc sql;
    select distinct division
    from us_pop;
quit;
```



US Divisions
East South Central
Pacific
Mountain
West South Central
Pacific
Mountain
New England
South Atlantic
South Atlantic
South Atlantic
South Atlantic



US Divisions
East North Central
East South Central
Middle Atlantic
Mountain
New England
Pacific
South Atlantic
West North Central
West South Central

Basic Structure of SQL Code

Printing a subset of observations

```
proc sql;
   select *
   from dataset
   where var in (“A”, “B”, “C”);
quit;
```

restrict observations
with where clause

Basic Structure of SQL Code

Printing a subset of observations

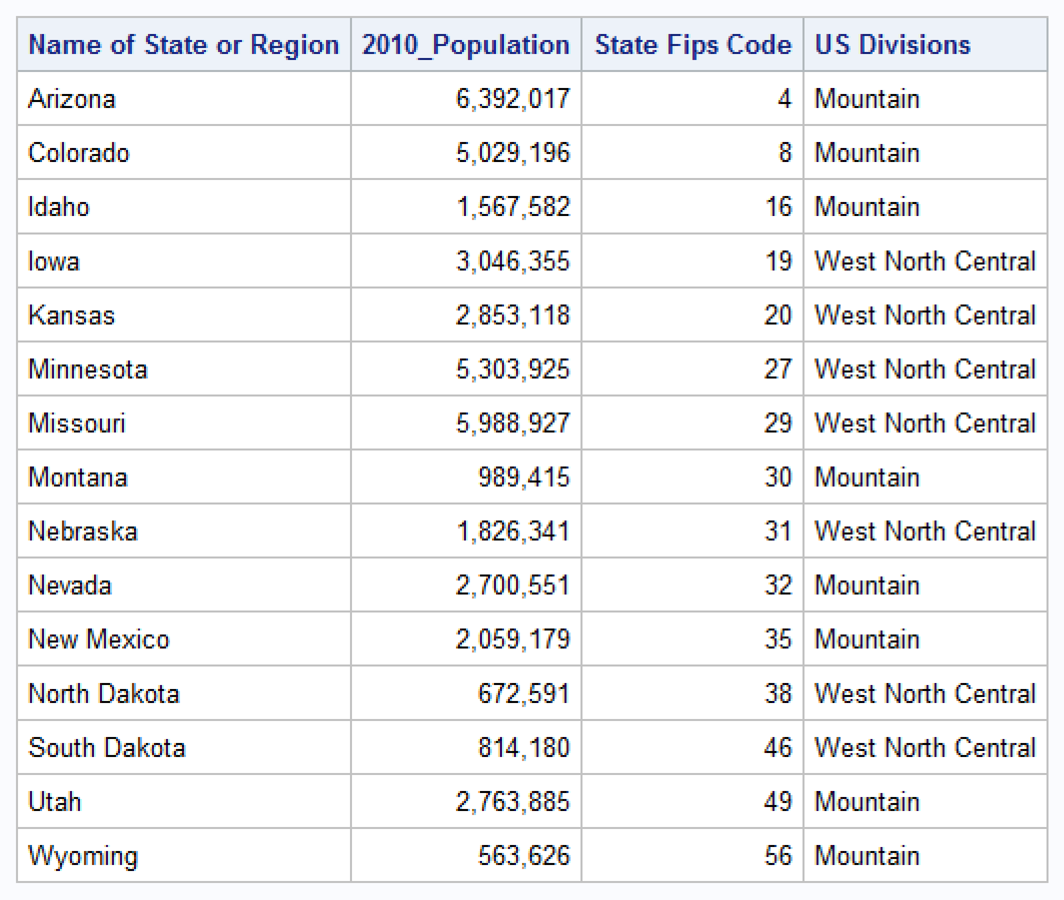
Traditional SAS Code:

```
proc print data=us_pop;
    where division in ("West North Central", "Mountain");
run;
```

```
proc sql;
    select *
    from us_pop
    where division in ("West North Central", "Mountain");
quit;
```

Basic Structure of SQL Code

Printing a subset of observations



Name of State or Region	2010_Population	State Fips Code	US Divisions
Arizona	6,392,017	4	Mountain
Colorado	5,029,196	8	Mountain
Idaho	1,567,582	16	Mountain
Iowa	3,046,355	19	West North Central
Kansas	2,853,118	20	West North Central
Minnesota	5,303,925	27	West North Central
Missouri	5,988,927	29	West North Central
Montana	989,415	30	Mountain
Nebraska	1,826,341	31	West North Central
Nevada	2,700,551	32	Mountain
New Mexico	2,059,179	35	Mountain
North Dakota	672,591	38	West North Central
South Dakota	814,180	46	West North Central
Utah	2,763,885	49	Mountain
Wyoming	563,626	56	Mountain

Basic Structure of SQL Code

Other examples of where clause in SQL

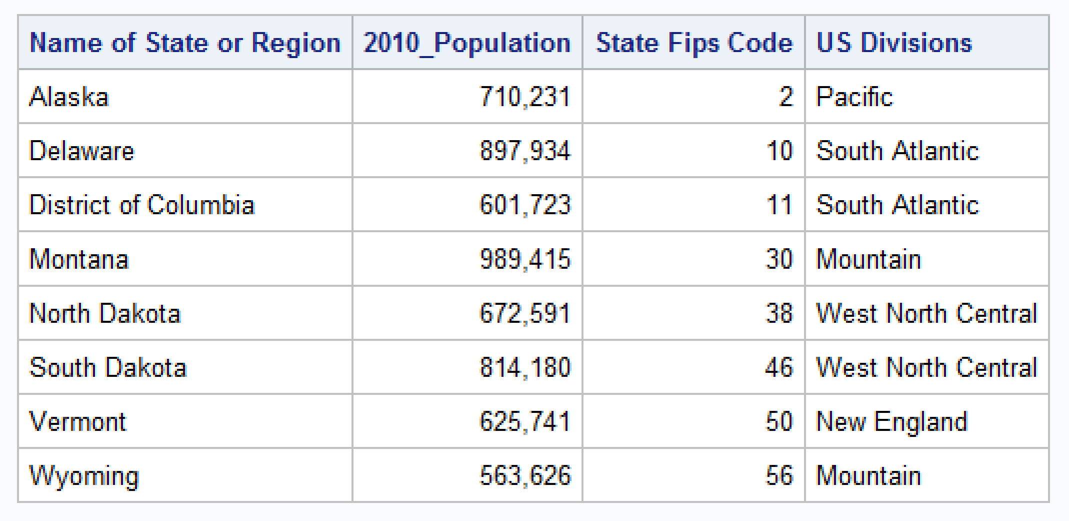
```
proc sql;
    select *
    from us_pop
    where pop_2010 between 0 and 1000000;
quit;
```

```
proc sql;
    select *
    from us_pop
    where 0 le pop_2010 le 1000000 and division = "Mountain";
quit;
```

Basic Structure of SQL Code

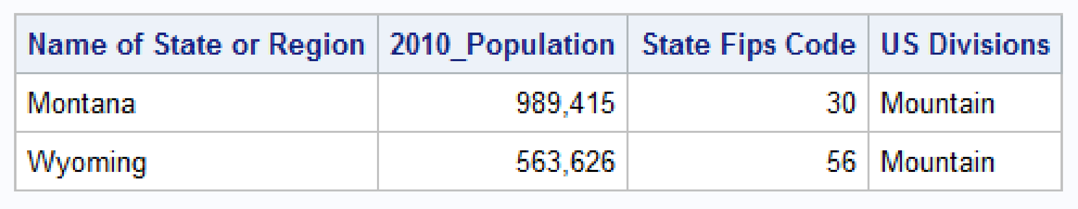
Other examples of where clause in SQL

```
where pop_2010 between 0 and 1000000;
```



Name of State or Region	2010_Population	State Fips Code	US Divisions
Alaska	710,231	2	Pacific
Delaware	897,934	10	South Atlantic
District of Columbia	601,723	11	South Atlantic
Montana	989,415	30	Mountain
North Dakota	672,591	38	West North Central
South Dakota	814,180	46	West North Central
Vermont	625,741	50	New England
Wyoming	563,626	56	Mountain

```
where 0 le pop_2010 le 1000000 and division = "Mountain";
```



Name of State or Region	2010_Population	State Fips Code	US Divisions
Montana	989,415	30	Mountain
Wyoming	563,626	56	Mountain

Basic Structure of SQL Code

Sort observations in a dataset

```
proc sql;
    select *
    from dataset
    order by var;
quit;
```

```
proc sql;
    select *
    from dataset
    order by var1, var2 desc;
quit;
```

sort output using ‘order by’ clause

Basic Structure of SQL Code

Sort observations in a dataset

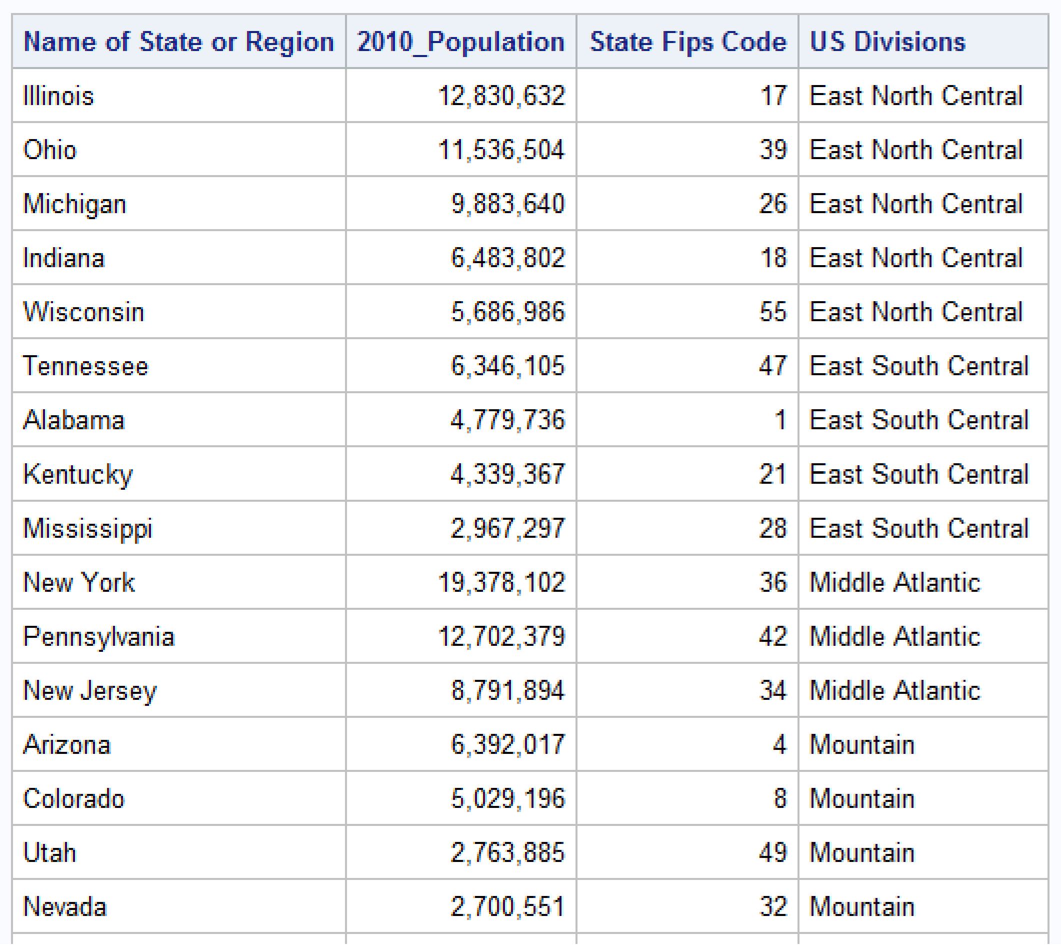
Traditional SAS Code:

