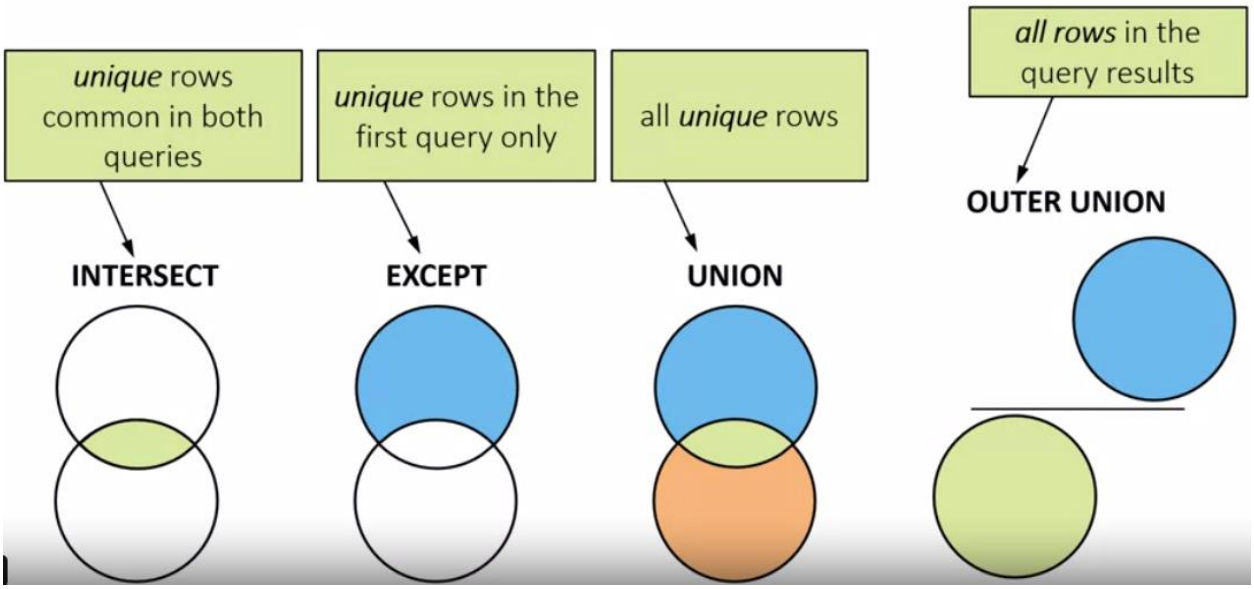
SAS Advanced Programmer

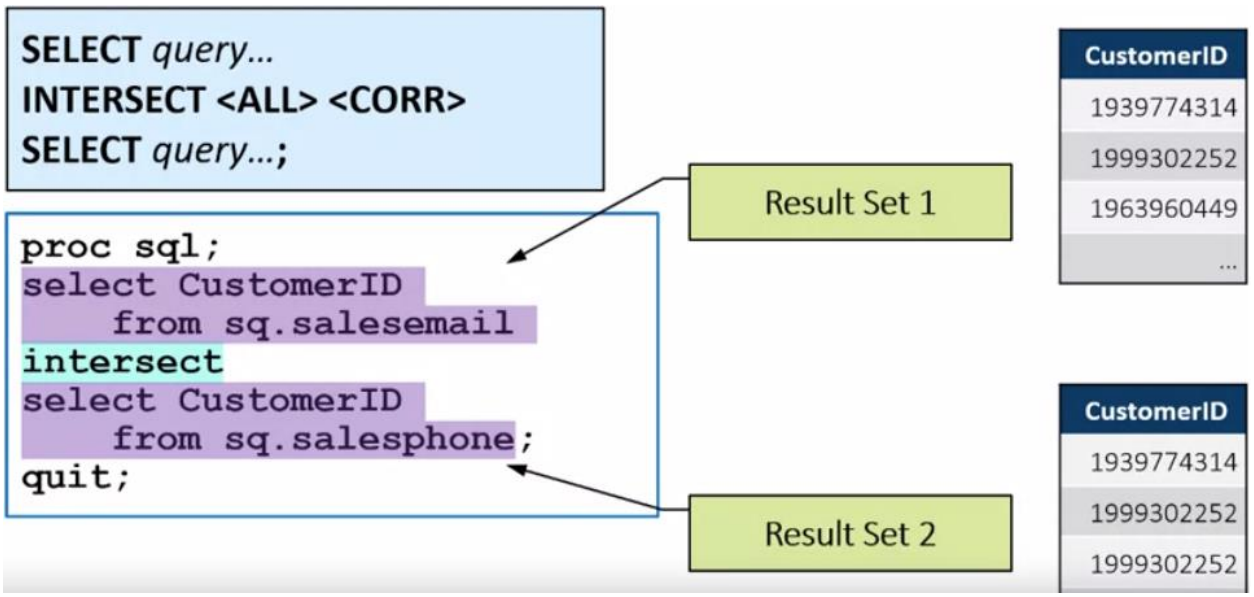
SAP105 Set Operators and Macro Variables

/* Defines the path to your data and assigns the libref. */

%let path=~/ESQ1M6;

libname sq "&path/data";





**********;

* Activity 5.01 *;

