Advanced SAS Programmer

SAP104

104 Subquery (in WHERE, HAVING, FROM, and SELECT Clauses)

```
/* Defines the path to your data and assigns the libref. */
%let path=~/ESQ1M6;

libname sq "&path/data";
************************************************************;
*  Activity 3.01                                           *;
*  1) Examine and run the two queries to explore the       *;
*     sq.smallcustomer and sq.smalltransaction tables.     *;
*     Confirm that the sq.smallcustomer contains 8 rows    *;
*     and the sq.smalltransaction contains 12 rows.        *;
*  2) In the next section, list the sq.smallcustomer and   *;
*     sq.smalltransaction table in the FROM clause and     *;
*     separate the tables by a comma. Run the query and    *;
*     view the log. What note do you see?                  *;
*  3) View the results. Name two issues with the report.   *;
************************************************************;

************************************************************;
*  Subquery That Returns a Single Value                    *;
************************************************************;
*  Syntax                                                  *;
*                                                          *;
*  PROC SQL;                                               *;
*  SELECT col-name, col-name                               *;
*     FROM input-table                                     *;
```

```
*    WHERE column operator (SELECT col-name           *;
*                          FROM input-table)       *;
*    ORDER BY col-name <DESC>;                     *;
*  QUIT;                                  *;
************************************************************;

************************************************************;
*  Demo                                   *;
*  1) Open the s104d01.sas program in the demos folder    *;
*    and find the Demo section. Run the query to explore *;
*    the sq.statepopulation table.                  *;
*  2) Run the query in the Future Subquery section to     *;
*    find the average of PopEstimate1 in the           *;
*    sq.statepopulation table. View the results.        *;
*  3) Complete the outer query to find all states that    *;
*    have a higher PopEstimate1 value than the result of *;
*    the previous query's value of 6278420. Sort the     *;
*    results by descending PopEstimate1.               *;
*  4) Remove the value 6278420 in the WHERE clause,      *;
*    replace it with the query that found the average of *;
*    PopEstimate1, and enclose the inner query in       *;
*    parentheses.                           *;
*    Note: Remember to remove the semicolon from the     *;
*         subquery when copying and pasting.           *;
*  5) Replace the subquery with a new query. Find the     *;
*    average of Population_2010 from the sashelp.us_data *;
*    table.                                 *;
*    Note: The table in a subquery can be a different    *;
*         table than the outer query.                *;
```

```
************************************************************;

**********************************;
*Explore the STATEPOPULATION table*;
**********************************;
proc sql;
select Name, PopEstimate1
        from sq.statepopulation;
quit;
```

Name	PopEstimate1
AL	4864745
AK	741504
AZ	6945452
AR	2990410
CA	39209127
CO	5540921

```
**********************************;
*Future Subquery               *;
**********************************;
proc sql;
select avg(PopEstimate1)
   from sq.statepopulation;
quit;
```

6278420

```
**********************************;
*Outer Query                   *;
**********************************;
proc sql;
select Name, PopEstimate1
   from sq.statepopulation
```

```
   where PopEstimate1 > 6278420
   order by PopEstimate1 desc;
quit;


proc sql;
select Name, PopEstimate1
   from sq.statepopulation
   where PopEstimate1 > (select avg(PopEstimate1)
                                          from sq.statepopulation)
   order by PopEstimate1 desc;
quit;
```

Name	PopEstimate1
CA	39209127
TX	27937492
FL	20629982
NY	19641589
IL	12826895
PA	12783538
OH	11635003
GA	10304763
NC	10156679
MI	9951890
NJ	8874516
VA	8410946
WA	7294680
AZ	6945452
MA	6826022
TN	6645011
IN	6633344

```
proc sql;
select Name, PopEstimate1
   from sq.statepopulation
   where PopEstimate1 > (select avg(Population_2010)
                                          from sashelp.us_data)
```

```
   order by PopEstimate1 desc;
quit;
```

Name	PopEstimate1
CA	39209127
TX	27937492
FL	20629982
NY	19641589
IL	12826895
PA	12783538
OH	11635003
GA	10304763
NC	10156679
MI	9951890
NJ	8874516
VA	8410946
WA	7294680
AZ	6945452
MA	6826022
TN	6645011
IN	6633344
MO	6087203

```
************************************************************;
*  Activity 4.01                                 *;
*  1) Examine and run the first query. Confirm that the   *;
*     results contain one row and two columns.           *;
*  2) Add the first query as a subquery in the second     *;
*     query to find all states with PopEstimate1 higher   *;
*     than the average estimated state population.       *;
*  3) Run the query. What is the syntax error in the log? *;
************************************************************;


proc sql;
select avg(PopEstimate1), "Average Estimated Population"
            from sq.statepopulation;
```

```
quit;
```

6278420	Average Estimated Population

```
proc sql;
select Name, PopEstimate1
        from sq.statepopulation
        where PopEstimate1 > (select avg(PopEstimate1), "Average Estimated Population"
                                                        from sq.statepopulation);
quit;
ERROR: A subquery cannot select more than one column.
ERROR: A Composite expression (usually a subquery) is used incorrectly in an expression.

************************************************************;
*  Activity 4.02                          *;
*  1) Examine and run the first query. View the results.  *;
*  2) Modify the second query. Copy the value returned by *;
*     the first query into the subquery against the       *;
*     HAVING clause to return divisions with an average   *;
*     PopEstimate1 value greater than the total average   *;
*     of PopEstimate1.                    *;
*  3) Remove the static value and add the subquery in the *;
*     HAVING clause.                      *;
*  4) How many divisions have a higher average          *;
*     PopEstimate1 than the average PopEstimate1 of all   *;
*     the states?                         *;
************************************************************;


proc sql;
select avg(PopEstimate1)
        from sq.statepopulation;
```

quit;