```
proc sort data=us_pop;
   by division descending population_2010;
run;

proc print data=us_pop;
run;
```

```
proc sql;
   select *
   from us_pop
   order by division, population_2010 desc;
quit;
```

Basic Structure of SQL Code

Sort observations in a dataset



Name of State or Region	2010_Population	State Fips Code	US Divisions
Illinois	12,830,632	17	East North Central
Ohio	11,536,504	39	East North Central
Michigan	9,883,640	26	East North Central
Indiana	6,483,802	18	East North Central
Wisconsin	5,686,986	55	East North Central
Tennessee	6,346,105	47	East South Central
Alabama	4,779,736	1	East South Central
Kentucky	4,339,367	21	East South Central
Mississippi	2,967,297	28	East South Central
New York	19,378,102	36	Middle Atlantic
Pennsylvania	12,702,379	42	Middle Atlantic
New Jersey	8,791,894	34	Middle Atlantic
Arizona	6,392,017	4	Mountain
Colorado	5,029,196	8	Mountain
Utah	2,763,885	49	Mountain
Nevada	2,700,551	32	Mountain

Creating New Variables

University of Nebraska Medical Center

Creating New Variables

Creating new variables

new variable definition

new variable name

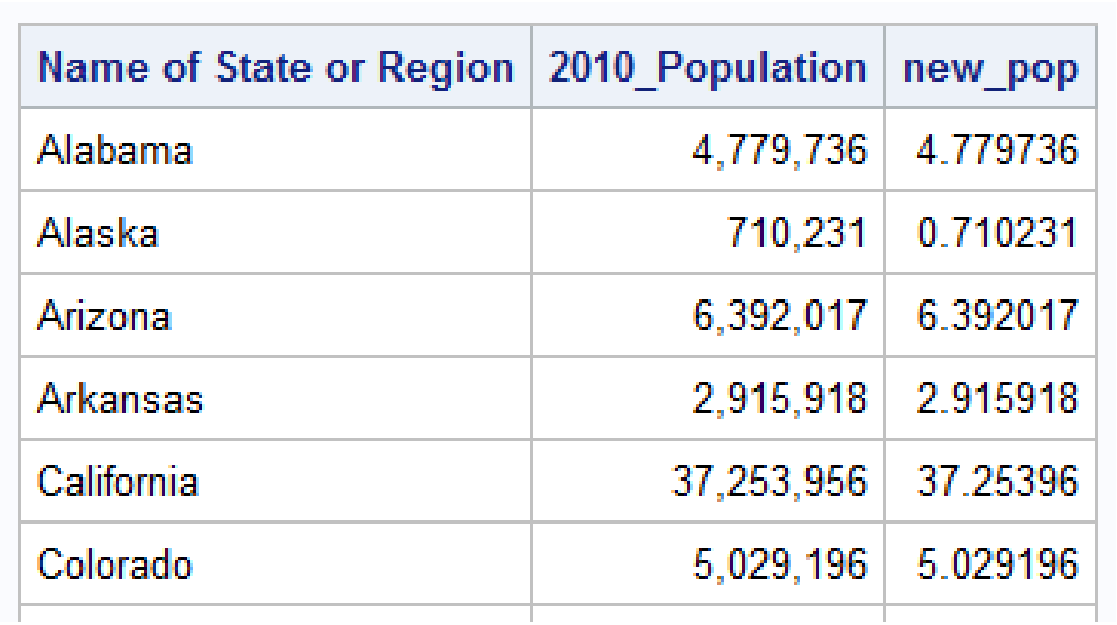
```
proc sql;
    select statename, pop_2010, pop_2010/1000000 as new_pop
    from us_pop;
quit;
```

(not optional)

Creating New Variables

Creating new variables

```
proc sql;
    select statename, pop_2010, pop_2010/1000000 as new_pop
    from us_pop;
quit;
```



Name of State or Region	2010_Population	new_pop
Alabama	4,779,736	4.779736
Alaska	710,231	0.710231
Arizona	6,392,017	6.392017
Arkansas	2,915,918	2.915918
California	37,253,956	37.25396
Colorado	5,029,196	5.029196

Creating New Variables

Cleaning up new variables

new variable format

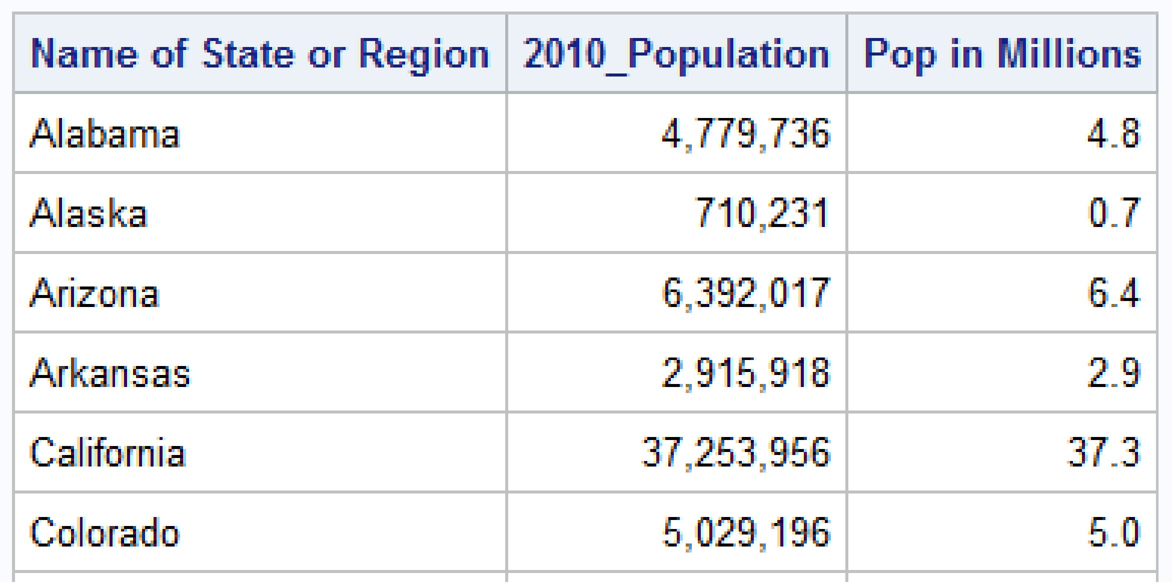
```
proc sql;
    select
        statename,
        pop_2010,
        pop_2010/1000000 as new_pop format=8.1 label="Pop in Millions"
    from us_pop;
quit;
```

new variable label

Creating New Variables

Cleaning up new variables

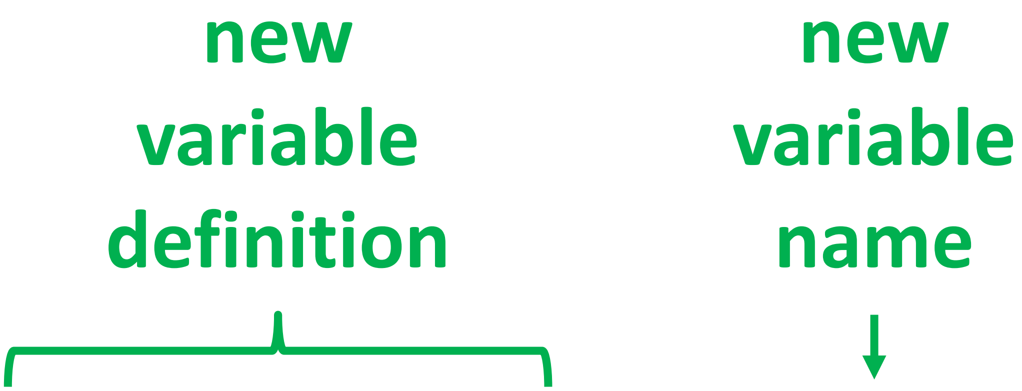
```
proc sql;
    select
        statename,
        pop_2010,
        pop_2010/1000000 as new_pop format=8.1 label="Pop in Millions"
    from us_pop;
quit;
```



Name of State or Region	2010_Population	Pop in Millions
Alabama	4,779,736	4.8
Alaska	710,231	0.7
Arizona	6,392,017	6.4
Arkansas	2,915,918	2.9
California	37,253,956	37.3
Colorado	5,029,196	5.0

Creating New Variables

Summary Variables



new variable definition

new variable name

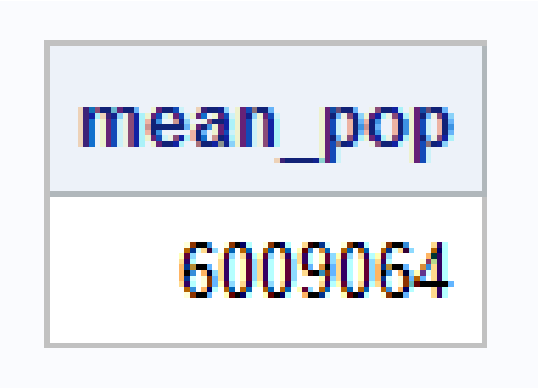
```
proc sql;
   select mean(pop_2010) as mean_pop
   from us_pop;
quit;
```

Creating New Variables

Summary Variables

```
proc sql;
   select mean(pop_2010) as mean_pop
   from us_pop;
quit;
```

The summary function is applied to the entire data set (when there is no group by clause) →

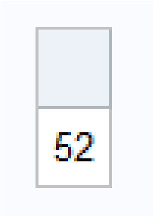


mean_pop
6009064

Creating New Variables

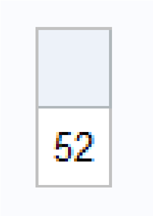
Using the count function

```
proc sql;
   select count(*)
   from us_pop;
quit;
```



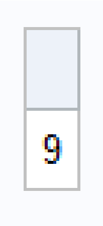
52

```
proc sql;
   select count(division)
   from us_pop;
run;
```



52

```
proc sql;
   select count(distinct division)
   from us_pop;
run;
```



9

Creating New Variables

Summary Variables by Group

new variable definition

new variable name

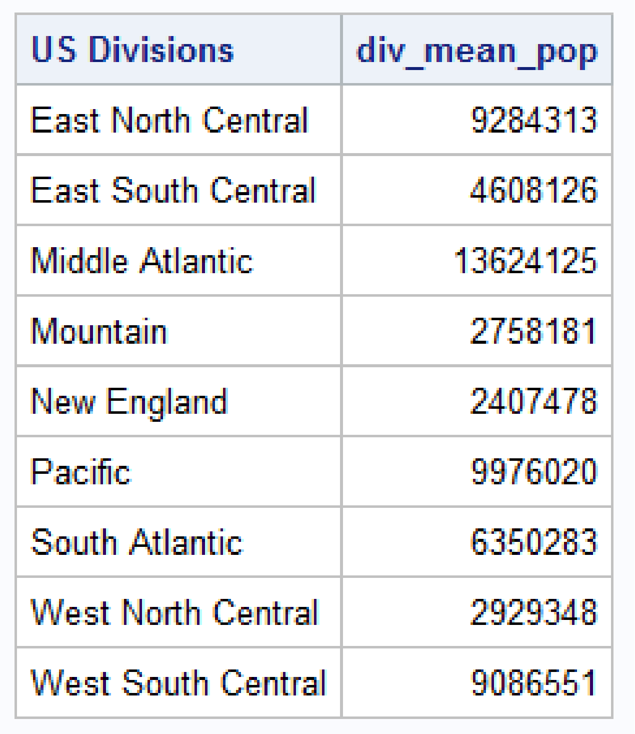
```
proc sql;
   select division, mean(pop_2010) as div_mean_pop
   from us_pop
   group by division;
quit;
```

Grouping variable: will calculate summary statistics for each unique value of this variable

Creating New Variables

Summary Variables by Group

