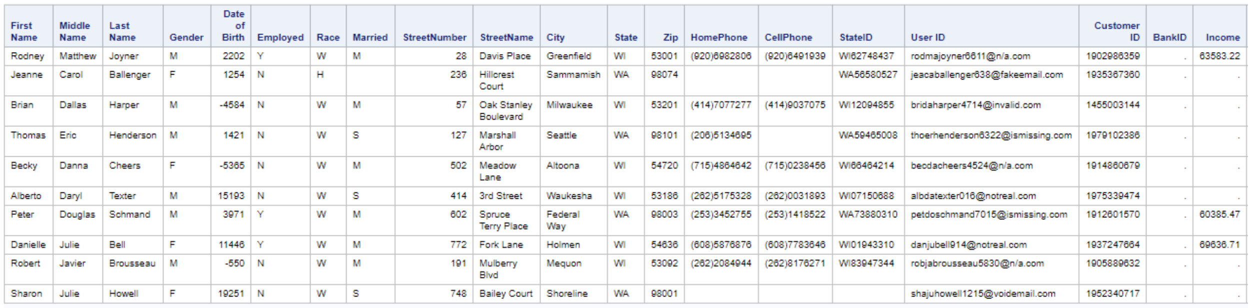
Advanced SAS Programmer

SAP1: SQL using SAS

W1 PROC SQL

W2 Generating Simple Reports, Summarizing and Grouping Data, Creating and Managing Tables, Using DICTIONARY Tables

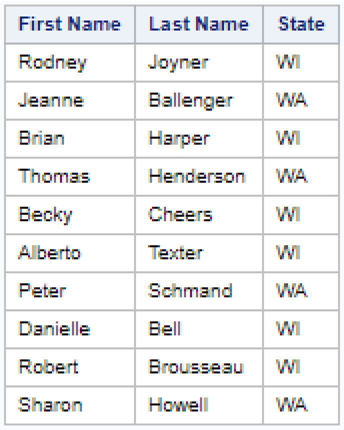
```
/* Defines the path to your data and assigns the libref. */

%let path=~/ESQ1M6;

libname sq "&path/data";


*******************************************************;
*  Activity 1.04                              *;
*  1) Remove the asterisk and select only the FirstName,  *;
*     LastName, and State columns. Run the query. View    *;
*     the log and results.                        *;
*  2) Remove the OBS=10 data set option and add the       *;
*     INOBS=10 PROC SQL option after the PROC SQL        *;
*     keywords and before the semicolon. Run the query.   *;
*     Are the results the same using the INOBS=10 option? *;
*     What about the log?                          *;
*  3) After the INOBS= option, add the NUMBER option. Run *;
*     the query. Which column was added to the results?   *;
*******************************************************;
proc sql;

select *

            from sq.customer(obs=10);

quit;
```



First Name	Middle Name	Last Name	Gender	Date of Birth	Employed	Race	Married	StreetNumber	StreetName	City	State	Zip	HomePhone	CellPhone	StateID	User ID	Customer ID	BankID	Income
Rodney	Matthew	Joyner	M	2202	Y	W	M	28	Davis Place	Greenfield	WI	53001	(920)6982806	(920)6491939	WI62748437	rodmajoyner6611@n/a.com	1902986359	.	63583.22
Jeanne	Carol	Ballenger	F	1254	N	H		236	Hillcrest Court	Sammamish	WA	98074			WA56580527	jeacaballenger638@fakeemail.com	1935367360	.	.
Brian	Dallas	Harper	M	-4584	N	W	M	57	Oak Stanley Boulevard	Milwaukee	WI	53201	(414)7077277	(414)9037075	WI12094855	bridaharper4714@invalid.com	1455003144	.	.
Thomas	Eric	Henderson	M	1421	N	W	S	127	Marshall Arbor	Seattle	WA	98101	(206)5134695		WA59465008	thoerhenderson6322@ismissing.com	1979102386	.	.
Becky	Danna	Cheers	F	-5365	N	W	M	502	Meadow Lane	Altoona	WI	54720	(715)4864642	(715)0238456	WI66464214	becdacheers4524@n/a.com	1914860679	.	.
Alberto	Daryl	Texter	M	15193	N	W	S	414	3rd Street	Waukesha	WI	53186	(262)5175328	(262)0031893	WI07150688	albdatexter016@notreal.com	1975339474	.	.
Peter	Douglas	Schmand	M	3971	Y	W	M	602	Spruce Terry Place	Federal Way	WA	98003	(253)3452755	(253)1418522	WA73880310	petdoschmand7015@ismissing.com	1912601570	.	60385.47
Danielle	Julie	Bell	F	11446	Y	W	M	772	Fork Lane	Holmen	WI	54636	(608)5876876	(608)7783646	WI01943310	danjubell914@notreal.com	1937247664	.	69636.71
Robert	Javier	Brousseau	M	-550	N	W	M	191	Mulberry Blvd	Mequon	WI	53092	(262)2084944	(262)8176271	WI83947344	robjabrousseau5830@n/a.com	1905889632	.	.
Sharon	Julie	Howell	F	19251	N	W	S	748	Bailey Court	Shoreline	WA	98001				shajuhowell1215@voidemail.com	1952340717	.	.