6278420

```
proc sql;
select Division, avg(PopEstimate1) as avgDivisionPop
        from sq.statepopulation
        group by Division
        having avgDivisionPop > 6278420;
quit;
```

Division	avgDivisionPop
2	13766548
3	9364018
5	7103557
7	9883222
9	10552964

```
proc sql;
select Division, avg(PopEstimate1) as avgDivisionPop
        from sq.statepopulation
        group by Division
        having avgDivisionPop > (select avg(PopEstimate1)
                                 from sq.statepopulation);
quit;
```

Division	avgDivisionPop
2	13766548
3	9364018
5	7103557
7	9883222
9	10552964

```
************************************************************;
*  Subquery That Returns Multiple Values                   *;
************************************************************;
```

```
*  Syntax                                  *;
*                                        *;
*  PROC SQL;                               *;
*  SELECT col-name, col-name               *;
*    FROM input-table                      *;
*    WHERE column operator (SELECT col-name  *;
*                          FROM input-table)  *;
*    ORDER BY col-name <DESC>;               *;
*  QUIT;                                   *;
************************************************************;

************************************************************;
*  Demo                                    *;
*  1) Open the s104d02.sas program in the demos folder    *;
*    and find the Demo section.            *;
*  2) Examine the Future Subquery section and run the     *;
*    query to find the states that reside in Division 3. *;
*    View the results.                     *;
*  3) Using the results from the Future Subquery section, *;
*    modify the outer query by entering the State       *;
*    abbreviations ("IL","IN","MI","OH","WI") in the    *;
*    WHERE clause. Run the query and view the results.   *;
*  4) Replace the values in parentheses with the query    *;
*    from step 2 by copying and pasting the query inside *;
*    the parentheses. The query is now the subquery. Be  *;
*    sure to remove the semicolon from inside the       *;
*    parentheses. Highlight and run the query. Discuss   *;
*    the error in the log.                 *;
*    Note: This program will return a syntax error.      *;
```

```
*  5) Remove the Division column from the subquery.      *;
*     Highlight and run the query. Discuss the results.   *;
*  6) Change the Division value from 3 to 6 in the       *;
*     subquery. Replace the 3 at the end of the new table *;
*     name in the CREATE TABLE statement to a 6. Highlight*;
*     and run the query. Discuss the results.            *;
************************************************************;


***********************************;
* Future Subquery              *;
***********************************;
proc sql;
select Division, Name
   from sq.statepopulation
   where Division = '3';
quit;
```

Division	Name
3	IL
3	IN
3	MI
3	OH
3	WI

```
***********************************;
*Outer Query                  *;
***********************************;
proc sql;
create table division3 as
select *
          from sq.customer
          where State in ('IL','IN','MI','OH','WI');
```

quit;

Total rows: 16022 Total columns: 22 Rows 1-100

	FirstName	MiddleName	LastName	Gender	DOB	Employed	Race	Married
1	Rodney	Matthew	Joyner	M	2202	Y	W	M
2	Brian	Dallas	Harper	M	-4584	N	W	M
3	Becky	Danna	Cheers	F	-5365	N	W	M
4	Alberto	Daryl	Texter	M	15193	N	W	S

```
proc sql;
create table division3 as
select *
        from sq.customer
        where State in (select Name
                                from sq.statepopulation
                                where Division = '3');
quit;
```

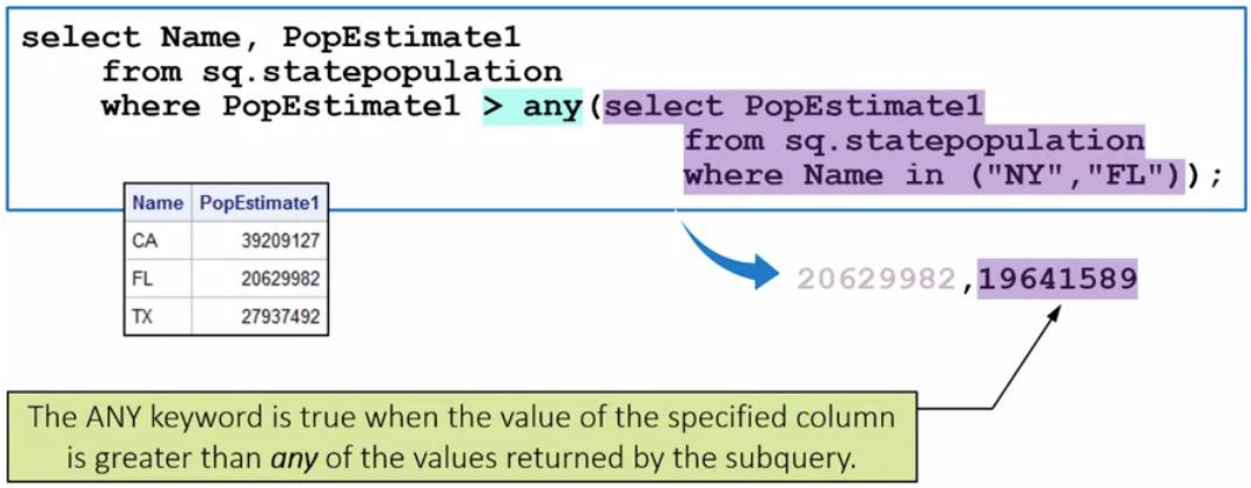
Total rows: 16022 Total columns: 22 Rows 1-1