```
proc sql;
   select division, mean(pop_2010) as div_mean_pop
   from us_pop
   group by division;
quit;
```



US Divisions	div_mean_pop
East North Central	9284313
East South Central	4608126
Middle Atlantic	13624125
Mountain	2758181
New England	2407478
Pacific	9976020
South Atlantic	6350283
West North Central	2929348
West South Central	9086551

Creating New Variables

Summary Variables by Group

Note: It’s important to have your grouping variable in both your select and your group by clauses!

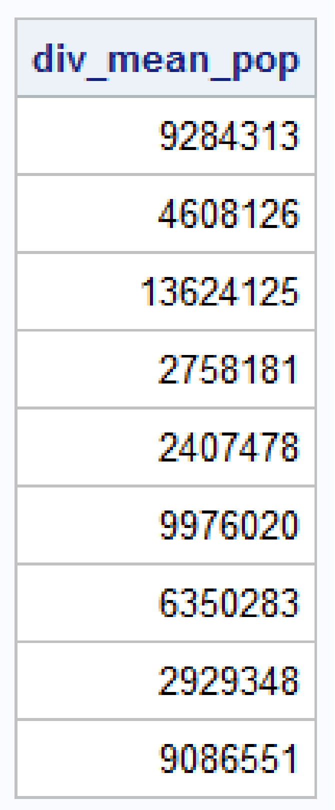
```
proc sql;
   select division, mean(pop_2010) as div_mean_pop
   from us_pop
   group by division;
quit;
```

Creating New Variables

Summary Variables by Group

This is what happens if you leave the grouping variable out of the select clause…

```
proc sql;
    select mean(pop_2010) as div_mean_pop
    from us_pop
    group by division;
quit;
```



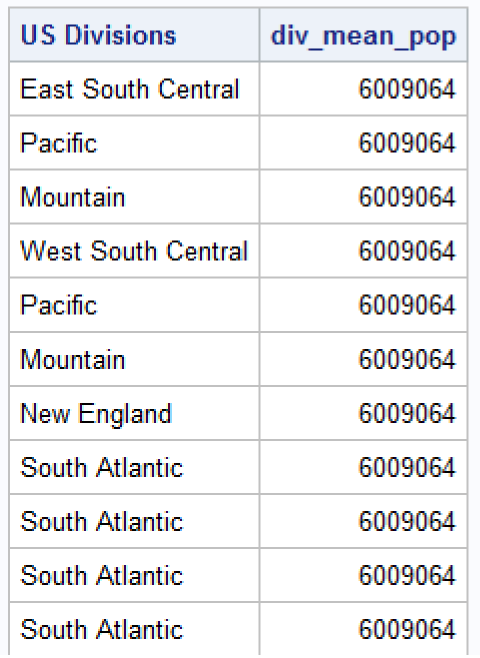
div_mean_pop
9284313
4608126
13624125
2758181
2407478
9976020
6350283
2929348
9086551

Creating New Variables

Summary Variables by Group

This is what happens if you forget to includes the group by clause:

```
proc sql;
    select division, mean(pop_2010) as div_mean_pop
    from us_pop;
quit;
```



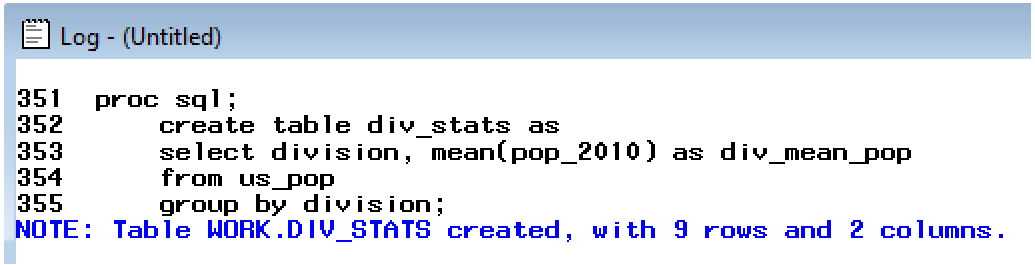
US Divisions	div_mean_pop
East South Central	6009064
Pacific	6009064
Mountain	6009064
West South Central	6009064
Pacific	6009064
Mountain	6009064
New England	6009064
South Atlantic	6009064
South Atlantic	6009064
South Atlantic	6009064
South Atlantic	6009064

Creating New SAS Datasets

new SAS dataset name

not optional

```
proc sql;
   create table div_stats as
   select division, mean(pop_2010) as div_mean_pop
   from us_pop
   group by division;
quit;
```

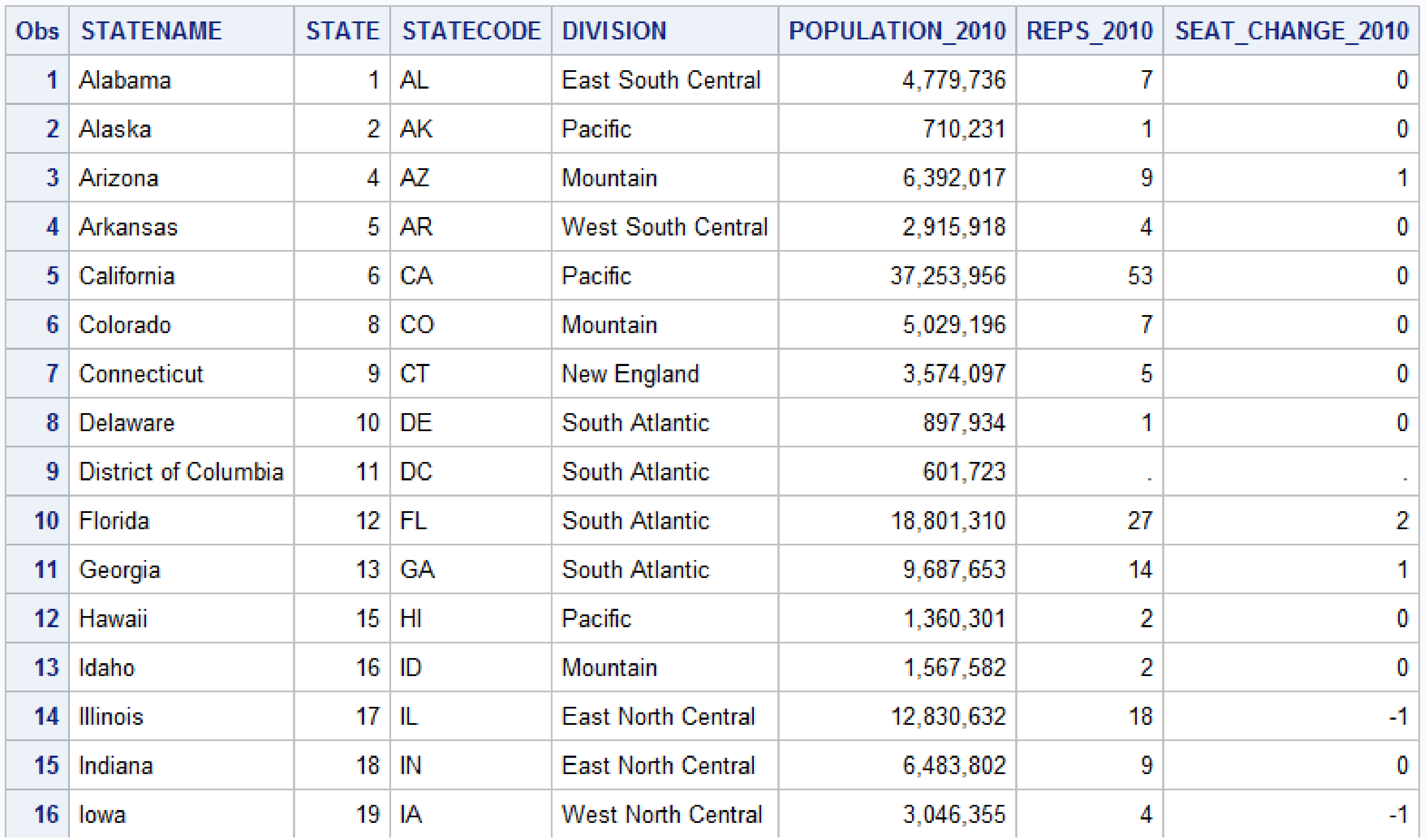


```
351  proc sql;
352       create table div_stats as
353       select division, mean(pop_2010) as div_mean_pop
354       from us_pop
355       group by division;
NOTE: Table WORK.DIV_STATS created, with 9 rows and 2 columns.
```

Joins

University of Nebraska
Medical Center

sashelp.us_data (abbr.)



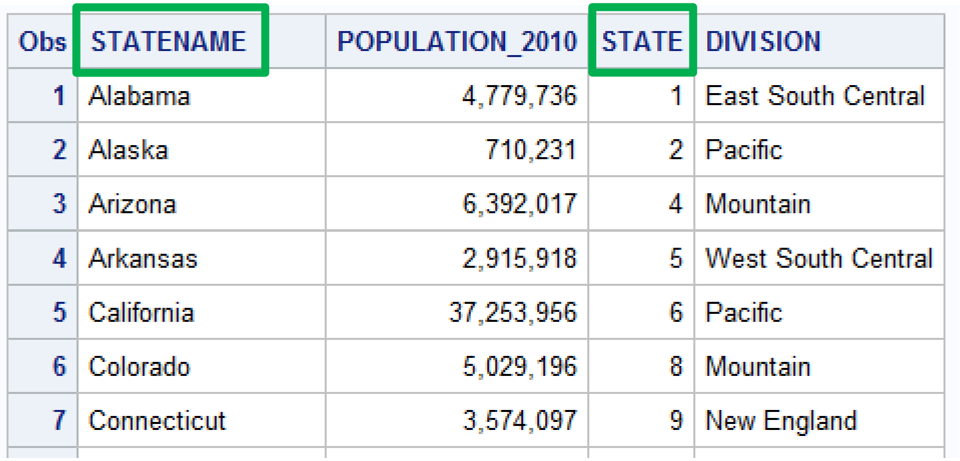
Obs	STATENAME	STATE	STATECODE	DIVISION	POPULATION_2010	REPS_2010	SEAT_CHANGE_2010
1	Alabama	1	AL	East South Central	4,779,736	7	0
2	Alaska	2	AK	Pacific	710,231	1	0
3	Arizona	4	AZ	Mountain	6,392,017	9	1
4	Arkansas	5	AR	West South Central	2,915,918	4	0
5	California	6	CA	Pacific	37,253,956	53	0
6	Colorado	8	CO	Mountain	5,029,196	7	0
7	Connecticut	9	CT	New England	3,574,097	5	0
8	Delaware	10	DE	South Atlantic	897,934	1	0
9	District of Columbia	11	DC	South Atlantic	601,723	.	.
10	Florida	12	FL	South Atlantic	18,801,310	27	2
11	Georgia	13	GA	South Atlantic	9,687,653	14	1
12	Hawaii	15	HI	Pacific	1,360,301	2	0
13	Idaho	16	ID	Mountain	1,567,582	2	0
14	Illinois	17	IL	East North Central	12,830,632	18	-1
15	Indiana	18	IN	East North Central	6,483,802	9	0
16	Iowa	19	IA	West North Central	3,046,355	4	-1

New Datasets