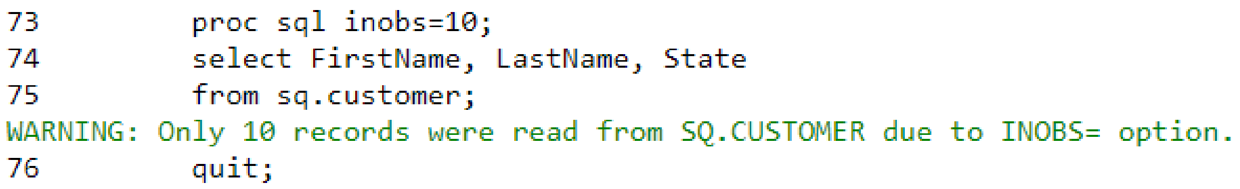
```
proc sql;

select FirstName, LastName, State
```

```
        from sq.customer(obs=10);

quit;
```

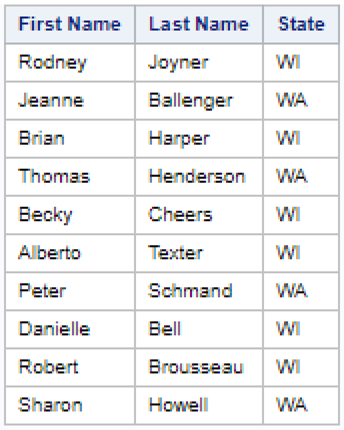


First Name	Last Name	State
Rodney	Joyner	WI
Jeanne	Ballenger	WA
Brian	Harper	WI
Thomas	Henderson	WA
Becky	Cheers	WI
Alberto	Texter	WI
Peter	Schmand	WA
Danielle	Bell	WI
Robert	Brousseau	WI
Sharon	Howell	WA

```
proc sql inobs=10;

select FirstName, LastName, State

        from sq.customer;

quit;
```

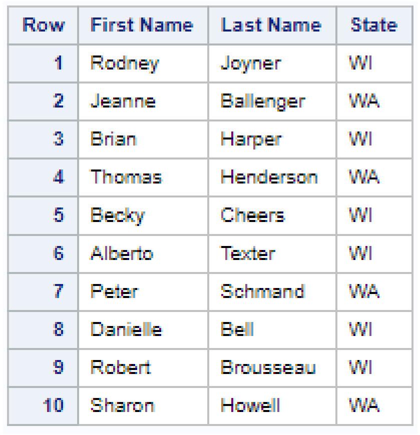


```
           proc sql inobs=10;
           select FirstName, LastName, State
           from sq.customer;
WARNING: Only 10 records were read from SQ.CUSTOMER due to INOBS= option.
           quit;
```



First Name	Last Name	State
Rodney	Joyner	WI
Jeanne	Ballenger	WA
Brian	Harper	WI
Thomas	Henderson	WA
Becky	Cheers	WI
Alberto	Texter	WI
Peter	Schmand	WA
Danielle	Bell	WI
Robert	Brousseau	WI
Sharon	Howell	WA

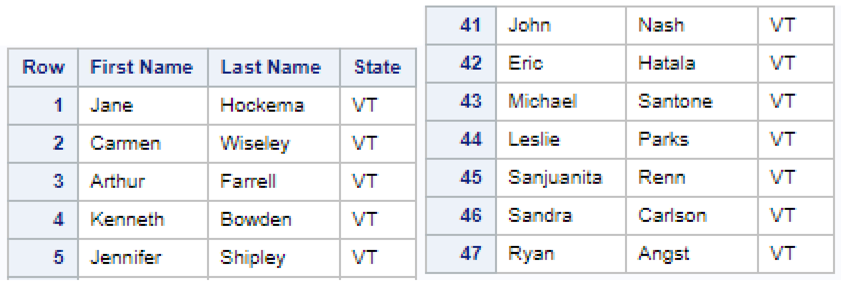
```
proc sql inobs=10 Number;

select FirstName, LastName, State

        from sq.customer;

quit;
```



Row	First Name	Last Name	State
1	Rodney	Joyner	WI
2	Jeanne	Ballenger	WA
3	Brian	Harper	WI
4	Thomas	Henderson	WA
5	Becky	Cheers	WI
6	Alberto	Texter	WI
7	Peter	Schmand	WA
8	Danielle	Bell	WI
9	Robert	Brousseau	WI
10	Sharon	Howell	WA