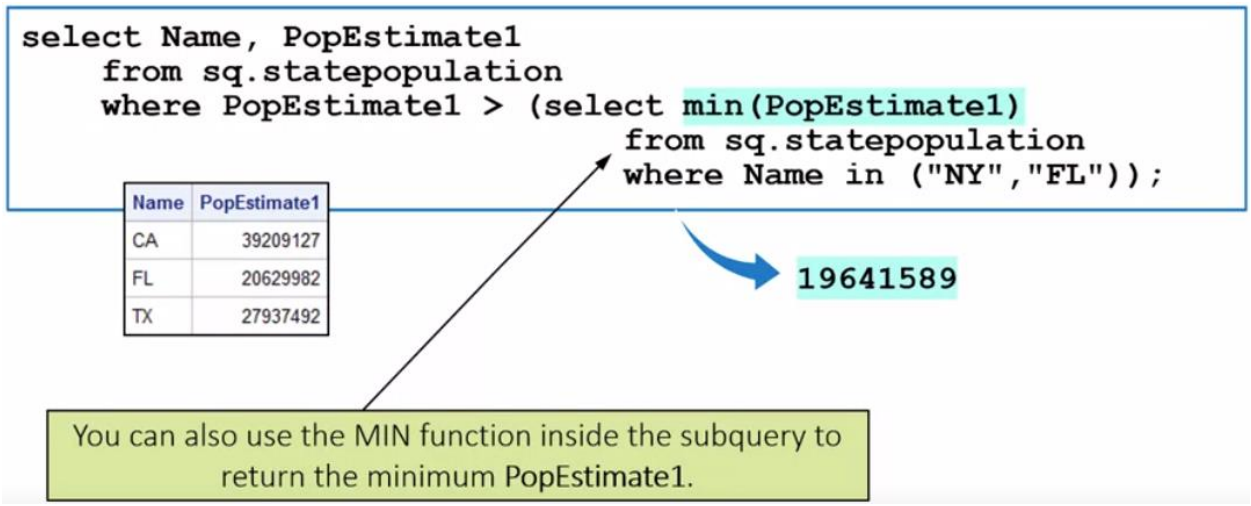
	FirstName	MiddleName	LastName	Gender	DOB	Employed	Race	Married
1	Rodney	Matthew	Joyner	M	2202	Y	W	M
2	Brian	Dallas	Harper	M	-4584	N	W	M
3	Becky	Danna	Cheers	F	-5365	N	W	M
4	Alberto	Daryl	Texter	M	15193	N	W	S

```
proc sql;
create table division6 as
select *
        from sq.customer
        where State in (select Name
                                from sq.statepopulation
                                where Division = '6');
quit;
```

Total rows: 4994 Total columns: 22 Rows 1-100

	FirstName	MiddleName	LastName	Gender	DOB	Employed	Race	Married
1	John	Jerry	Vedia	M	10789	N	B	M
2	Eric	Larry	Klarman	M	-1683	Y	W	M
3	Ann	Jessica	Hopkins	F	13852	Y	W	M
4	Ron	Brian	Milkent	M	5691	Y	W	M
5	Nelida	Christine	Hopson	F	2301	Y	W	S





**********;

* Activity 4.03 *;

* 1) Complete the query using the ANY keyword or MAX *;

* statistic. *;

* 2) Run the query. How many states have estimated *;

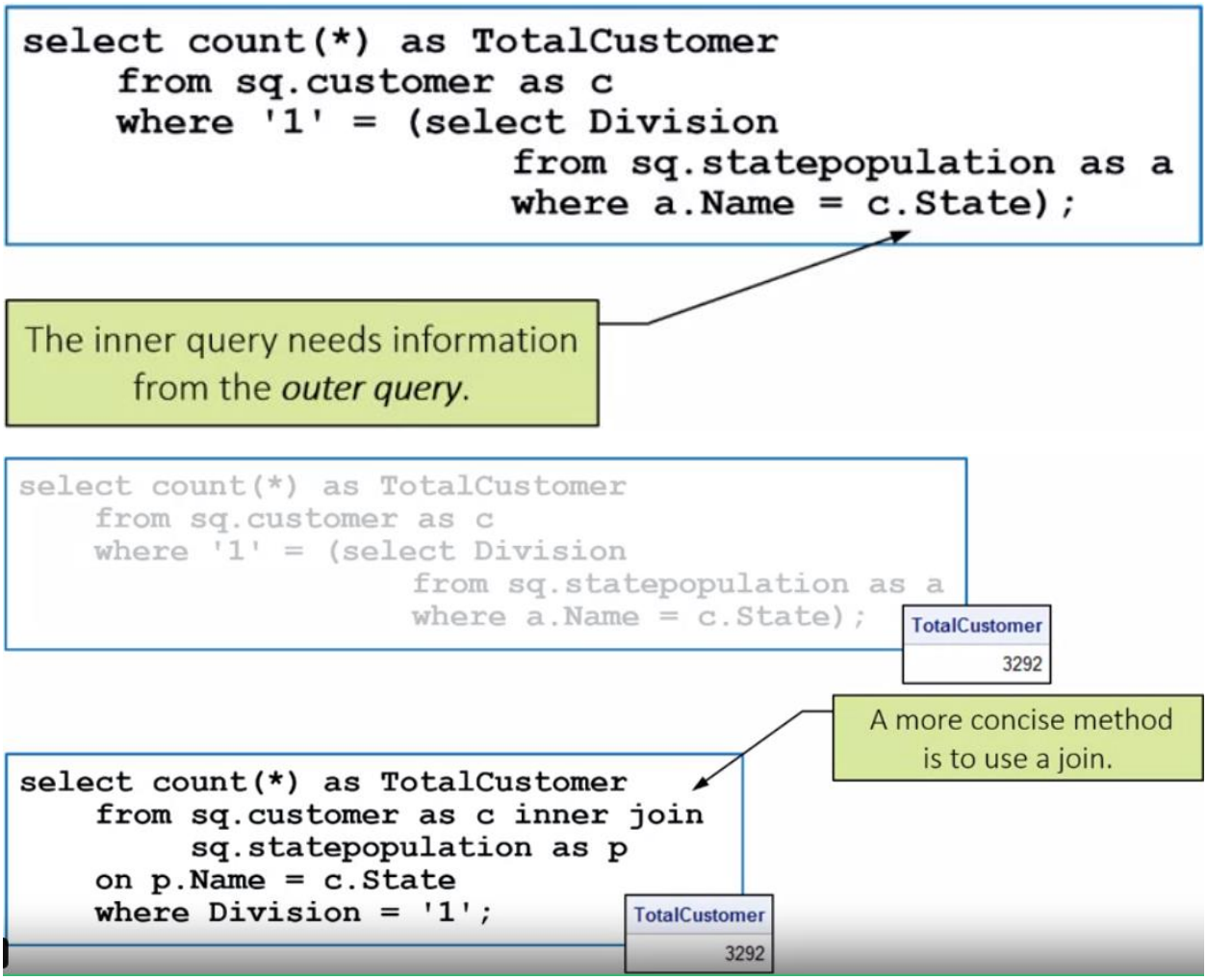
* populations lower than New York or Florida? *;