```
data us_pop;
    set sashelp.us_data
       (rename=(population_2010=pop_2010));
    keep statename state division pop_2010;
run;

data us_rep;
    set sashelp.us_data (keep=
           state
           statecode
           reps_2010);
run;

data us_seatch;
    set sashelp.us_data (keep=
           statecode
           statename
           seat_change_2010);
run;
```

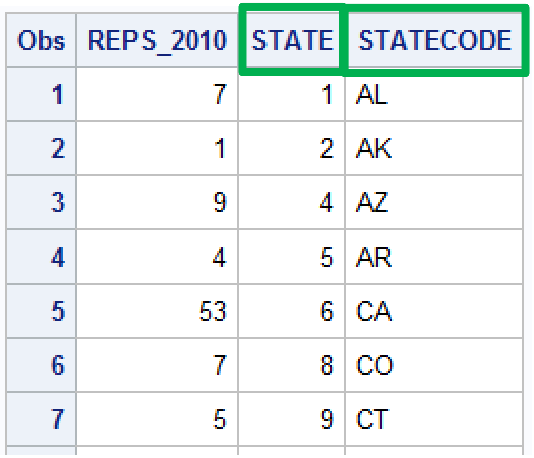
New Datasets

us_pop



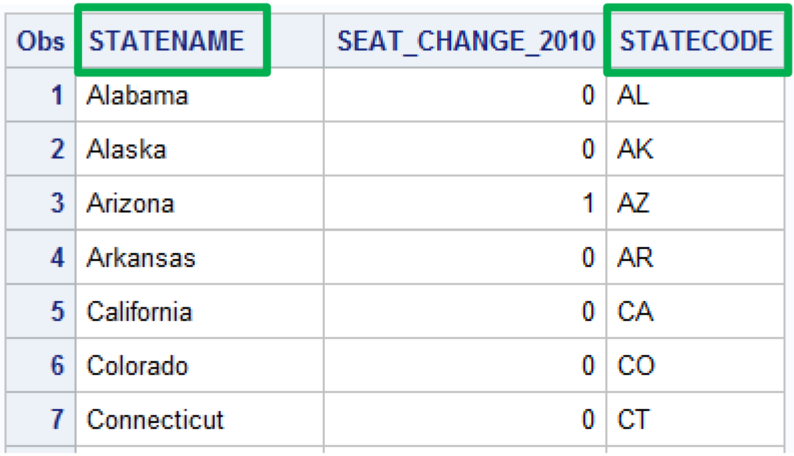
Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

us_rep



Obs	REPS_2010	STATE	STATECODE
1	7	1	AL
2	1	2	AK
3	9	4	AZ
4	4	5	AR
5	53	6	CA
6	7	8	CO
7	5	9	CT

us_seatch



Obs	STATENAME	SEAT_CHANGE_2010	STATECODE
1	Alabama	0	AL
2	Alaska	0	AK
3	Arizona	1	AZ
4	Arkansas	0	AR
5	California	0	CA
6	Colorado	0	CO
7	Connecticut	0	CT

Joins

Basic set up of a join

```
proc sql;
   select   dataset1.*,
            dataset2.*
   from     dataset1,
            dataset2
   where dataset1.common_var = dataset2.common_var;
quit;
```

Common variable structure:

data_source.variable_name

When joining tables, it is important to let SQL know which variables are coming from each dataset, especially when variables in different datasets have the same name.

Joins

Basic set up of a join

```
proc sql;
   select   dataset1.*,
            dataset2.*
   from     dataset1,
            dataset2
   where dataset1.common_var = dataset2.common_var;
quit;
```

Without the where statement, the output table would be a Cartesian product join, where every observation in the first dataset would be joined to the first observation in the second table, and that would be repeated for each subsequent observation in the second table, until you had a table with 52*52 = 2,704 observations!

Joins

Use abbreviations for dataset names

```
proc sql;
   select dataset1.*,
          dataset2.*
   from   dataset1,
          dataset2
   where dataset1.common_var = dataset2.common_var;
quit;
```

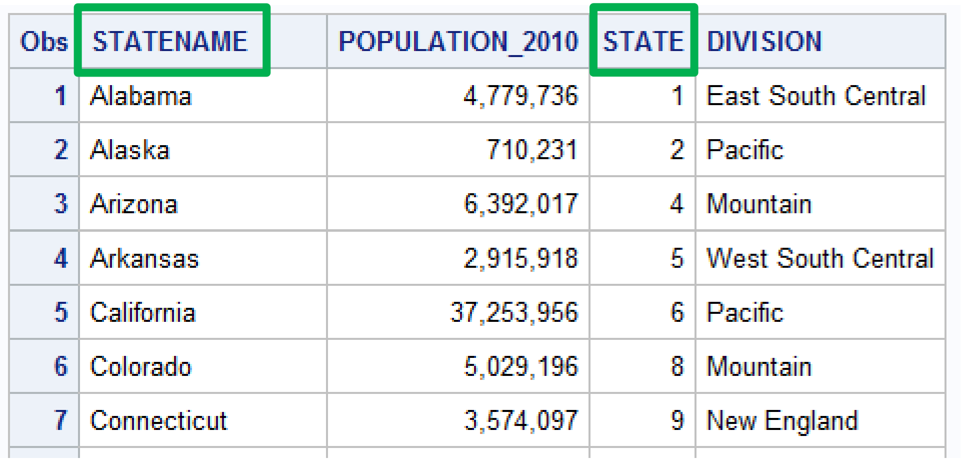
equivalent

```
proc sql;
   select abbr1.*,
          abbr2.*
   from   dataset1 as abbr1,
          dataset2 as abbr2
   where abbr1.common_var = abbr2.common_var;
quit;
```

the ‘as’ is optional in the from clause

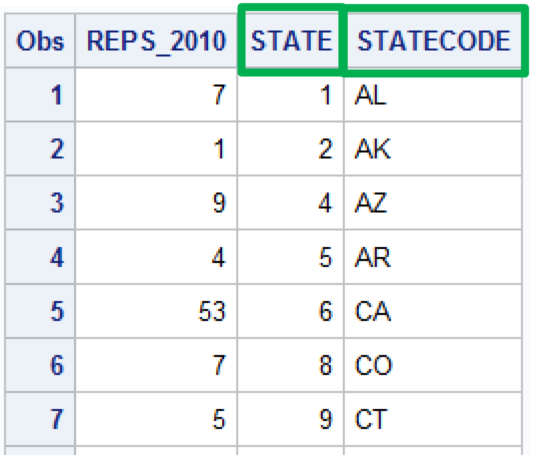
Desired Join

us_pop



Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

us_rep



Obs	REPS_2010	STATE	STATECODE
1	7	1	AL
2	1	2	AK
3	9	4	AZ
4	4	5	AR
5	53	6	CA
6	7	8	CO
7	5	9	CT

Joins

Example of a join – keep all variables

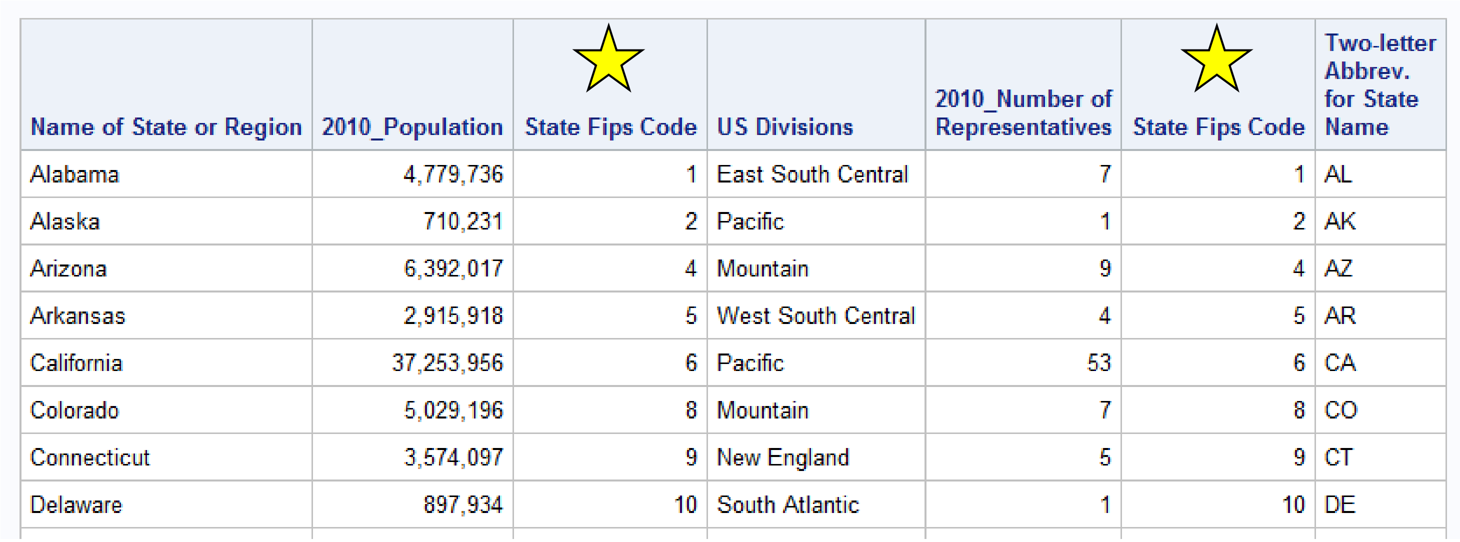
```
proc sql;
   select  pop.*,
           rep.*
   from    us_pop as pop,
           us_rep as rep
   where pop.state = rep.state;
quit;
```

Note that unlike merges in the data step, there’s no need to sort the input datasets prior to a join!

Joins

Example of a join – keep all variables

```
proc sql;
   select pop.*, rep.*
   from us_pop as pop, us_rep as rep
   where pop.state = rep.state;
quit;
```



Name of State or Region	2010_Population	★ State Fips Code	US Divisions	2010_Number of Representatives	★ State Fips Code	Two-letter Abbrev. for State Name
Alabama	4,779,736	1	East South Central	7	1	AL
Alaska	710,231	2	Pacific	1	2	AK
Arizona	6,392,017	4	Mountain	9	4	AZ
Arkansas	2,915,918	5	West South Central	4	5	AR
California	37,253,956	6	Pacific	53	6	CA
Colorado	5,029,196	8	Mountain	7	8	CO
Connecticut	3,574,097	9	New England	5	9	CT
Delaware	897,934	10	South Atlantic	1	10	DE

Joins

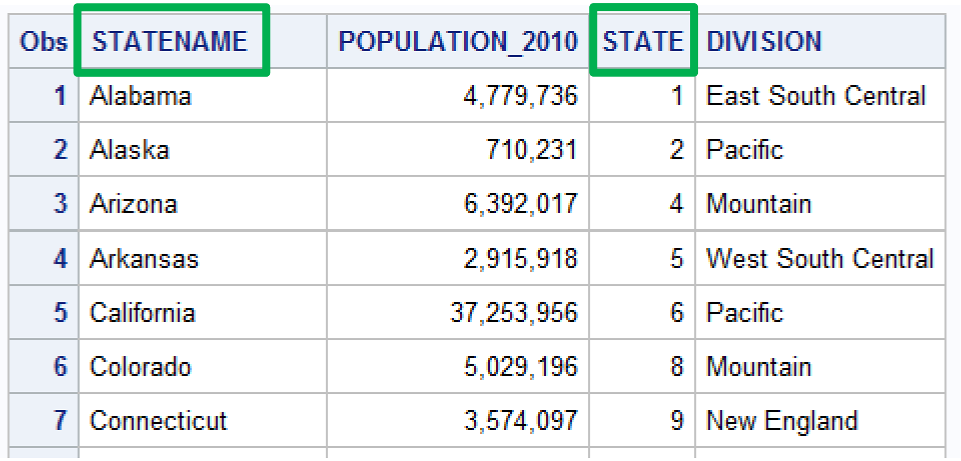
Save joined tables as SAS dataset