W2-1 Generating Simple Reports

```
*****************************************************;
*  Activity 2.01                              *;
*  1) Complete the WHERE clause to filter for customers   *;
*     in the state of VT and run the query.            *;
*  2) Add another expression using the OR operator to     *;
*     select only customers from the state of VT or SC.   *;
*     How many customers are from either VT or SC?        *;
*  3) Switch your current expression to use the IN        *;
*     operator. Add the state of GA. How many customers   *;
*     are from either VT, SC, or GA?                  *;
*****************************************************;


proc sql number;
select FirstName, LastName, State
  from sq.customer
  where State = 'VT';
quit;
```



Row	First Name	Last Name	State
1	Jane	Hockema	VT
2	Carmen	Wiseley	VT
3	Arthur	Farrell	VT
4	Kenneth	Bowden	VT
5	Jennifer	Shipley	VT
41	John	Nash	VT
42	Eric	Hatala	VT
43	Michael	Santone	VT
44	Leslie	Parks	VT
45	Sanjuanita	Renn	VT
46	Sandra	Carlson	VT
47	Ryan	Angst	VT

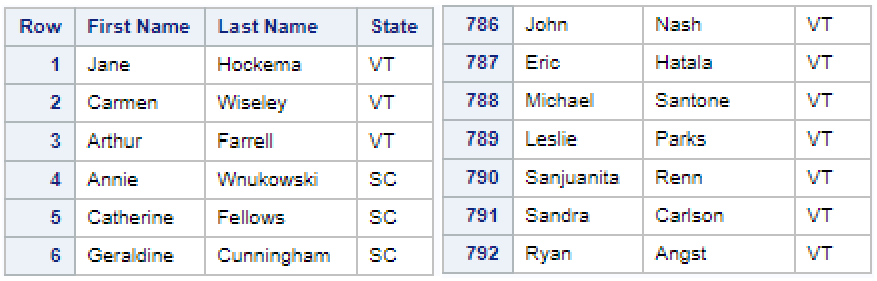
proc sql number;

select FirstName, LastName, State

  from sq.customer

  where State = 'VT' or State = 'SC';

quit;



Row	First Name	Last Name	State
1	Jane	Hockema	VT
2	Carmen	Wiseley	VT
3	Arthur	Farrell	VT
4	Annie	Wnukowski	SC
5	Catherine	Fellows	SC
6	Geraldine	Cunningham	SC
786	John	Nash	VT
787	Eric	Hatala	VT
788	Michael	Santone	VT
789	Leslie	Parks	VT
790	Sanjuanita	Renn	VT
791	Sandra	Carlson	VT
792	Ryan	Angst	VT

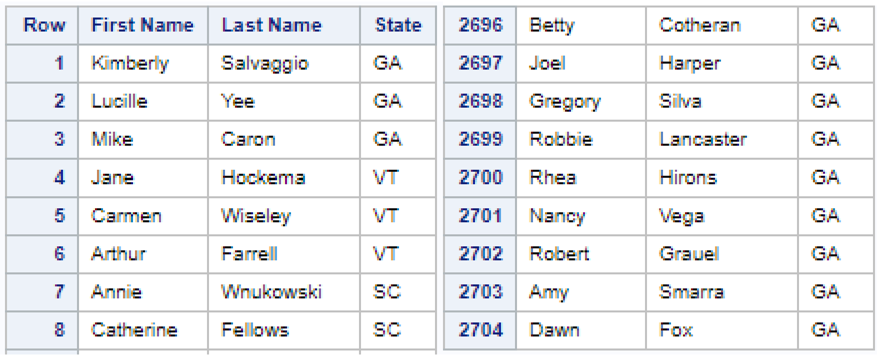
proc sql number;

select FirstName, LastName, State

  from sq.customer

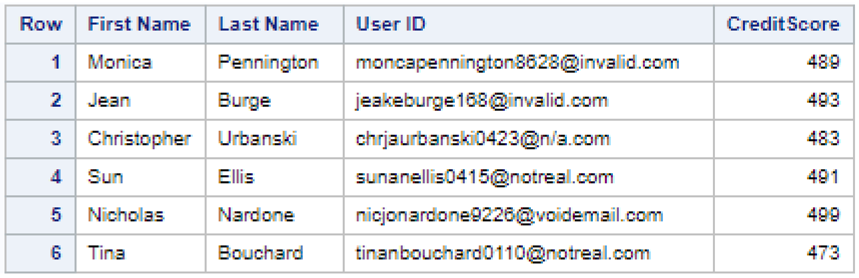
  where State IN ('VT', 'SC', 'GA');

quit;