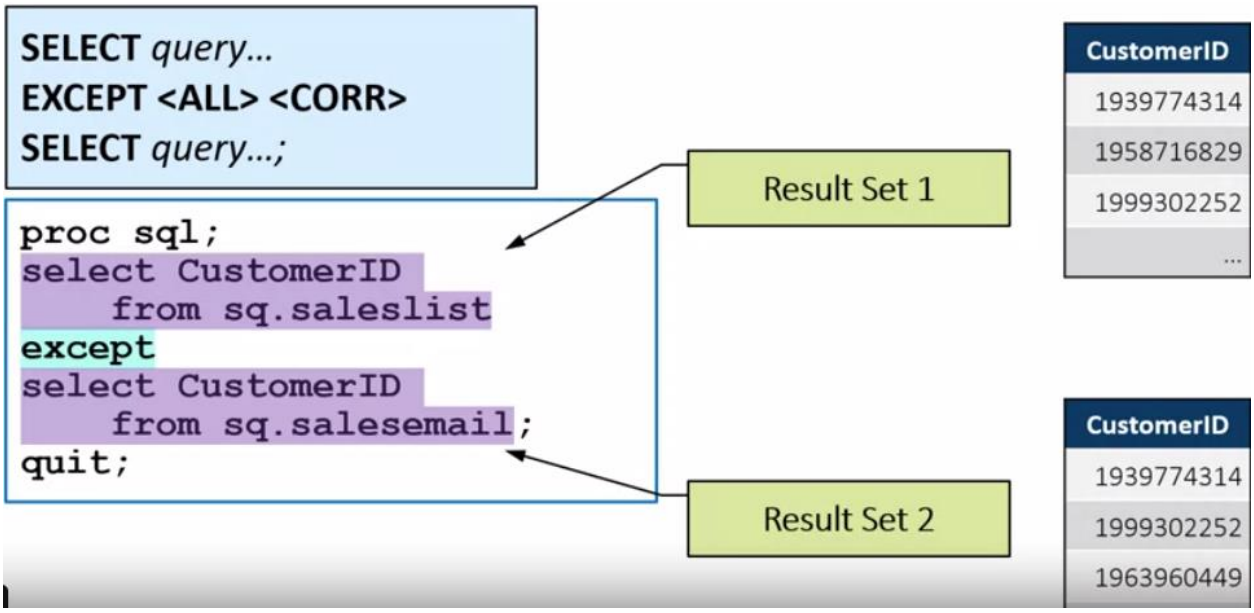
```
*  1) Run the first queries to preview the sq.salesemail  *;
*     and sq.salesphone tables. Examine the columns in    *;
*     both tables.                                        *;
*  2) In the Intersect section, examine and run the       *;
*     query. Did the query run successfully? Why not?     *;
*  3) Add the CORR keyword after the INTERSECT set        *;
*     operator. Run the query. Did the query run          *;
*     successfully? Why?                                  *;
************************************************************;


**********;
*Preview Tables     *;
**********;
proc sql inobs=5;
select *
   from sq.salesemail;
select *
   from sq.salesphone;
quit;
```

CustomerID	EmailResp
1939774314	Accepted
1999302252	Declined
1963960449	Declined
1908694347	Accepted
1960311448	Accepted

CustomerID	SalesRep	PhoneResp
1939774314	121038	Declined
1999302252	120145	Call Back
1999302252	120145	Accepted
1963960449	120145	Declined
1987175132	120145	Declined

```
**********;
*Intersect        *;
**********;
proc sql;
select *
   from sq.salesemail
intersect corr
select *
   from sq.salesphone;
quit;
```

CustomerID
1939774314
1963960449
1999302252




```
**********************************************************;
*  Activity 5.02                                         *;
*  1) Run the first queries to preview the sq.saleslist  *;
*     and sq.salesphone tables. Examine the columns in   *;
*     both tables.                                       *;
*  2) Complete the query to find all customers from the  *;
*     sq.saleslist table who have not responded to our   *;
*     sales call in sq.salesphone.                       *;
*  3) How many customers have not responded to our phone *;
*     call?                                              *;
**********************************************************;


*********************;
*Preview Tables     *;
*********************;
proc sql inobs=5;
select *
   from sq.saleslist;
```

```
select *
   from sq.salesphone;
quit;
```

UserID	CustomerID	CellPhone	HomePhone
marremartinez6531@ismissing.com	1939774314	(630)7734430	(630)1642888
kimlihuffman843@fakeemail.com	1958716829		(212)2023592
heacamoredock827@invalid.com	1999302252	(212)6871017	(212)1618048
heljaboone8613@invalid.com	1963960449		(212)7089847
jambeerskin7521@fakeemail.com	1987175132	(337)2330976	(337)0369173

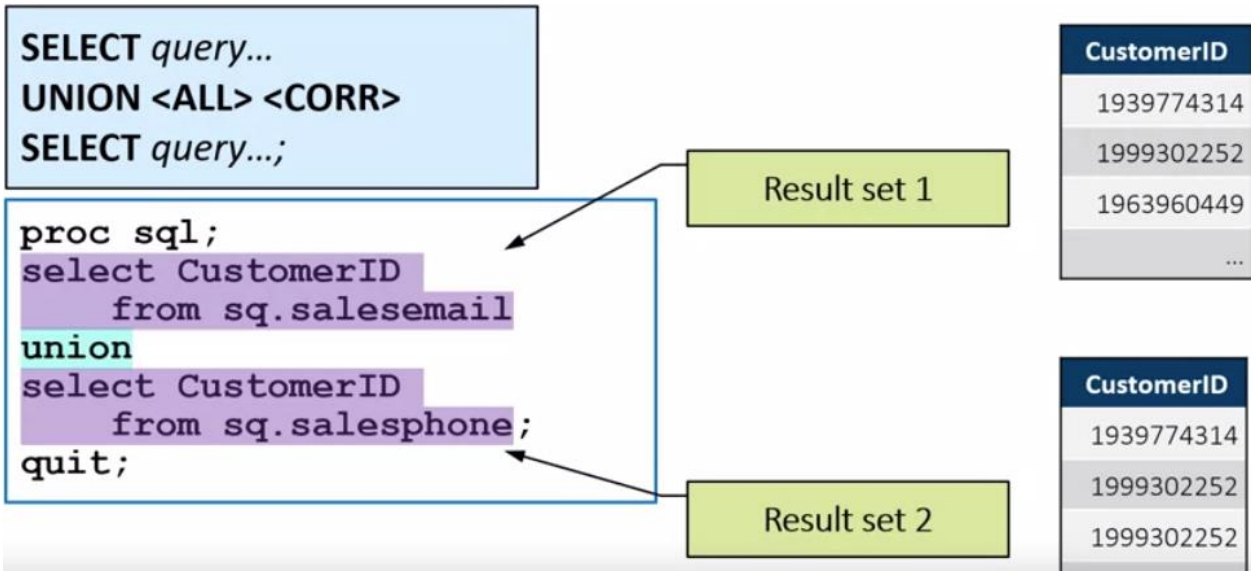
CustomerID	SalesRep	PhoneResp
1939774314	121038	Declined
1999302252	120145	Call Back
1999302252	120145	Accepted
1963960449	120145	Declined
1987175132	120145	Declined

```
**********;
*Except          *;
**********;
proc sql;
select *
   from sq.saleslist
except corr
select *
   from sq.salesphone;
quit;

/* OR */

select CustomerID
   from sq.saleslist
except
```

```
select CustomerID
   from sq.salesphone;
quit;
```