```
proc sql;
   create table pop_rep as
   select  pop.*,
           rep.*
   from    us_pop as pop,
           us_rep as rep
   where pop.state = rep.state;

   select *
   from pop_rep;
quit;
```

prints the new dataset

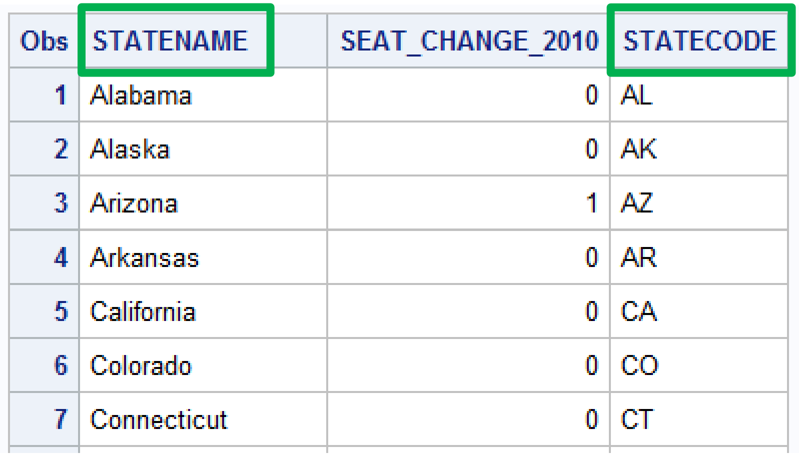
Desired Join

us_pop



Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

us_seatch



Obs	STATENAME	SEAT_CHANGE_2010	STATECODE
1	Alabama	0	AL
2	Alaska	0	AK
3	Arizona	1	AZ
4	Arkansas	0	AR
5	California	0	CA
6	Colorado	0	CO
7	Connecticut	0	CT

But we only want the following variables in the final joined table:

- Statecode
- Population
- Seat Change

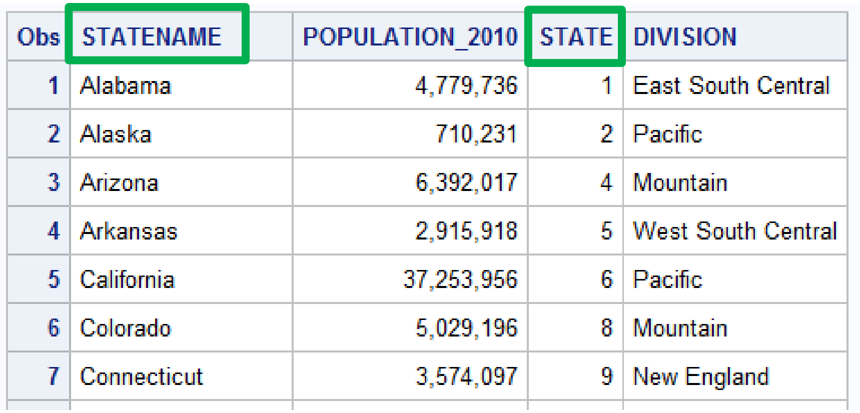
Joins

We only want the following variables in the final joined table:

- Statecode
- Population
- Seat Change

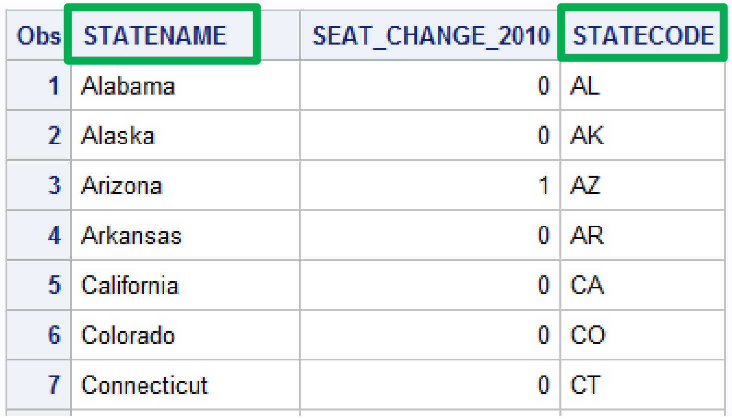
```
proc sql;
   select  statecode,
           pop_2010,
           seat_change_2010
   from    us_pop as pop,
           us_seatch as seat
   where   pop.statename = seat.statename;
quit;
```

us_pop



Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

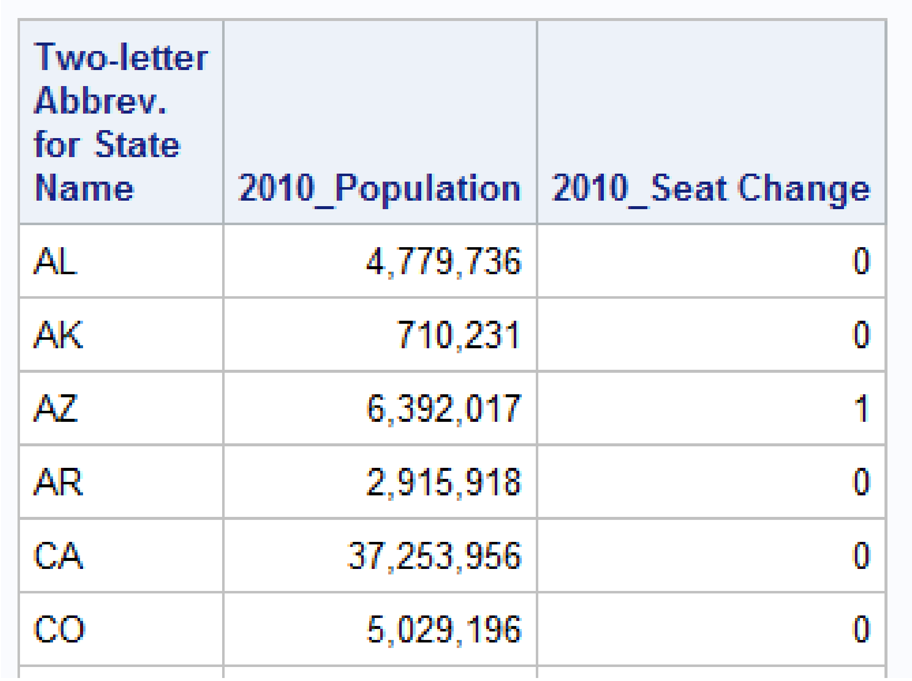
us_seatch



Obs	STATENAME	SEAT_CHANGE_2010	STATECODE
1	Alabama	0	AL
2	Alaska	0	AK
3	Arizona	1	AZ
4	Arkansas	0	AR
5	California	0	CA
6	Colorado	0	CO
7	Connecticut	0	CT

Joins

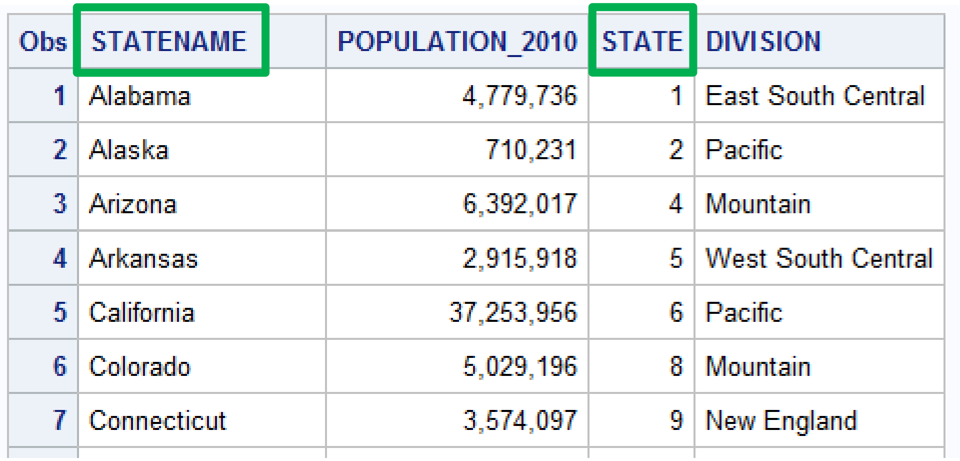
```
proc sql;
   select  statecode,
           pop_2010,
           seat_change_2010
   from    us_pop as pop,
           us_seatch as seat
   where   pop.statename = seat.statename;
quit;
```



Two-letter Abbrev. for State Name	2010_Population	2010_Seat Change
AL	4,779,736	0
AK	710,231	0
AZ	6,392,017	1
AR	2,915,918	0
CA	37,253,956	0
CO	5,029,196	0

Desired Join

us_pop

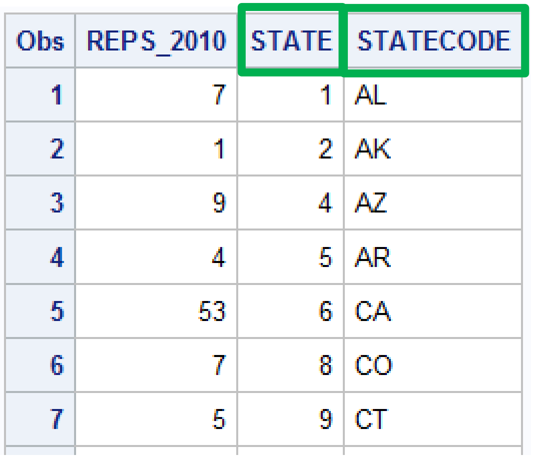


Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

Keep:

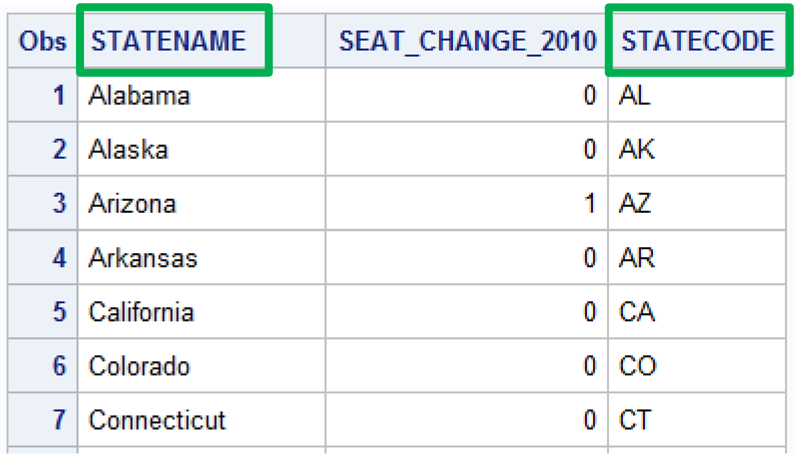
- Statename
- Population
- Representatives
- Seat Change

us_rep



Obs	REPS_2010	STATE	STATECODE
1	7	1	AL
2	1	2	AK
3	9	4	AZ
4	4	5	AR
5	53	6	CA
6	7	8	CO
7	5	9	CT

us_seatch



Obs	STATENAME	SEAT_CHANGE_2010	STATECODE
1	Alabama	0	AL
2	Alaska	0	AK
3	Arizona	1	AZ
4	Arkansas	0	AR
5	California	0	CA
6	Colorado	0	CO
7	Connecticut	0	CT

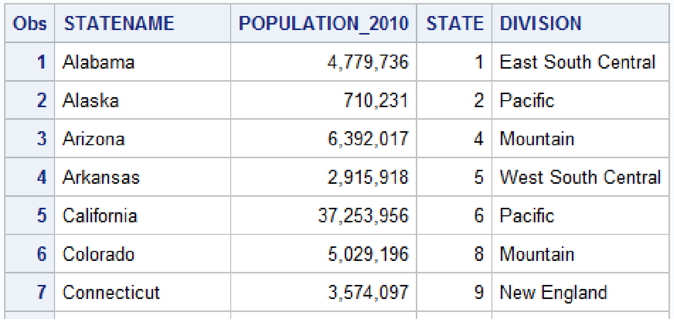
Joins

Keep:

- Statename
- Population
- Representatives
- Seat Change

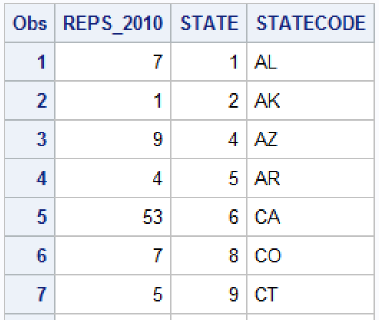
```
proc sql;
   select  pop.statename,
           pop_2010,
           reps_2010,
           seat_change_2010
   from    us_pop as pop,
           us_rep as rep,
           us_seatch as seat
   where   pop.state = rep.state AND
           rep.statecode = seat.statecode;
quit;
```

us_pop



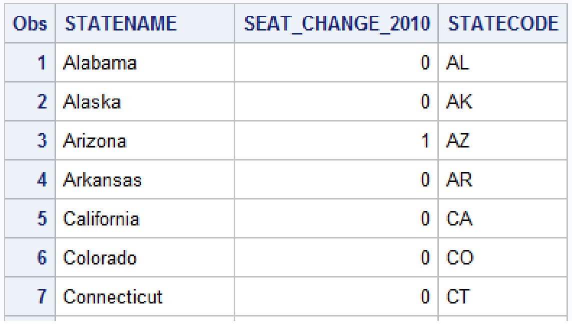
Obs	STATENAME	POPULATION_2010	STATE	DIVISION
1	Alabama	4,779,736	1	East South Central
2	Alaska	710,231	2	Pacific
3	Arizona	6,392,017	4	Mountain
4	Arkansas	2,915,918	5	West South Central
5	California	37,253,956	6	Pacific
6	Colorado	5,029,196	8	Mountain
7	Connecticut	3,574,097	9	New England

us_rep



Obs	REPS_2010	STATE	STATECODE
1	7	1	AL
2	1	2	AK
3	9	4	AZ
4	4	5	AR
5	53	6	CA
6	7	8	CO
7	5	9	CT

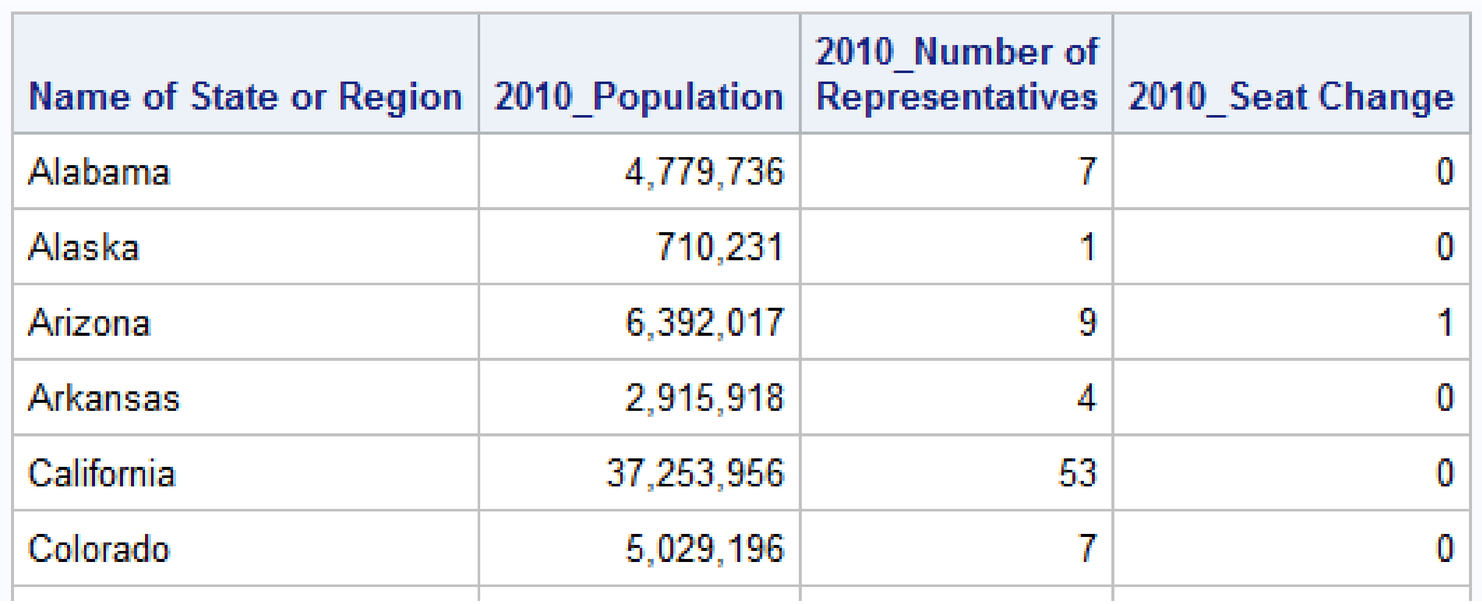
us_seatch



Obs	STATENAME	SEAT_CHANGE_2010	STATECODE
1	Alabama	0	AL
2	Alaska	0	AK
3	Arizona	1	AZ
4	Arkansas	0	AR
5	California	0	CA
6	Colorado	0	CO
7	Connecticut	0	CT

Joins