**********;

```
proc sql number;
select Name, PopEstimate1
   from sq.statepopulation
/*Complete*/
   where PopEstimate1 < ANY (select PopEstimate1
                        from sq.statepopulation
                        where Name in ("NY","FL"));
quit;
```

```
proc sql number;
select Name, PopEstimate1
   from sq.statepopulation
/*Complete*/
   where PopEstimate1 < (select max(PopEstimate1)
                        from sq.statepopulation
                        where Name in ("NY","FL"));
quit;
```

Row	Name	PopEstimate1
1	AL	4864745
2	AK	741504
3	AZ	6945452
4	AR	2990410
5	CO	5540921
6	CT	3578674
7	DE	949216
8	DC	686575
9	GA	10304763
10	HI	1428105



/*Practice Level 1: Using a Subquery That Returns a Single Value

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

The sq.statepopulation table contains estimated population statistics for every state in the US and its territories.

You want to determine which states have an estimated three-year population growth greater than the average for all states.

Using the sq.statepopulation table, write a query that displays the average three-year population growth for all states.

Use the nPopChg3 column to calculate the average from the sq.statepopulation table.

Run the program and view the results.

Use the query you just wrote to display the states that have a projected three-year population growth greater than

the overall average. Use the following requirements:

Include Name and nPopChg3 in the results.

Label nPopChg3 as Estimated Growth and format the values with commas.

Use the query from step 1 to subset the table.

Order the results by descending nPopChg3.

Add an appropriate title.

Run the query and view the results.

Which state has the greatest estimated growth in your results?

*/

```
proc sql;
select avg(nPopChg3)
        from sq.statepopulation
quit;
```

36355.1

```
title "States that have an estimated three-year population growth greater than the average for all states";
proc sql;
select Name, nPopChg3 label='Estimated Growth' format=comma16.
        from sq.statepopulation
        where nPopChg3 > (select avg(nPopChg3)
                        from sq.statepopulation)
        order by nPopChg3 desc;
quit;
title;
```

States that have an estimated three-year population growth greater than the average for all states

Name	Estimated Growth
TX	379,128
FL	322,513
CA	157,696
AZ	122,770
NC	112,820
WA	110,159
GA	106,420
CO	79,662
SC	62,908
NV	61,987
TN	61,216
UT	57,987
VA	52,478
OR	44,121
MN	43,024
MA	38,903

/*Practice Level 2: Using a Subquery with Multiple Functions

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

Use the sq.customer table to determine which Texas customers have higher than average credit scores.

Using the sq.customer table, write a query to define the high-credit threshold as 2 standard deviations greater than

the mean of CreditScore for TX customers.

Use the following expression to create the column HighScore. This will be your future subquery result:

sum(avg(CreditScore),(2*std(CreditScore)))

Run the query and view the results.

Write a second query to create a report showing Texas customers whose CreditScore value is greater than

the results from the previous query. Use the query you just wrote as a subquery. Use the following requirements:

Select CustomerID, FirstName, LastName, and CreditScore.

Subset the table for customers who are from the state of Texas and who have a greater CreditScore value than

the value from the query in step 1.

Order the results by descending CreditScore.

Add an appropriate title.

Run the query and view the results.

In your report, what is the greatest value for CreditScore? Note: Type a numeric value.

*/

```
proc sql;
select
        sum(avg(CreditScore),(2*std(CreditScore))) as HighScore
        from sq.customer
        where State='TX';
quit;
```

HighScore
765.0846

```
title "Texas customers that have higher than average credit scores";
proc sql;
select CustomerID, FirstName, LastName, CreditScore
        from sq.customer
        where State='TX' and
                CreditScore > (select sum(avg(CreditScore),(2*std(CreditScore))) as HighScore
                                                        from sq.customer)
        order by CreditScore desc;
quit;
title;
```

Texas customers that have higher than average credit scores

Customer ID	First Name	Last Name	CreditScore
1915019345	Christopher	Miras	848
1921584392	Christopher	Kiddy	845
1965260689	Morgan	Tamanaha	843
1940349233	Ruth	Poloskey	835
1988972911	Bea	Holzwarth	832
1955883210	Victor	Galway	831
1966934182	Joseph	Malacara	827

```
************************************************************;
* Using an In-Line View                                    *;
************************************************************;
* Syntax                                                   *;
*                                                          *;
* PROC SQL;                                                *;
*   SELECT col-name, col-name                              *;
*   FROM (SELECT column,...                                *;
*         FROM input-table...)                             *;
*   WHERE expression                                       *;
*   ORDER BY col-name <DESC>;                              *;
* QUIT;                                                    *;
************************************************************;


************************************************************;
* Demo                                                     *;
* 1) Open the s104d03.sas program in the demos folder      *;
*   and find the Demo section. Run the first query to      *;
*   explore the statepopulation and customer tables.       *;
* 2) Move to the next section, Temporary Table Solution.   *;
*   Discuss both queries in the section.                   *;
*   a) Run the first query to create the totalcustomer     *;
```

```
*       temporary table. View the results.              *;
*     b) Run the second query to join the totalcustomer   *;
*       and sq.statepopulation tables and calculate the  *;
*       new column PctCustomer that calculates the       *;
*       percentage of customers in each state based on   *;
*       the current year's estimated population. View    *;
*       the results.                                     *;
*  3) Copy only the query that is used to create the      *;
*     totalcustomer table. Move to the Using an In-Line   *;
*     View section of the program. Paste the query (not   *;
*     including the CREATE TABLE statement) in the FROM   *;
*     clause to create an in-line view. Be sure to remove *;
*     the semicolon. Highlight and run the query. View    *;
*     the syntax error in the log.                        *;
*  4) Remove the ORDER BY clause in the subquery. Run the *;
*     query and view the results.                         *;
************************************************************;

*************************************************;
*Explore the CUSTOMER and STATEPOPULATION table *;
*************************************************;
proc sql inobs=10;
select FirstName, MiddleName, LastName, State
   from sq.customer;
select Name, EstimateBase
   from sq.statepopulation;
quit;
```