CustomerID
1905044343
1908694347
1918638906
1958716829
1960311448

```
select distinct CustomerID
   from sq.saleslist
   where CustomerID not in(select CustomerID
                           from sq.salesphone);
```




```
**********************************************************;
*  Using the UNION Set Operator                          *;
**********************************************************;
*  Syntax                                                *;
*                                                        *;
*  PROC SQL;                                             *;
*  SELECT query                                          *;
```

```
*    UNION <ALL> <CORR>                          *;
*  SELECT query;                                 *;
*  QUIT;                                         *;
************************************************************;

************************************************************;
*  Demo                                          *;
*  1) Open the s105d01.sas program in the demos folder    *;
*    and find the Demo section. Under Explore the         *;
*    SALESEMAIL and SALESPHONE Tables, run the two        *;
*    queries. Discuss the tables and the desired          *;
*    outcome.                                    *;
*  2) In the next section, complete the query to find all *;
*    unique customers who responded to either an email    *;
*    or phone call. Begin with the SELECT statement and   *;
*    select all columns from the sq.salesemail table.     *;
*  3) Use the UNION set operator followed by another      *;
*    SELECT statement to select all columns from the      *;
*    sq.salesphone table. Run the query and examine the   *;
*    syntax error.                               *;
*  4) Add the keyword CORR after the UNION set operator.  *;
*    Run the query and examine the log and results.       *;
*    Note: The CORR keyword aligns the columns that have *;
*          the same name in both tables and removes any   *;
*          columns not found in both tables.              *;
*  5) Remove the CORR keyword, and specify the CustomerID *;
*    column in both SELECT clauses. Run the query and     *;
*    examine the log and results.                *;
*  6) Add another SELECT statement at the first line in   *;
```

```
*    the SQL procedure. Use the COUNT(*) function to     *;
*    count all rows. Name the column TotalNum. Add a     *;
*    FROM clause and use the previous query as a         *;
*    subquery in the FROM clause (in-line view). Be sure *;
*    to add parentheses around the subquery. Run the     *;
*    query and examine the results.                      *;
************************************************************;


*********************************************;
*Explore the SALESEMAIL and SALESPHONE tables   *;
*********************************************;
proc sql;
select *
   from sq.salesemail;
select *
   from sq.salesphone;
quit;
```

CustomerID	EmailResp
1939774314	Accepted
1999302252	Declined
1963960449	Declined
1908694347	Accepted
1960311448	Accepted
1905044343	Declined

CustomerID	SalesRep	PhoneResp
1939774314	121038	Declined
1999302252	120145	Call Back
1999302252	120145	Accepted
1963960449	120145	Declined
1987175132	120145	Declined
1970095442	121137	Accepted

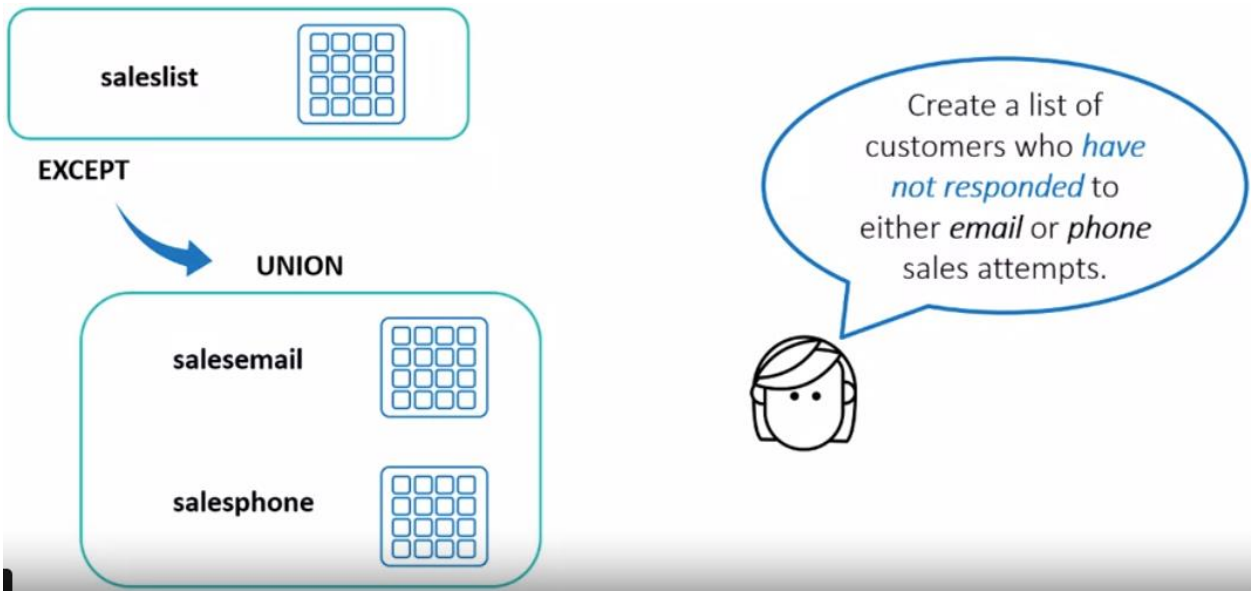
```
*************************************************;
*Find the Number of Unique Customers          *;
*************************************************;
proc sql;
select *
   from sq.salesemail
   union corr
select *
   from sq.salesphone;
quit;


proc sql;
select CustomerID
   from sq.salesemail
   union
select customerID
   from sq.salesphone;
quit;
```

CustomerID
1905044343
1908694347
1939774314
1960311448
1963960449
1970095442
1987175132
1999302252

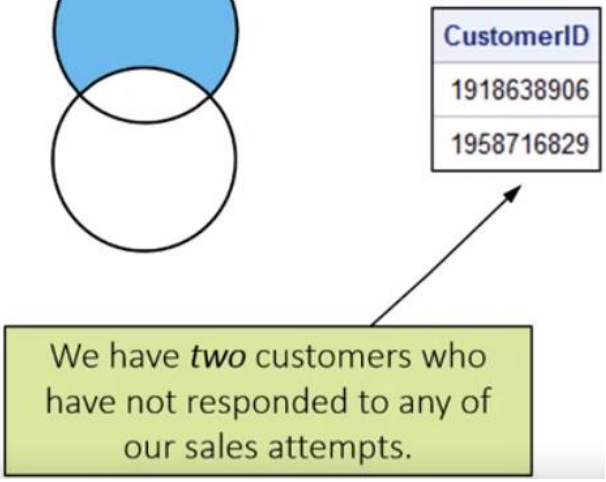
```
proc sql;
select count(*) as TotalNum
from (select CustomerID
          from sq.salesemail
```

```
        union
         select customerID
        from sq.salesphone);
quit;
```

TotalNum
8




```
proc sql;
select CustomerID
    from sq.saleslist
except
(select customerid
    from sq.salesemail
union
 select customerID
    from sq.salesphone);
quit;
```



/*Practice Level 1: Using the EXCEPT Set Operator

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 EXCEPT set operator to generate a report listing of merchants in the sq.merchant table who are not listed in the sq.transaction table.