```
proc sql;
    select  pop.statename,
            pop_2010,
            reps_2010,
            seat_change_2010
    from    us_pop as pop,
            us_rep as rep,
            us_seatch as seat
    where   pop.state = rep.state AND
            rep.statecode = seat.statecode;
quit;
```



Name of State or Region	2010_Population	2010_Number of Representatives	2010_Seat Change
Alabama	4,779,736	7	0
Alaska	710,231	1	0
Arizona	6,392,017	9	1
Arkansas	2,915,918	4	0
California	37,253,956	53	0
Colorado	5,029,196	7	0

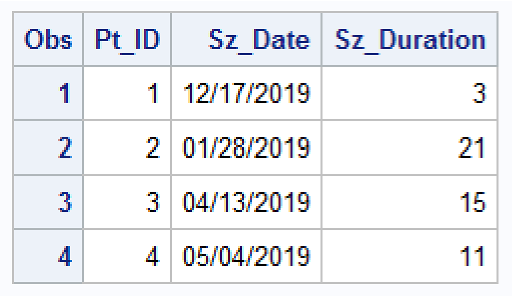
Some Good Uses for SQL

University of Nebraska Medical Center

Joins related to dates

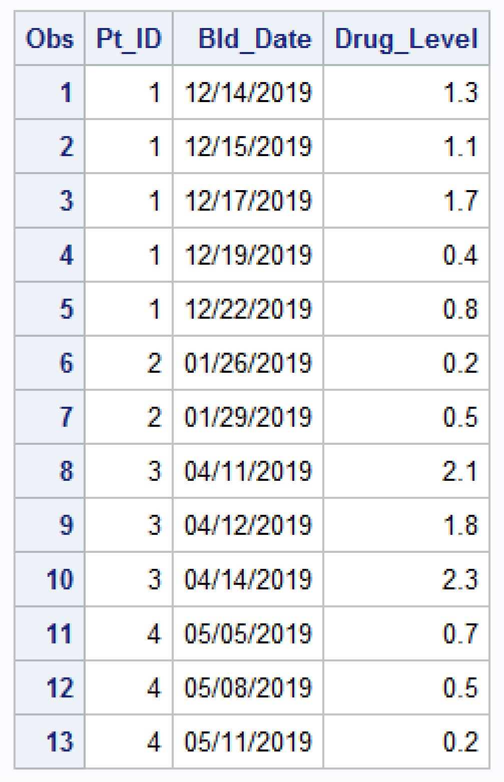
The goal is to match up blood data that was collected *before* the day the seizure occurred, but no more than one week before

seizure



Obs	Pt_ID	Sz_Date	Sz_Duration
1	1	12/17/2019	3
2	2	01/28/2019	21
3	3	04/13/2019	15
4	4	05/04/2019	11

blood

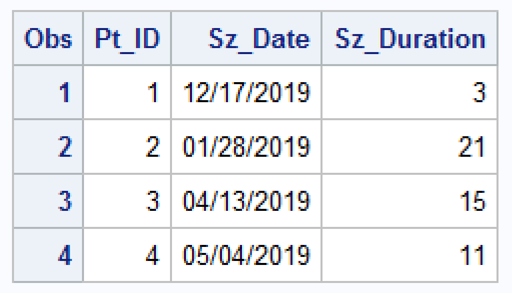


Obs	Pt_ID	Bld_Date	Drug_Level
1	1	12/14/2019	1.3
2	1	12/15/2019	1.1
3	1	12/17/2019	1.7
4	1	12/19/2019	0.4
5	1	12/22/2019	0.8
6	2	01/26/2019	0.2
7	2	01/29/2019	0.5
8	3	04/11/2019	2.1
9	3	04/12/2019	1.8
10	3	04/14/2019	2.3
11	4	05/05/2019	0.7
12	4	05/08/2019	0.5
13	4	05/11/2019	0.2

Joins related to dates

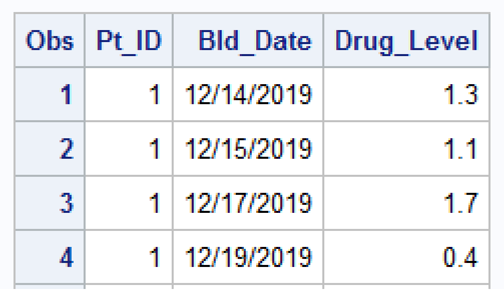
```
proc sql;
   select  s.*,
           b.*,
           (bld_date - sz_date) as days_diff
   from    seizure as s,
           blood as b
   where   s.pt_ID = b.pt_ID AND
           -7 le CALCULATED days_diff le -1;
quit;
```

seizure



Obs	Pt_ID	Sz_Date	Sz_Duration
1	1	12/17/2019	3
2	2	01/28/2019	21
3	3	04/13/2019	15
4	4	05/04/2019	11

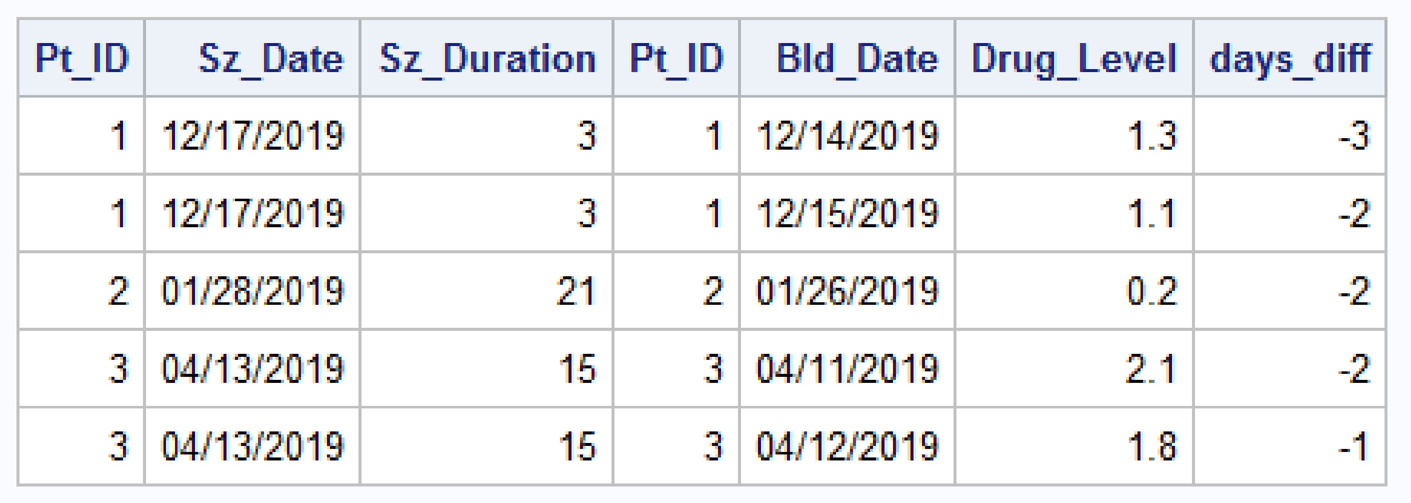
blood



Obs	Pt_ID	Bld_Date	Drug_Level
1	1	12/14/2019	1.3
2	1	12/15/2019	1.1
3	1	12/17/2019	1.7
4	1	12/19/2019	0.4

Joins related to dates

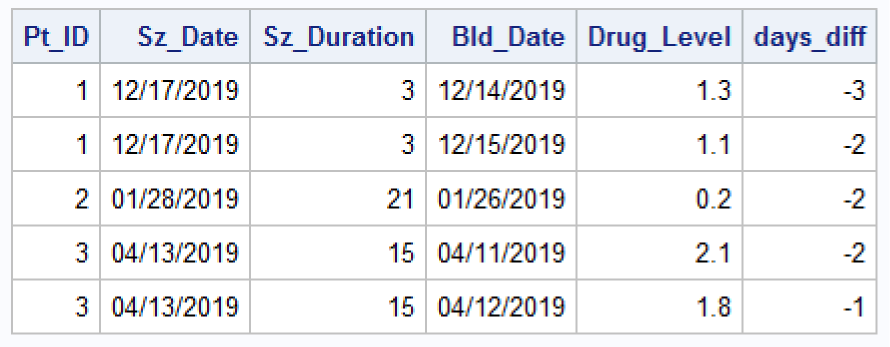
```
proc sql;
    select    s.*,
              b.*,
              (bld_date - sz_date) as days_diff
    from      seizure as s,
              blood as b
    where  s.pt_ID = b.pt_ID AND
              -7 le CALCULATED days_diff le -1;
quit;
```



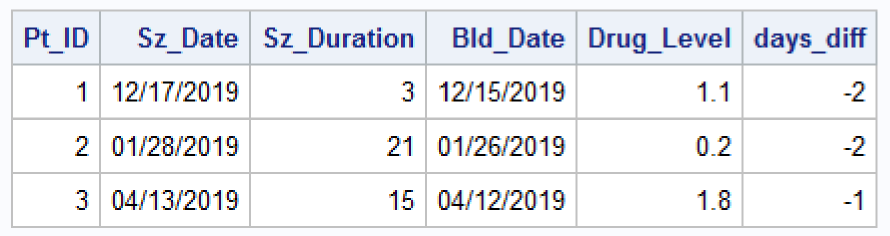
Pt_ID	Sz_Date	Sz_Duration	Pt_ID	Bld_Date	Drug_Level	days_diff
1	12/17/2019	3	1	12/14/2019	1.3	-3
1	12/17/2019	3	1	12/15/2019	1.1	-2
2	01/28/2019	21	2	01/26/2019	0.2	-2
3	04/13/2019	15	3	04/11/2019	2.1	-2
3	04/13/2019	15	3	04/12/2019	1.8	-1

Joins related to dates