First Name	Middle Name	Last Name	State
Rodney	Matthew	Joyner	WI
Jeanne	Carol	Ballenger	WA
Brian	Dallas	Harper	WI
Thomas	Eric	Henderson	WA
Becky	Danna	Cheers	WI
Alberto	Daryl	Texter	WI
Peter	Douglas	Schmand	WA
Danielle	Julie	Bell	WI
Robert	Javier	Brousseau	WI
Sharon	Julie	Howell	WA

Name	EstimateBase
AL	4780138
AK	710249
AZ	6392288
AR	2916028
CA	37254523
CO	5029316
CT	3574147
DE	897934
DC	601766
FL	18804580

```
**********************************;
* Temporary Table Solution       *;
**********************************;
proc sql;
create table totalcustomer as
select State,count(*) as TotalCustomer
   from sq.customer
   group by State
         order by TotalCustomer desc;
quit;
```

Total rows: 51 Total columns: 2

	State	TotalCustomer
1	CA	18134
2	TX	9893
3	NY	6851
4	IL	5684
5	FL	5102

```
proc sql;
select c.State,
      c.TotalCustomer,
      s.EstimateBase,
      c.TotalCustomer/s.EstimateBase as PctCustomer format=percent7.3
   from totalcustomer as c inner join
        sq.statepopulation as s
   on c.State = s.Name
   order by PctCustomer;
quit;
```

State	TotalCustomer	EstimateBase	PctCustomer
VT	47	625744	.008%
WV	217	1853001	.012%
ME	159	1328369	.012%
DE	110	897934	.012%
MD	768	5773798	.013%
SC	745	4625381	.016%
NH	234	1316464	.018%
PA	2405	12702873	.019%
HI	263	1360307	.019%

```
****************************************;
* Using an In-Line View              *;
****************************************;
```

```
proc sql;
select c.State,
      c.TotalCustomer,
      s.EstimateBase,
      c.TotalCustomer/s.EstimateBase as PctCustomer format=percent7.3
   from (select State,count(*) as TotalCustomer
                     from sq.customer
                     group by State
                      ) as c inner join
        sq.statepopulation as s
     on c.State = s.Name
   order by PctCustomer;
quit;
```