Write a query using the following requirements:

Select MerchantID from the sq.merchant table.

Use the EXCEPT set operator.

Select MerchantID from the sq.transaction table.

Order the results by MerchantID.

Add an appropriate title.

Run the query and view the results.

How many merchants do not have a transaction? Note: Type a numeric value.

*/

```
title "List of merchants not in transaction";
proc sql;
select MerchantID
        from sq.merchant
        except
select MerchantID
        from sq.transaction
        order by MerchantID;
quit;
title;
```

List of merchants not in transaction

Merchant ID
502136
517039
518339
565543
565603
585048

/*Practice Level 2: Using the EXCEPT Set Operator with the DISTINCT Keyword

If necessary, start SAS Studio before you begin.

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

Using the sq.statepopulation table, generate a list of state codes for states without any customers.

Note: In this data, DC an PR are values for State.

Write a query to list the unique Name values in the sq.statepopulation table. This list represents all available states.

Note: In this data, DC an PR are valid values for State.

Write a separate query to list the unique State values from the sq.customer table.

This list represents all states where customers reside.

Combine the queries using the EXCEPT set operator to display states with no customers.

Add an appropriate title and run the query.

Which state has no customers?

*/

```
title "State without Customers";

proc sql;

select distinct p.Name

        from sq.statepopulation as p

        except

select distinct c.State

        from sq.customer as c;

quit;

title;
```

State without Customers

Name
PR

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

*  Using the OUTER UNION Set Operator                *;

************************************************************;

*  Syntax                                *;
```

```
*                                   *;
*  PROC SQL;                              *;
*    SELECT query                          *;
*    OUTER UNION <CORR>                          *;
*    SELECT query;                         *;
*  QUIT;                                *;
*                                   *;
*  table(RENAME=(old-name-1=new-name-1))          *;
************************************************************;

************************************************************;
*  Demo                                *;
*  1) Open the s105d02.sas program in the demos folder    *;
*    and find the Demo section. Run the query to perform *;
*    an OUTER UNION concatenation of the sq.salesemail   *;
*    and sq.salesphone tables. Examine the results.      *;
*  2) Add the CORR keyword after the OUTER UNION set      *;
*    operator. Run the query and examine the results.    *;
*  3) Add the RENAME= option after the salesemail table   *;
*    and rename the column EmailResp to Resp. After the  *;
*    salesphone table, rename the column PhoneResp to    *;
*    Resp. Run the query and examine the results.        *;
*  4) Remove the RENAME= option after each table. Modify  *;
*    the first SELECT clause and select the column       *;
*    CustomerID. In the first clause, also select        *;
*    EmailResp and change the column name to Resp using  *;
*    the AS keyword. Modify the second SELECT clause and *;
*    select the CustomerID, SalesRep, and PhoneResp      *;
*    columns. Change the PhoneResp column name to Resp   *;
```

```
*    using the AS keyword. Run the query and examine the *;
*    results.                                *;
*  5) Add the CREATE TABLE statement to create a table    *;
*    from the query results. Name the table response1.   *;
*  6) Find the SAS DATA Step section. Briefly describe    *;
*    how you can retrieve the same results using the SAS *;
*    DATA step.                              *;
************************************************************;

title "All Customer Responses";
title2 "Phone and Email";
proc sql;
select *
   from sq.salesemail
   outer union
select *
   from sq.salesphone;
quit;
title;
```

All Customer Responses
Phone and Email

CustomerID	EmailResp	CustomerID	SalesRep	PhoneResp
1939774314	Accepted	.	.	
1999302252	Declined	.	.	
1963960449	Declined	.	.	
1908694347	Accepted	.	.	
1960311448	Accepted	.	.	
1905044343	Declined	.	.	
.		1939774314	121038	Declined
.		1999302252	120145	Call Back
.		1999302252	120145	Accepted
.		1963960449	120145	Declined
.		1987175132	120145	Declined
.		1970095442	121137	Accepted

```
title "All Customer Responses";
title2 "Phone and Email";
proc sql;
select *
   from sq.salesemail (rename=(EmailResp=Resp))
   outer union corr
select *
   from sq.salesphone (rename=(PhoneResp=Resp));
quit;
title;


title "All Customer Responses";
title2 "Phone and Email";
proc sql;
select CustomerId, EmailResp as Resp
   from sq.salesemail
   outer union corr
select CustomerId, SalesRep, PhoneResp as Resp
```

```
   from sq.salesphone;
quit;
title;
```

All Customer Responses
Phone and Email

CustomerID	Resp	SalesRep
1939774314	Accepted	.
1999302252	Declined	.
1963960449	Declined	.
1908694347	Accepted	.
1960311448	Accepted	.
1905044343	Declined	.
1939774314	Declined	121038
1999302252	Call Back	120145
1999302252	Accepted	120145
1963960449	Declined	120145
1987175132	Declined	120145
1970095442	Accepted	121137

```
title "All Customer Responses";
title2 "Phone and Email";
proc sql;
create table response1 as
select CustomerId, EmailResp as Resp
   from sq.salesemail
   outer union corr
select CustomerId, SalesRep, PhoneResp as Resp
   from sq.salesphone;
quit;
title;