```
proc sql;
 create table sz_bld as
 select
      s.*,
      b.*,
      (bld_date - sz_date) as days_diff
 from seizure as s,
      blood as b
 where s.pt_ID = b.pt_ID AND
       -7 le CALCULATED days_diff le -1
 order by pt_id, days_diff;

 select *
 from sz_bld;
quit;

*If you only want one blood draw per
patient, and prefer the draw that was
closest to the seizure date;
data sz_bld_single;
     set sz_bld;
     by pt_id days_diff;

     if last.pt_id then output;
run;
```



Pt_ID	Sz_Date	Sz_Duration	Bld_Date	Drug_Level	days_diff
1	12/17/2019	3	12/14/2019	1.3	-3
1	12/17/2019	3	12/15/2019	1.1	-2
2	01/28/2019	21	01/26/2019	0.2	-2
3	04/13/2019	15	04/11/2019	2.1	-2
3	04/13/2019	15	04/12/2019	1.8	-1



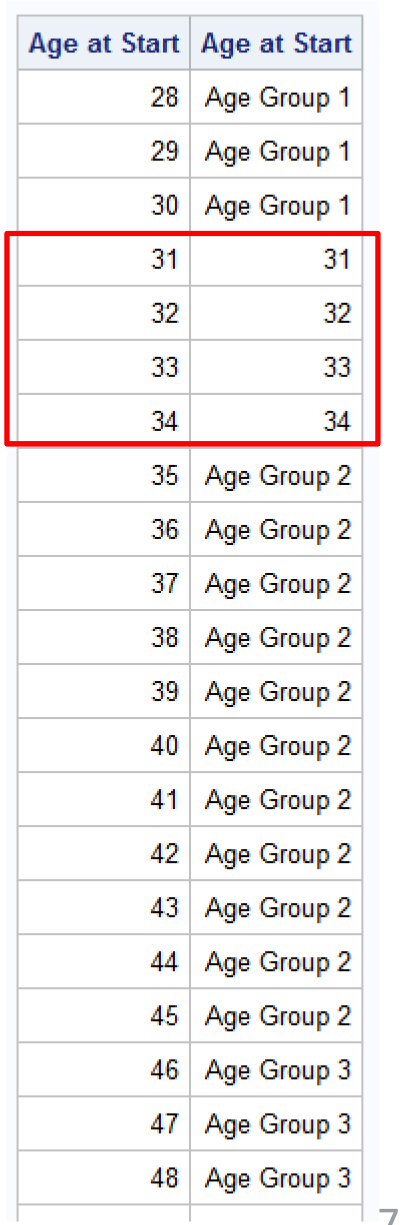
Pt_ID	Sz_Date	Sz_Duration	Bld_Date	Drug_Level	days_diff
1	12/17/2019	3	12/15/2019	1.1	-2
2	01/28/2019	21	01/26/2019	0.2	-2
3	04/13/2019	15	04/12/2019	1.8	-1

Confirming User-Defined Formats

```
proc format;
   value age_group low - 30 = "Age Group 1"
                   35 - 45  = "Age Group 2"
                   46 - 55  = "Age Group 3"
                   56 - 65  = "Age Group 4"
                   66 - high  = "Age Group 5";
run;

proc sql;
   select distinct ageatstart,
                   ageatstart format=age_group.
   from sashelp.heart;
quit;
```

Confirming User-Defined Formats

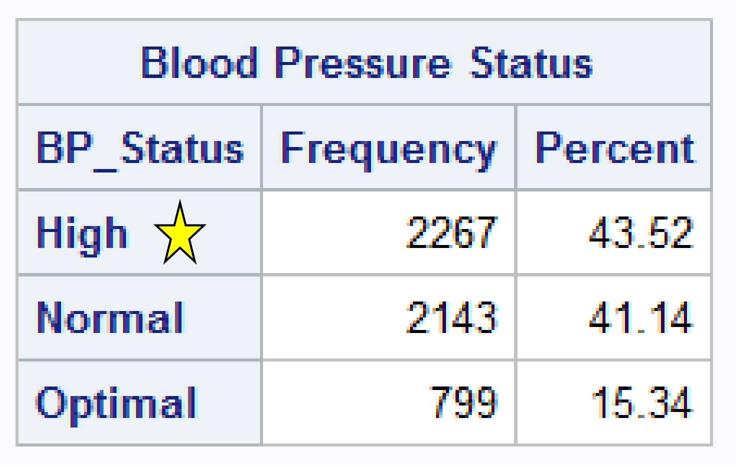
```
proc format;
 value age_group
  low - 30 = "Age Group 1"
  35 - 45  = "Age Group 2“
  46 - 55  = "Age Group 3“
  56 - 65  = "Age Group 4"
  66 - high  = "Age Group 5";
run;