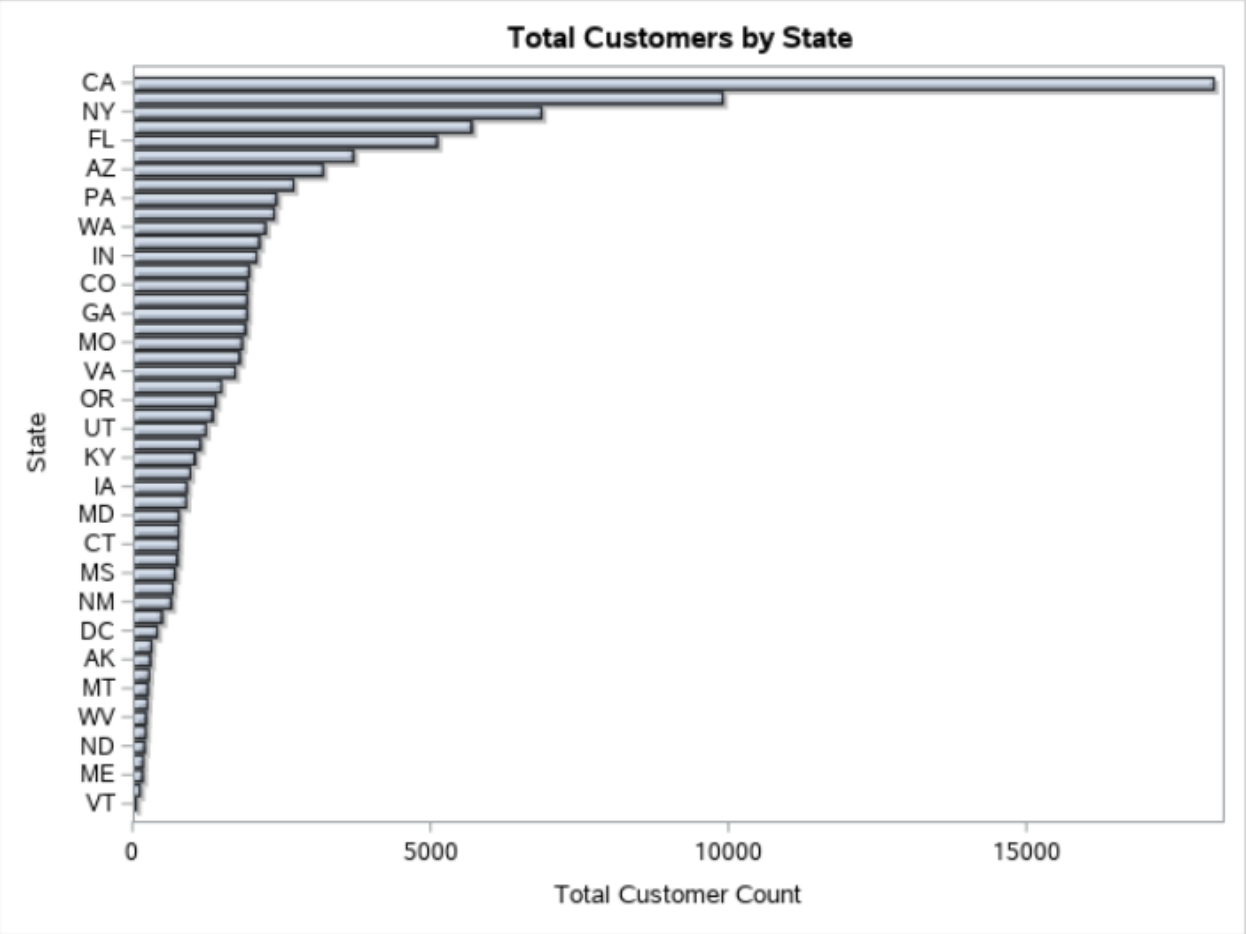
State	TotalCustomer	EstimateBase	PctCustomer
VT	47	625744	.008%
WV	217	1853001	.012%
ME	159	1328369	.012%
DE	110	897934	.012%
MD	768	5773798	.013%
SC	745	4625381	.016%
NH	234	1316464	.018%
PA	2405	12702873	.019%
HI	263	1360307	.019%

```
**********************************************************;
*  Activity 4.04                                 *;
*  1) Create a view named VWtotalcustomer from the query. *;
*     Run the query and examine the log.              *;
*  2) Run the code in the section Use the View in the     *;
*     PROCS Below. Which state has the highest number of  *;
*     customers?                                    *;
**********************************************************;
```

```
proc sql;
/*Create a View*/
Create view VWtotalcustomer as
select State,count(*) as TotalCustomer
   from sq.customer
   group by State;
quit;



*************************************************************;
*          USE THE VIEW IN THE PROCS BELOW          *;
*                                    DO NOT EDIT CODE BELOW            *;
*************************************************************;
title "Total Customers by State";
proc sql;
select *
   from VWtotalcustomer
          order by TotalCustomer desc;
quit;



proc sgplot data=VWtotalcustomer;
          hbar State / response=TotalCustomer
            dataskin=crisp
            categoryorder=respdesc;
   xaxis label="Total Customer Count";
quit;
title;
```

Total Customers by State

State	TotalCustomer
CA	18134
TX	9893
NY	6851
IL	5684
FL	5102
OH	3696
AZ	3185
MI	2687
PA	2405
NC	2362
WA	2221
NJ	2122
IN	2067
MN	1947
CO	1920
TN	1913
GA	1912
WI	1888
MO	1829
MA	1792



/*Practice Level 1: Using an In-Line View

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

In this practice you determine which customers have an extremely high credit score relative to other customers

in their ZIP code (Zip). Similar to the practice in the previous lesson,

an extremely high credit score is defined as greater than 2 standard deviations above the mean of CreditScore.

However, rather than use an overall high-credit threshold for all customers,

we need to calculate the threshold for each value of Zip. This can be accomplished using an in-line view.

Open s104p06.sas from the practices folder.

Run the query to summarize the HighZipCredit threshold for each Zip value for the first 1000 rows.

Use the query you just ran as an in-line view to join with the sq.customer table. Use the following requirements:

Select c.CustomerID, c.Zip, and c.CreditScore from the sq.customer table, and select s.HighZipCredit from the in-line view. Format the c.Zip column using the Z5. format.

Perform an inner join with the sq.customer table and the in-line view from step 1. Give the sq.customer table the alias c and the in-line view the alias s.

Remove the INOBS= option from the in-line view.

Use c.Zip = s.Zip as the join criteria.

Filter rows where the customer's c.CreditScore value is greater than the s.HighZipCredit value.

Order the results by Zip and descending CreditScore.

Add an appropriate title to the report.

Run the query and view the results.

What is the last Zip value and the corresponding CreditScore value in the final report?

*/

/*s104p06.sas*/

```
title 'High Credit Threshold by Zipcode';
proc sql inobs=1000;
select Zip,
     sum(avg(CreditScore),(2*std(CreditScore))) as HighZipCredit
   from sq.customer
          where CreditScore is not null
   group by Zip;
quit;
```

High Credit Threshold by Zipcode

Zip	HighZipCredit
6050	753.6741
6101	727.6
6320	640
6340	637
6360	783.7965
6418	690
6450	560
6457	864.3396

```
title "Customers with Above High Credit Threshold by Zipcode";
proc sql;
select c.customerID, c.Zip format=z5., c.CreditScore
        from sq.customer as c inner join (
                select Zip,
                sum(avg(CreditScore),(2*std(CreditScore))) as HighZipCredit
        from sq.customer
                where CreditScore is not null
        group by Zip) as z
   on c.Zip = z.Zip
   where c.CreditScore > z.HighZipCredit
   order by c.Zip, c.CreditScore desc;
quit;
title;
```

Customers with Above High Credit Threshold by Zipcode

Customer ID	Zip	CreditScore
1942378776	00501	812
1974119755	00501	793
1991947460	01027	798
1666368801	01085	845
1916975658	01101	847
1992140250	01101	813
1934252655	01101	797
1961755048	01752	808

/*Practice Level 2: Building a Complex Query Using In-Line Views

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

In this practice, you determine which employees have the highest salary for their job title in every state.

Using the sq.employee table, write a query to calculate the maximum Salary value for each value of JobTitle within each State.