**********;
*  SAS DATA Step    *;
**********;
```

```
data response2;
   length Resp $12;
   set sq.salesemail(rename=(EmailResp=Resp))
      sq.salesphone(rename=(PhoneResp=Resp));
run;
```

Table: WORK.RESPONSE1 | View: Column names | Filter:

Columns

- Select all
- CustomerID
- Resp
- SalesRep

Total rows: 12 Total columns: 3

	CustomerID	Resp	SalesRep
1	1939774314	Accepted	.
2	1999302252	Declined	.
3	1963960449	Declined	.
4	1908694347	Accepted	.
5	1960311448	Accepted	.
6	1905044343	Declined	.
7	1939774314	Declined	121038
8	1999302252	Call Back	120145
9	1999302252	Accepted	120145
10	1963960449	Declined	120145
11	1987175132	Declined	120145
12	1970095442	Accepted	121137

/*Practice: Using the OUTER UNION Set Operator

If necessary, start SAS Studio before you begin.

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

Create a report that shows the email and phone offer responses along with the sales representative if available.

Using the sq.salesphone table as input, write a query to list the following columns:

CustomerID

a new column named Response based on the existing PhoneResp column

SalesRep labeled Sales Rep

a new column named Channel with the constant text Phone

Run the query and view the results.

Using the sq.salesemail table as input, write a query to list the following columns:

CustomerID

a new column named Response based on the existing EmailResp column

a new column named Channel with the constant text Email

Run the query and view the results.

Combine the two query results using the OUTER UNION set operation. Use the following requirements:

Be mindful of the column alignment. Use the SET operator modifiers as needed.

Order the results by CustomerID and Response.

Add an appropriate title.

Run the query and view the final results.

How many identifiable sales representatives had an accepted offer? Note: Type a numeric value.

*/

```
title "Email and Phone Offer Responses with Sales Rep";
proc sql;
select CustomerID, PhoneResp as Response, SalesRep 'Sales Rep', 'Phone' as Channel
        from sq.salesphone
        outer union corr
select CustomerId, EmailResp as Response, 'Email' as Channel
        from sq.salesemail
        order by CustomerID, Response;
quit;
title;
```

Email and Phone Offer Responses with Sales Rep

CustomerID	Response	Sales Rep	Channel
1905044343	Declined	.	Email
1908694347	Accepted	.	Email
1939774314	Accepted	.	Email
1939774314	Declined	121038	Phone
1960311448	Accepted	.	Email
1963960449	Declined	.	Email
1963960449	Declined	120145	Phone
1970095442	Accepted	121137	Phone
1987175132	Declined	120145	Phone
1999302252	Accepted	120145	Phone
1999302252	Call Back	120145	Phone
1999302252	Declined	.	Email

```
/*s105s05.sas*/

title 'Offer Results with Sales Rep';
proc sql;
select CustomerID,
      PhoneResp as Response,
      SalesRep 'Sales Rep',
      'Phone' as Channel
   from sq.salesphone
   outer union corr
select CustomerID,
      EmailResp as Response,
     'Email' as Channel
   from sq.salesemail
   order by 1,2;
quit;
title;
```

```
************************************************************;
*  Activity 6.01                                *;
*  1) Run the program. Examine the log and results.      *;
*     Confirm that the name of the newly created table is *;
*     customerga and contains 957 rows.                  *;
*  2) Replace the values GA and 650 in the %LET          *;
*     statements with NC and 700. Run the program.       *;
*     Examine the log and results. What is the name of   *;
*     the newly created table? How many rows?            *;
*  3) Change the double quotation marks in the WHERE     *;
*     clause expression to single quotation marks. Run   *;
*     the query. How many rows are in the new table?     *;
************************************************************;

%let State=GA;
%let CreditMin=650;
proc sql;
create table Customer&State as
select CustomerID, Employed, Race,
      Married, State, CreditScore
   from sq.customer
   where State="&State" and
         CreditScore > &CreditMin;
quit;
```

Table: WORK.CUSTOMERGA | View: Column names | Filter: (none)

Columns: Select all, CustomerID, Employed, Race, Married, State, CreditScore

Total rows: 957 Total columns: 6 — Rows 1-100

	CustomerID	Employed	Race	Married	State	CreditScore
1	1947051903	Y	B		GA	747
2	1979976713	N	W	D	GA	768
3	1912807991	N	B	M	GA	689
4	1902970238	N	W	M	GA	673
5	1989851187	Y	W		GA	674
6	1946575893	Y	W		GA	702
7	1977080370	N	B	M	GA	665
8	1924464624	Y	B	M	GA	680

```
%let State=NC;
%let CreditMin=700;
proc sql;
create table Customer&State as
select CustomerID, Employed, Race,
      Married, State, CreditScore
   from sq.customer
   where State="&State" and
         CreditScore > &CreditMin;
quit;
```

Table: WORK.CUSTOMERNC | View: Column names | Filter: (none)

Columns: Select all, CustomerID, Employed, Race, Married, State, CreditScore

Total rows: 700 Total columns: 6 — Rows 1-100

	CustomerID	Employed	Race	Married	State	CreditScore
1	1927226127	N	W	W	NC	713
2	1967842633	Y	B	M	NC	711
3	1911681153	Y	W	M	NC	717
4	1909086342	N	W	M	NC	772
5	1950971444	N	W	D	NC	733
6	1987841671	Y	W	M	NC	758
7	1923642050	Y	W	M	NC	733

```
************************************************************;
*  Using a PROC SQL Data-Driven Macro Variable          *;
************************************************************;
*  Syntax                                  *;
*                                        *;
*  PROC SQL NOPRINT;                          *;
*    SELECT col-name                          *;
*    INTO :macvar                            *;
*    FROM table;                             *;
*  QUIT;                                   *;
*                                        *;
*  %PUT &=macvar;                             *;
*  OPTIONS SYMBOLGEN;                          *;
************************************************************;


************************************************************;
*  Demo                                    *;
*  1) Open the s106d01.sas program in the demos folder    *;
*    and find the Demo section. Run the first query in   *;
*    the Create the Macro Variable Using PROC SQL       *;
*    section. View the results.                  *;
*  2) In the same query, add the NOPRINT option in the    *;
*    PROC SQL statement. At the end of the query, add a  *;
*    %PUT statement to view the value of the newly      *;
*    created macro variable in the SAS log. Run the     *;
*    query and view the SAS log.                 *;
*  3) Move to STEP 2 and apply the newly created macro    *;
*    variable AvgEst1 in the TITLE statement and the     *;
*    WHERE clause. Run the query. View the log and      *;
```

```
*    results.                              *;
*    Note: To format a macro variable in the TITLE      *;
*         statement, you can use                  *;
*         %left(%qsysfunc(putn(&AvgEst1,dollar16.))).   *;
*         Scroll to the bottom of the program file for  *;
*         this demo to view the solution code.         *;
*  4) Above the query, add the statement OPTIONS         *;
*    SYMBOLGEN to see the value that was substituted in  *;
*    the code. Below the query, turn of the options by   *;
*    adding the OPTIONS NOSYMBOLGEN statement. Run the   *;
*    query and examine the log.                  *;
************************************************************;