proc sql;
select distinct ageatstart,
                ageatstart format=age_group.
   from sashelp.heart;
quit;
```



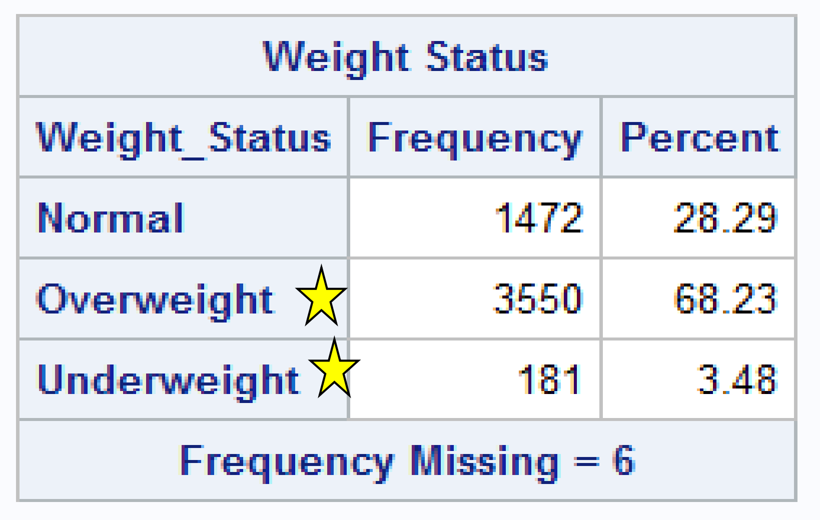
Age at Start	Age at Start
28	Age Group 1
29	Age Group 1
30	Age Group 1
31	31
32	32
33	33
34	34
35	Age Group 2
36	Age Group 2
37	Age Group 2
38	Age Group 2
39	Age Group 2
40	Age Group 2
41	Age Group 2
42	Age Group 2
43	Age Group 2
44	Age Group 2
45	Age Group 2
46	Age Group 3
47	Age Group 3
48	Age Group 3

Confirming User-Defined Variables

The goal is to create a new variable ‘risk’, which takes the value ‘At risk’ if any of the following variables have values with a star:



Blood Pressure Status		
BP_Status	Frequency	Percent
High ☆	2267	43.52
Normal	2143	41.14
Optimal	799	15.34



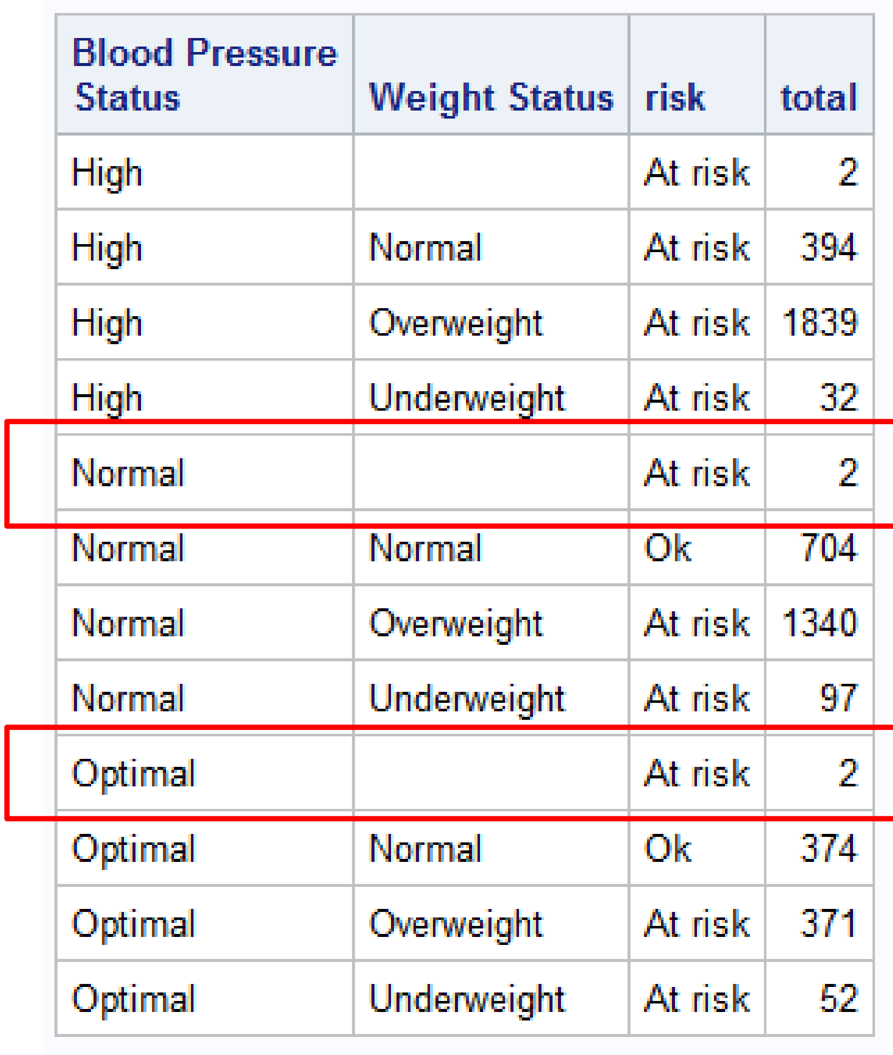
Weight Status		
Weight_Status	Frequency	Percent
Normal	1472	28.29
Overweight ☆	3550	68.23
Underweight ☆	181	3.48
Frequency Missing = 6		

Confirming User-Defined Variables

```
data heart;
    set sashelp.heart;

    length risk $8.;

    *Create overall risk variable;
    if  bp_status = "High" OR
        weight_status ne "Normal" then risk = "At risk";
    else risk = "Ok";

run;



proc sql;
    select distinct bp_status,
                    weight_status,
                    risk,
                    count(*) as total
    from heart
    group by bp_status, weight_status, risk;
quit;
```

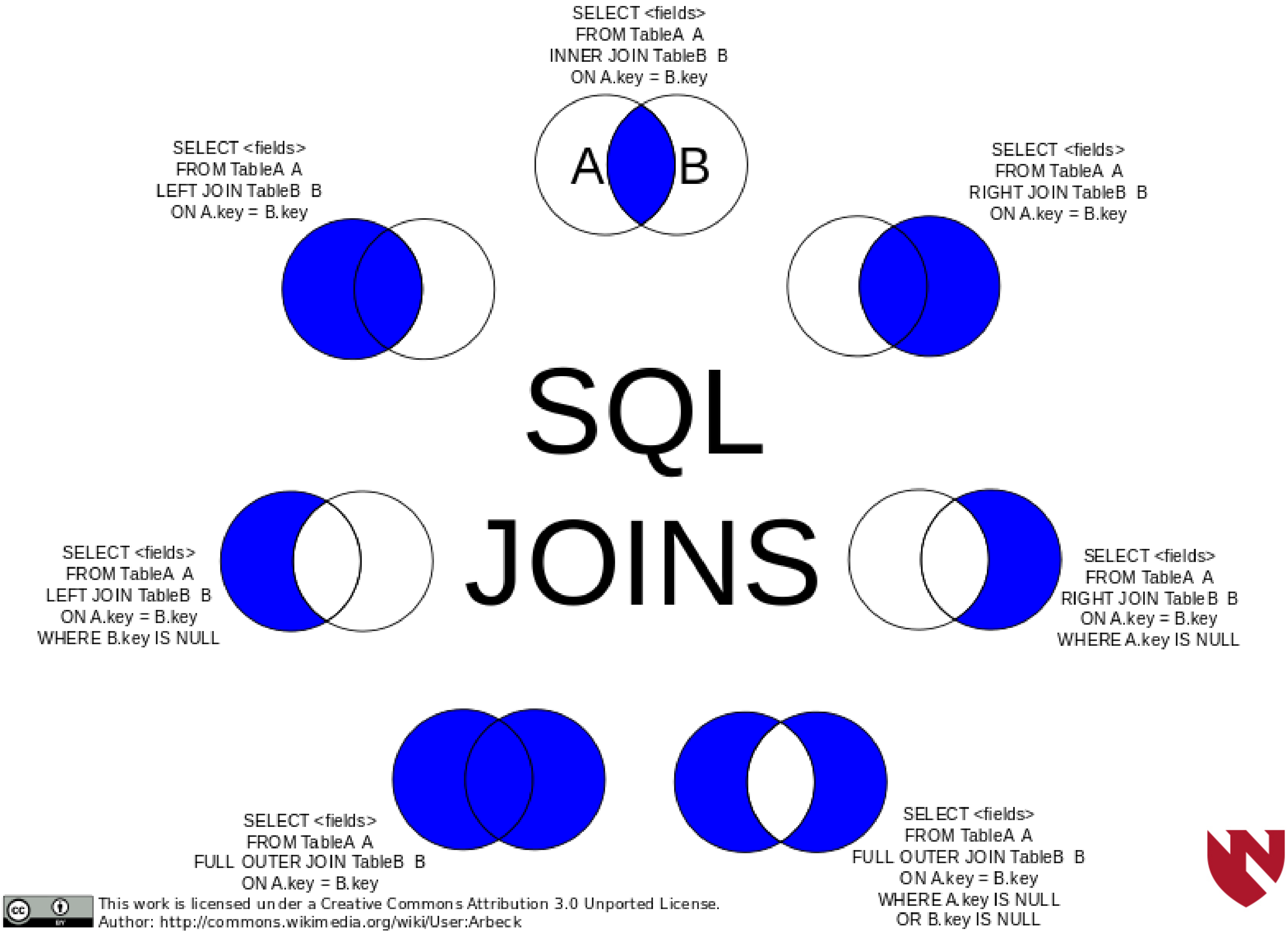
Confirming User-Defined Variables

```
proc sql;
    select distinct
     bp_status,
     weight_status,
     risk,
     count(*) as total
    from heart
    group by
     bp_status,
     weight_status,
     risk;
quit;
```



Blood Pressure Status	Weight Status	risk	total
High		At risk	2
High	Normal	At risk	394
High	Overweight	At risk	1839
High	Underweight	At risk	32
Normal		At risk	2
Normal	Normal	Ok	704
Normal	Overweight	At risk	1340
Normal	Underweight	At risk	97
Optimal		At risk	2
Optimal	Normal	Ok	374
Optimal	Overweight	At risk	371
Optimal	Underweight	At risk	52

Other Helpful Joins



```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

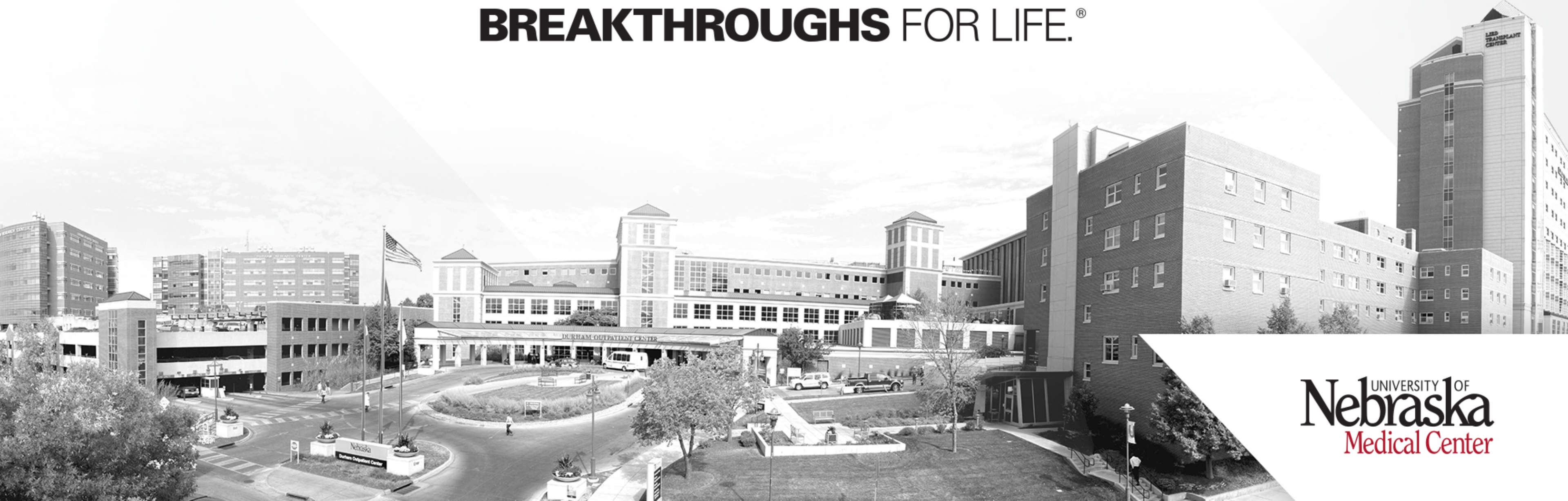
Questions?

kksamson@unmc.edu

University of Nebraska
Medical Center

University of Nebraska℠
Medical Center

BREAKTHROUGHS FOR LIFE.®



UNIVERSITY OF
Nebraska
Medical Center