Use the following requirements:

Convert the values of the State column to uppercase and name the column State to standardize the state code values.

Select JobTitle and calculate the maximum salary. Name the column MaxJobSalary.

Filter rows where State is not null.

Group the results by State and JobTitle.

Order the results by State.

Add an appropriate title.

Run the query and view the results.

What is the MaxJobSalary value for a Chief Executive Officer in FL?

Note: Enter a numeric value with no formatting (including commas).

*/

```
title "Maximum Salary by JobTitle within each State";

proc sql;

select upcase(e.State) as State, e.JobTitle, max(e.Salary) as MaxJobSalary

        from sq.employee as e

        where e.State is not null

        group by e.State, e.JobTitle

        order by e.State;

quit;

title;
```

Maximum Salary by JobTitle within each State

State	JobTitle	MaxJobSalary
CA	Accountant II	36230
CA	Accountant III	38545
CA	Applications Developer I	43630
CA	Auditor I	40450
CA	Auditor II	45100
CA	BI Specialist II	44425
CA	Chief Marketing Officer	207885
CA	Clerk I	27215
CA	Concession Assistant III	27685

/*Use the query you just ran as an in-line view to join with the sq.employee table. Use the following requirements:

Display EmployeeID, EmployeeName, State, JobTitle, and Salary for the highest paid employee

for each value of JobTitle in every state.

Perform an inner join with the sq.employee table and the in-line view from step 1.

Give the sq.employee table the alias detail and the in-line view the alias summary.

Hint: Remove the ORDER BY clause when using an in-line view.

Use JobTitle, State, and Salary equal to MaxJobSalary as the join criteria.

Order the report by State and JobTitle.

Add an appropriate title and format the Salary values with a dollar sign and comma.

Run the query and view the results.

What is the last name of the highest paid Auditor I in CA?

*/

```
title "Employee with Highest Salary by JobTitle within each State";

proc sql;

select detail.EmployeeID, detail.EmployeeName, upcase(detail.State) as State, detail.JobTitle,
detail.Salary format=dollar12.

        from sq.employee as detail inner join

            (select upcase(e.State) as State, e.JobTitle, max(e.Salary) as MaxJobSalary

                        from sq.employee as e

                        where e.State is not null
```

```
                        group by e.State, e.JobTitle) as summary
        on detail.State = summary.state
        and detail.JobTitle = summary.JobTitle
        and detail.Salary = summary.MaxJobSalary
        order by detail.State, detail.JobTitle;
quit;
title;
```

Employee with Highest Salary by JobTitle within each State

EmployeeID	EmployeeName	State	JobTitle	Salary
120759	Apr, Nishan	CA	Accountant II	$36,230
120757	Knopfmacher, Paul	CA	Accountant III	$38,545
120803	Droste, Victor	CA	Applications Developer I	$43,630
120764	Worton, Steven	CA	Auditor I	$40,450
120763	Capps, Ramond	CA	Auditor II	$45,100
120808	Dupree, Marcel	CA	BI Specialist II	$44,425
120260	Fletcher, Christine	CA	Chief Marketing Officer	$207,885

```
************************************************************;
* Remerging Summary Statistics                    *;
************************************************************;
* Syntax                                  *;
*                                       *;
* PROC SQL;                                 *;
*    SELECT col-name, summary function(column)        *;
*    FROM table;                              *;
* QUIT;                                   *;
************************************************************;


************************************************************;
* Demo                                    *;
* 1) Open the s104d04.sas program in the demos folder    *;
*    and find the Demo section. Run the query to select  *;
```

```
*    the Name and PopEstimate1 columns from the        *;
*    sq.statepopulation table.                         *;
*  2) In the SELECT clause, add the SUM function to sum  *;
*    PopEstimate1. Format the column using the COMMA12  *;
*    format. Run the query and examine log and results.  *;
*    Note: All values are the sum of the entire        *;
*         PopEstimate1 column, whereas the Name and    *;
*         PopEstimate1 columns have individual rows    *;
*         from the input table.                        *;
*  3) Modify the SELECT clause by dividing PopEstimate1  *;
*    and sum(PopEstimate1). Replace the COMMA12. format  *;
*    with the PERCENT7.2 format. Name the new column    *;
*    PctPop. Run the query and examine the log and      *;
*    results.                                          *;
*  4) Add an ORDER BY clause and sort the results by    *;
*    descending PctPop. Run the query and examine the   *;
*    results.                                          *;
*************************************************************;

proc sql;
select Name, PopEstimate1, sum(PopEstimate1)
   from sq.statepopulation;
quit;
```

```
73         proc sql;
74         select Name, PopEstimate1, sum(PopEstimate1) format=comma12.
75             from sq.statepopulation;
NOTE: The query requires remerging summary statistics back with the original data.
76         quit;
```

Name	PopEstimate1	
AL	4864745	3.2648E8
AK	741504	3.2648E8
AZ	6945452	3.2648E8
AR	2990410	3.2648E8
CA	39209127	3.2648E8

```
proc sql;
select Name, PopEstimate1/sum(PopEstimate1) as PctPop format=percent7.2
   from sq.statepopulation
   order by PctPop desc;
quit;
```

Name	PctPop
CA	12.0%
TX	8.56%
FL	6.32%
NY	6.02%
IL	3.93%
PA	3.92%

```
************************************************************;
*  Activity 4.05                                           *;
*  1) Examine and run the query. Examine the log and the   *;
*     results. What note do you see in the log?            *;
*  2) Add the PROC SQL option NOREMERGE. Run the query.    *;
*     Did it run successfully? What was the error in the   *;
*     log?                                                 *;
*  3) Add a GROUP BY clause after the FROM clause and      *;
*     group by Region. Run the query. Did it run           *;
*     successfully?                                        *;
************************************************************;

proc sql;
select Region,
```

sum(PopEstimate1) as TotalRegion format=comma14.