************************************************;
*STEP 1: Create the Macro Variable Using PROC SQL *;
************************************************;
proc sql;
select avg(PopEstimate1)
   into :AvgEst1
         from sq.statepopulation;
quit;
```

6278420

```
proc sql noprint;
select avg(PopEstimate1)
   into :AvgEst1
         from sq.statepopulation;
quit;
%put &=AvgEst1;
```

```
78          %put &=AvgEst1;
AVGEST1= 6278420
```

**********;

*STEP 2: Use the Macro Variable *;

**********;

```
title "Average Estimated Population for Next Year: &AvgEst1";/*Add macro variable*/
proc sql;
select Name, PopEstimate1
   from sq.statepopulation
   where PopEstimate1 > &AvgEst1;
quit;
title;
```

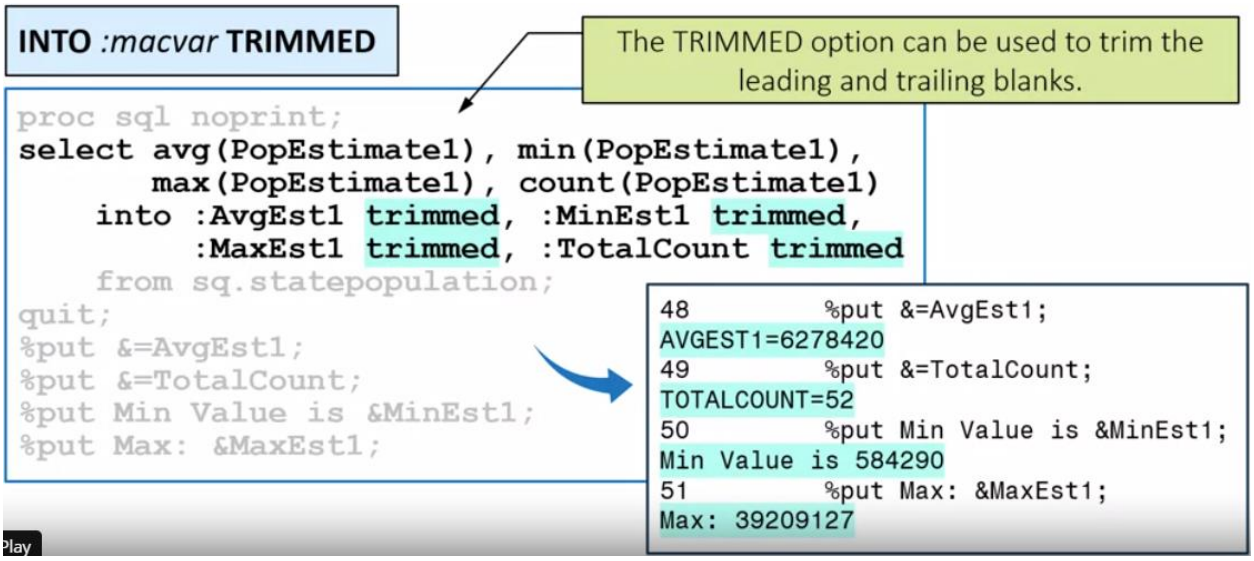
Average Estimated Population for Next Year: 6278420

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

```
options symbolgen;
title "Average Estimated Population for Next Year: &AvgEst1";/*Add macro variable*/
proc sql;
```

```
select Name, PopEstimate1
   from sq.statepopulation
   where PopEstimate1 > &AvgEst1;
quit;
title;
options nosymbolgen;
```

```
73         options symbolgen;
SYMBOLGEN:  Macro variable AVGEST1 resolves to  6278420
74         title "Average Estimated Population for Next Year: &AvgEst1";/*Add macro variable*/
75         proc sql;
76         select Name, PopEstimate1
77             from sq.statepopulation
78             where PopEstimate1
SYMBOLGEN:  Macro variable AVGEST1 resolves to  6278420
78       !                        > &AvgEst1;
79         quit;
```




```
**********************************************************;
*  Activity 6.02                          *;
*  1) Run the program in STEP 1 to create the macro      *;
*     variables MaxPop and TotalCtry. The macro variables *;
*     store the estimated maximum three-year population   *;
*     estimate for a country and the total number of      *;
*     countries. View the log. Notice that MaxPop stores  *;
*     a value in scientific notation.                     *;
```

```
*  2) Examine STEP 2 of the program to find the country   *;
*     with the maximum estimated three-year population.   *;
*     Run the program to use the macro variables that you *;
*     created. Confirm that no rows were returned.        *;
*  3) Add the FORMAT=10. column modifier to the           *;
*     max(estYear3Pop) column in STEP 1. Run the entire   *;
*     program. Which country has the largest three-year   *;
*     estimated population?                               *;
************************************************************;


*********;
*STEP 1 *;
*********;
proc sql noprint;
select max(EstYear3Pop), count(distinct CountryCode)
   into :MaxPop trimmed, :TotalCtry trimmed
   from sq.globalfull;
quit;
%put &=MaxPop &=TotalCtry;
```

```
91          *********;
92          *STEP 1 *;
93          *********;
94          proc sql noprint;
95          select max(EstYear3Pop), count(distinct CountryCode)
96              into :MaxPop trimmed, :TotalCtry trimmed
97              from sq.globalfull;
98          quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.01 seconds
      user cpu time       0.00 seconds
      system cpu time     0.00 seconds
      memory              6928.15k
      OS Memory           37036.00k
      Timestamp           05/25/2021 05:08:36 AM
      Step Count                        53  Switch Count  0
      Page Faults                       0
      Page Reclaims                     451
      Page Swaps                        0
      Voluntary Context Switches        12
      Involuntary Context Switches      0
      Block Input Operations            288
      Block Output Operations           8


99          %put &=MaxPop &=TotalCtry;
MAXPOP=1.1413E9 TOTALCTRY=151
```

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

*STEP 1 *;

*********;

proc sql noprint;

select max(EstYear3Pop) format=10., count(distinct CountryCode)

   into :MaxPop trimmed, :TotalCtry trimmed

   from sq.globalfull;

quit;

%put &=MaxPop &=TotalCtry;
```

*********;

*STEP 2 *;

*********;

```
title "Country with the Largest 3 Year Estimated Population";
title2 "Out of &TotalCtry Countries";
proc sql;
select distinct CountryCode, ShortName, Region,
    EstYear3Pop format=comma16.
  from sq.globalfull
  where EstYear3Pop = &MaxPop;
quit;
title;
```

Country with the Largest 3 Year Estimated Population
Out of 151 Countries

CountryCode	ShortName	Region	EstYear3Pop
CHN	China	East Asia & Pacific	1,141,324,637

```
81          %put &=MaxPop &=TotalCtry;
MAXPOP=1141324637 TOTALCTRY=151
```

**********;

* Concatenating Values in Macro Variables *;

**********;

* Syntax *;

* *;

* PROC SQL; *;

* SELECT col-name *;

* INTO :macvar SEPARATED BY "delimiter" *;

* FROM table; *;

```
*  QUIT;                                *;
*                                     *;
*  QUOTE(argument)                     *;
*  STRIP(argument)                     *;
************************************************************;

************************************************************;
*  Demo                                *;
*  1) Open the s106d02.sas program in the demos folder    *;
*     and find the Demo section. Run the code in Step 1:  *;
*     Create Macro Variables. Examine the results and log *;
*     to see the values of the newly created macro        *;
*     variables &Division and &StateList.                 *;
*  2) Discuss the query under Step 2: Use Macro           *;
*     Variables. The table being created will end with    *;
*     the value of the macro variable &Division. The      *;
*     customer table will attempt to be subset by a list  *;
*     of values from the &StateList macro variable. Run   *;
*     the query and examine the errors.                   *;
*     Note: The name of the new table will end with the   *;
*           value of the macro variable.                  *;
*  3) Move back to Step 1. Add the QUOTE function around  *;
*     the Name column. Run the query and %PUT statements. *;
*     Examine the results and log.                        *;
*  4) Add the STRIP function inside the QUOTE function.   *;
*     Run the query and %PUT statements. Examine the      *;
*     results and log.                                    *;
*  5) Move to Step 2. Run the query. Examine the results  *;
*     and log.                                            *;
```

```
*  6) Move to Step 1. Change the value in the %LET       *;
*     statement to 9 and add the NOPRINT option in the    *;
*     PROC SQL statement to finalize the program. Run the *;
*     entire program. Examine the results and log.       *;
************************************************************;


************************************************;
* Step 1: Create Macro Variables              *;
************************************************;
%let Division=3;


proc sql;
select Name
   into :StateList SEPARATED BY ","
   from sq.statepopulation
   where Division = "&Division";
quit;
%put &=Division;
%put &=StateList;
```

Name
IL
IN
MI
OH
WI

```
128         %put &=Division;
DIVISION=3
129         %put &=StateList;
STATELIST=IL,IN,MI,OH,WI

************************************************;
* Step 1: Create Macro Variables              *;
```

**********;

%let Division=3;

proc sql;

select quote(strip(Name))

into :StateList SEPARATED BY ","

from sq.statepopulation

where Division = "&Division";

quit;

%put &=Division;

%put &=StateList;

"IL"
"IN"
"MI"
"OH"
"WI"

```
84          %put &=Division;
DIVISION=3
85          %put &=StateList;
STATELIST="IL","IN","MI","OH","WI"
```

%let Division=3;

proc sql noprint;

select quote(strip(Name))

into :StateList SEPARATED BY ","

from sq.statepopulation

where Division = "&Division";

quit;

%put &=Division;

%put &=StateList;

```
*********************************************;
* Step 2: Use Macro Variables               *;
*********************************************;
options symbolgen;
proc sql;
create table division&Division as
select *
        from sq.customer
        where State in (&StateList);
quit;
options nosymbolgen;
```

```
87          options symbolgen;
88          proc sql;
89          create table division&Division as
SYMBOLGEN:  Macro variable DIVISION resolves to 3
90          select *
91          from sq.customer
92          where State in (&StateList);
SYMBOLGEN:  Macro variable STATELIST resolves to "IL","IN","MI","OH","WI"
NOTE: Table WORK.DIVISION3 created, with 16022 rows and 22 columns.
```

```
%let Division=9;


proc sql noprint;
select quote(strip(Name))
   into :StateList SEPARATED BY ","
   from sq.statepopulation
   where Division = "&Division";
quit;
%put &=Division;
%put &=StateList;
```

```
*************************************************;
* Step 2: Use Macro Variables              *;
*************************************************;
options symbolgen;
proc sql;
create table division&Division as
select *
        from sq.customer
        where State in (&StateList);
quit;
options nosymbolgen;
```

```
87          options symbolgen;
88          proc sql;
89          create table division&Division as
SYMBOLGEN:  Macro variable DIVISION resolves to 9
90          select *
91          from sq.customer
92          where State in (&StateList);
SYMBOLGEN:  Macro variable STATELIST resolves to "AK","CA","HI","OR","WA"
NOTE: Table WORK.DIVISION9 created, with 22296 rows and 22 columns.
```

```
%let Division=3;
proc sql noprint;
select Name format=$quote4.
   into :StateList SEPARATED BY ","
   from sq.statepopulation
   where Division = '&Division';
quit;


%put &=StateList;


*************************************************;
* Step 2: Use Macro Variables              *;
```

**********;

options symbolgen;

proc sql;

create table division&Division as

select *

from sq.customer

where State in (&StateList);

quit;

options nosymbolgen;

```
81         %put &=StateList;
STATELIST="IL","IN","MI","OH","WI"
82
83         ************************************************;
84         * Step 2: Use Macro Variables                  *;
85         ************************************************;
86         options symbolgen;
87         proc sql;
88         create table division&Division as
SYMBOLGEN:  Macro variable DIVISION resolves to 3
89         select *
90         from sq.customer
91         where State in (&StateList);
SYMBOLGEN:  Macro variable STATELIST resolves to "IL","IN","MI","OH","WI"
NOTE: Table WORK.DIVISION3 created, with 16022 rows and 22 columns.
```

/*Practice Level 1: Creating a Macro Variable from an SQL Query

If necessary, start SAS Studio before you begin.

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

Write a program that dynamically returns states in a specified region that

have a three-year estimated population change greater than the median population change for the entire region.

Using the sq.statepopulation table, write a query to create the macro variable MedianEst to store

the value of the median nPopChg3 of Region 1. Use the following requirements:

Calculate the median of nPopChg3.

Use the INTO clause to create a macro variable named MedianEst. Use the TRIMMED keyword.

Filter the results for rows in the Region column with the value of 1, which is character.

Add the NOPRINT option in the PROC SQL statement.

Below the query, view the new macro variable in the log using %PUT:%put &=MedianEst;

Run the query and view the log.

What is the value of MedianEst? Note: Type a numeric value.