from sq.statepopulation;

quit;

```
73          proc sql;
74          select Region,
75                 sum(PopEstimate1) as TotalRegion format=comma14.
76             from sq.statepopulation;
NOTE: The query requires remerging summary statistics back with the original data.
77          quit;
```

Region	TotalRegion
3	326,477,837
4	326,477,837
4	326,477,837
3	326,477,837
4	326,477,837

proc sql noremerge;

select Region,

sum(PopEstimate1) as TotalRegion format=comma14.

from sq.statepopulation;

quit;

```
73          proc sql noremerge;
74          select Region,
75                 sum(PopEstimate1) as TotalRegion format=comma14.
76             from sq.statepopulation;
ERROR: The query requires remerging summary statistics back with the original data.  This is disallowed due to the NOREMERGE proc
       option or NOSQLREMERGE system option.
NOTE: PROC SQL set option NOEXEC and will continue to check the syntax of statements.
77          quit;
```

proc sql noremerge;

select Region,

sum(PopEstimate1) as TotalRegion format=comma14.

from sq.statepopulation

group by Region;

quit;

Region	TotalRegion
1	56,058,789
2	67,996,917
3	122,401,186
4	76,614,450
X	3,406,495

/*Practice Level 1: Remerging Summary Statistics

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

Use the sq.statepopulation table to determine which states have the most estimated births for next year.

Include a column showing each state's births as the percent of national births.

Write a query to remerge summary statistics in the sq.statepopulation table. Use the following requirements:

Select the Name and Births columns.

Create a new column named PctBirth by dividing Births1 for each state by the sum of Births1 for all states.

Format the new column using the PERCENT format.

Order the results by descending PctBirth.

Add an appropriate title to the report.

Run the query and view the results.

Which state has the highest percentage of estimated births for next year?

*/

/*s104s09.sas*/

```
title 'Estimate Percentage of Births by Each State';
proc sql;
select Name, Births1,
    Births1/sum(Births1) as PctBirth format=percent7.2
```

```
   from sq.statepopulation
   order by PctBirth desc;
quit;
title;

/*Alternate Solution*/
title 'Estimate Percentage of Births by Each State';
proc sql;
select Name, Births1,
      Births1/(select sum(Births1)
                  from sq.statepopulation)
                     as PctBirth format=percent7.2
   from sq.statepopulation
   order by PctBirth desc;
quit;
title;
```

Estimate Percentage of Births by Each State

Name	Births1	PctBirth
CA	490384	12.3%
TX	400968	10.0%
NY	235826	5.91%
FL	224643	5.63%
IL	156299	3.91%
PA	140256	3.51%
OH	138674	3.47%

/*Practice Level 2: Using a Subquery in the SELECT Clause with an In-Line Views

If necessary, start SAS Studio before you begin.

If you restarted your SAS session, submit your libname.sas program to access the practice data.

Find the top 10 countries that have the highest percentage of estimated global population using the population estimates of ages 15+ in the sq.globalfull table.

Use the sq.globalfull table to write a query to sum the EstYear1Pop of all countries.

Use the following requirements:

Use an in-line view to select the distinct CountryCode and EstYear1Pop values from the sq.globalfull table.

Note: You must find the distinct estimated population of each country because

the data contains the estimated population for each country multiple times.

Sum the EstYear1Pop column and name the column new column EstPct. Format the column using the COMMA format.

Run the query and view the results.

What is the value of EstPct? Note: Copy and paste the value from the results or type it exactly as shown in the results.

*/

```
proc sql;

select distinct CountryCode, EstYear1Pop

        from sq.globalfull;

quit;
```

CountryCode	EstYear1Pop
AFG	17922774
AGO	14211077
ALB	2347104
ARE	7831304
ARG	32086629
ARM	2333626
AUS	19077182

```
/*s104s10.sas*/

/*part 1*/

title 'Estimated Population of Ages 15+ for Next Year';

proc sql;

select sum(EstYear1Pop) as EstPct format=comma16.

   from (select distinct CountryCode, EstYear1Pop

        from sq.globalfull);

quit;

title;
```

Estimated Population of Ages 15+ for Next Year

EstPct
5,271,101,653

/*Create a new query using the query you just ran to determine the estimated global population percentage of each country.

Use the following requirements:

Select the distinct CountryCode and ShortName values.

Determine the percentage population by creating a new column.

Divide the EstYear1Pop value of each country by the total value calculated in step 1.

Name the new column PctPop and format using the PERCENT format.

Use the sq.globalfull table.

Order the results by descending PctPop.

Limit the results to the top 10 countries.

Add an appropriate title.

Run the query and view the results.

Which country has the highest estimated population of individuals 15 or over?

*/

```
proc sql;
select distinct CountryCode, ShortName
        from sq.globalfull;
quit;
```

CountryCode	ShortName
AFG	Afghanistan
AGO	Angola
ALB	Albania
ARE	United Arab Emirates
ARG	Argentina
ARM	Armenia
AUS	Australia

/*part 2*/

```
title 'Top 10 Countries by Estimate Population';
```

```
title2 'Ages 15+';
proc sql outobs=10;
select distinct CountryCode, ShortName,
      EstYear1Pop/
            (select sum(EstYear1Pop) as EstPct format=comma16.
               from (select distinct CountryCode, EstYear1Pop
                        from sq.globalfull))
                           as PctPop format=percent7.2
   from sq.globalfull
   order by PctPop desc;
quit;
title;
```

Top 10 Countries by Estimate Population Ages 15+

CountryCode	ShortName	PctPop
CHN	China	21.3%
IND	India	17.4%
USA	United States	4.87%
IDN	Indonesia	3.48%
BRA	Brazil	2.98%
PAK	Pakistan	2.28%
RUS	Russia	2.28%
BGD	Bangladesh	2.12%
JPN	Japan	2.10%
NGA	Nigeria	1.87%