*/

```
proc sql noprint;
select median(nPopChg3)
        into :MedianEst trimmed
        from sq.statepopulation
        where Region='1';
quit;
```

%put &=MedianEst;

```
96          %put &=MedianEst;
MEDIANEST=3341
```

/*Using the sq.statepopulation table, write another query to select states and their three-year population change

in Region 1 that have a higher estimate than the median of all states in that region. Use the following requirements:

Select the Name and nPopChg3 columns. Format nPopChg3 so that values are displayed with commas.

Filter rows for states in region 1 that have an nPopChg3 value greater than the macro variable &MedianEst created in step 1.

Order by nPopChg3 descending.

Add the following titles:

title "States in Region 1 with a 3-Year Estimated Population Change Greater than the Median";

title2 "Median Estimate: &MedianEst";

Run the query and view the results.

How many states are listed in the report? Note: Type a numeric value.

*/

```
title "States in Region 1 with a 3-Year Estimated Population Change Greater than the Median";
title2 "Median Estimate: &MedianEst";
proc sql;
select Name, nPopChg3 format=comma16.
        from sq.statepopulation
        where Region='1' and nPopChg3 > &MedianEst
        order by nPopChg3 desc;
quit;
```

States in Region 1 with a 3-Year Estimated Population Change Greater than the Median
Median Estimate: 3341

Name	nPopChg3
MA	38,903
NJ	19,977
PA	16,613
NH	6,691

/*At the beginning of your program, create a macro variable named RegionNum to specify the region to analyze.

Use the following requirements:

Use the %LET statement to create the macro variable RegionNum and set the value equal to 1.

In your program, replace every location of the character value 1 with &RegionNum.

Make sure that the macro variable is enclosed in double quotation marks.

Hint: You need to do this replacement in the WHERE clause in each query and in the TITLE statement.

Run the program and view the final results. These results should be the same as the last query you ran (four states).

In the %LET statement, replace the value 1 with 2.

Run the entire program.

How many states in Region 2 have a higher three-year estimated population change than the median of all states in that region?

Note: Type a numeric value.

*/

```
%let RegionNum=2;

proc sql noprint;

select median(nPopChg3)

        into :MedianEst trimmed

        from sq.statepopulation

        where Region="&RegionNum";

quit;


%put &=MedianEst;


title "States in Region &RegionNum with a 3-Year Estimated Population Change Greater than the
Median";

title2 "Median Estimate: &MedianEst";

proc sql;

select Name, nPopChg3 format=comma16.

        from sq.statepopulation

        where Region="&RegionNum" and nPopChg3 > &MedianEst

        order by nPopChg3 desc;

quit;
```

States in Region 2 with a 3-Year Estimated Population Change Greater than the Median
Median Estimate: 15174

Name	nPopChg3
MN	43,024
IN	31,796
OH	25,313
WI	21,517
MI	19,468
MO	17,840

```
/*Practice Level 2: Creating a Macro Variable with a List of Values from an SQL Query

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 write a program that dynamically determines a list of countries in a specified region.

Then determine the percentage of people who have borrowed from a financial institution or used a credit card (% age 15+),

and whether it is increasing or decreasing from the year 1 estimate to the year 3 estimate.

Using sq.globalmetadata, write a query to create a macro variable that lists the distinct values of CountryCode

for a specified region. Use the following requirements:

Create a macro variable named RegionValue with the value South Asia.

Use the RegionValue macro variable in a query to create a data-driven macro variable named Countries that

lists the country codes in the specified region. The value list should be comma separated, and

each value should be enclosed in quotation marks.

Use the NOPRINT option in the PROC SQL statement.

Use %PUT below your query to view the value of the macro variable Countries.

Run the query and %PUT statement, and then view the log.

What is the value of the macro variable Countries as shown in the log?

Note: Copy and paste the value from the log window or type the value as shown in the log.

*/

```
%let RegionValue=South Asia;

proc sql noprint;

select quote(strip(CountryCode))

        into :Countries separated by ","

        from sq.globalmetadata

        where Region="&RegionValue";

quit;

%put &=Countries;
```

/*Using the sq.globalfindex, write another query to select the countries in the region that you specified.

Convert the whole numbers to percentages, and determine whether the estimate value is increasing, decreasing, or unknown.

Use the following requirements:

1) Select the CountryCode and IndicatorName columns.

2) Create three new columns as follows:

First divide EstYear1 by 100 and name the column EstYear1PCT. Format using percentages.

Follow the previous step for EstYear3.

Use the CASE expression to determine whether the value is Increasing, Decreasing, or Unknown.

The value is Unknown when the value is null. Name the column Forecast.

Hint: In the CASE expression, make sure that the first WHEN tests whether a value is missing.

3) Filter rows by CountryCode using the Countries macro variable and whether IndicatorName is equal to

Borrowed from a financial institution or used a credit card (% age 15+).

4) Order by Forecast.

5) Add the title Countries in &RegionValue.

6) Change the RegionValue macro variable at the beginning of the program to North America.

7) Run the query and view the results.

What is the Forecast value when CountryCode is CAN?

*/

```
%let RegionValue=North America;

proc sql noprint;

select quote(strip(CountryCode))

        into :Countries separated by ","

        from sq.globalmetadata

        where Region="&RegionValue";

quit;

%put &=Countries;
```

```
80          %put &=Countries;
COUNTRIES="AFG","BGD","BTN","IND","LKA","NPL","PAK"
```

```
title "Countries in &RegionValue";
proc sql;
select CountryCode, IndicatorName,
        (EstYear1 / 100) as EstYear1PCT format=percent7.2,
        (EstYear3 / 100) as EstYear3PCT format=percent7.2,
        case when EstYear1 is null or EstYear3 is null then "Unknown"
                when calculated EstYear1PCT < calculated EstYear3PCT then "Increasing"
                when calculated EstYear1PCT > calculated EstYear3PCT then "Decreasing"
                else "No Change"
        end as Forecast
    from sq.globalfindex
    where CountryCode IN (&Countries)
    and IndicatorName="Borrowed from a financial institution or used a credit card (% age 15+)"
    order by Forecast;
quit;
```

Countries in North America

CountryCode	IndicatorName	EstYear1PCT	EstYear3PCT	Forecast
CAN	Borrowed from a financial institution or used a credit card (% age 15+)	76.5%	82.8%	Increasing
USA	Borrowed from a financial institution or used a credit card (% age 15+)	64.6%	68.4%	Increasing