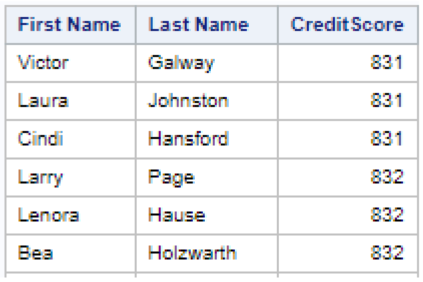
Row	First Name	Last Name	State
1	Kimberly	Salvaggio	GA
2	Lucille	Yee	GA
3	Mike	Caron	GA
4	Jane	Hockema	VT
5	Carmen	Wiseley	VT
6	Arthur	Farrell	VT
7	Annie	Wnukowski	SC
8	Catherine	Fellows	SC
2696	Betty	Cotheran	GA
2697	Joel	Harper	GA
2698	Gregory	Silva	GA
2699	Robbie	Lancaster	GA
2700	Rhea	Hirons	GA
2701	Nancy	Vega	GA
2702	Robert	Grauel	GA
2703	Amy	Smarra	GA
2704	Dawn	Fox	GA

```
*****************************************************;
*  Activity 2.02                              *;
*  1) Examine the query. Add a WHERE clause to find all   *;
*     customers with a CreditScore value that is less     *;
*     than 500 and run the query. What do you notice      *;
*     about the values in the CreditScore column? How     *;
*     many rows are in your report?                       *;
*  2) Include the AND operator in the WHERE clause to     *;
*     find all rows that are less than 500 and not null.  *;
*     Use a method of your choice. How many rows are in   *;
*     your final report?                                  *;
*****************************************************;


proc sql number;
select FirstName, LastName, UserID, CreditScore
        from sq.customer
        where CreditScore < 500 and CreditScore is not null;
quit;
```



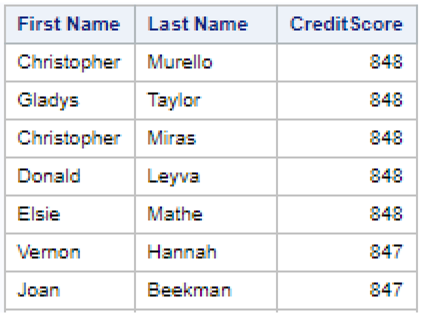
Row	First Name	Last Name	User ID	CreditScore
1	Monica	Pennington	moncapennington8628@invalid.com	489
2	Jean	Burge	jeakeburge168@invalid.com	493
3	Christopher	Urbanski	chrjaurbanski0423@n/a.com	483
4	Sun	Ellis	sunanellis0415@notreal.com	491
5	Nicholas	Nardone	nicjonardone9226@voidemail.com	499
6	Tina	Bouchard	tinanbouchard0110@notreal.com	473

```
*****************************************************;
*  Activity 2.03                              *;
*  1) Complete the ORDER BY clause and sort by            *;
*     CreditScore. Run the query and examine the report.  *;
*     What is the default sort order?                     *;
*  2) Add the keyword DESC after the CreditScore column   *;
*    in the ORDER BY clause. Run the query and examine   *;
*    the report. What does the DESC option do?          *;
*  3) Add a secondary sort column to sort by LastName.   *;
*    Run the query. Who is the first customer on the     *;
*    report?                                   *;
*  4) Remove LastName from the SELECT clause and rerun    *;
*    the query. Are the results still sorted by LastName *;
*    within CreditScore?                        *;
*******************************************************;

proc sql;
select FirstName, LastName, CreditScore
        from sq.customer
        where CreditScore > 830
        order by CreditScore;
quit;
```



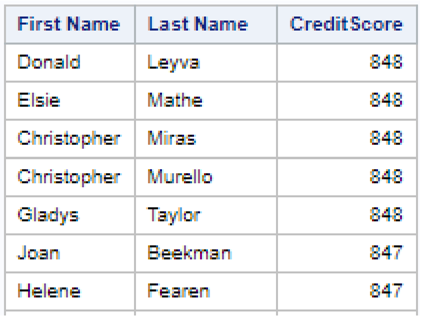
First Name	Last Name	CreditScore
Victor	Galway	831
Laura	Johnston	831
Cindi	Hansford	831
Larry	Page	832
Lenora	Hause	832
Bea	Holzwarth	832

```
proc sql;
select FirstName, LastName, CreditScore
        from sq.customer
        where CreditScore > 830
        order by CreditScore DESC;
quit;
```



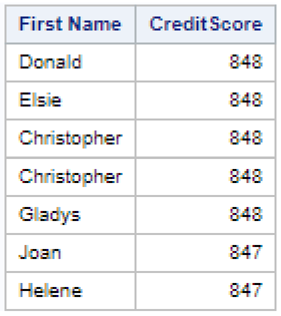
First Name	Last Name	CreditScore
Christopher	Murello	848
Gladys	Taylor	848
Christopher	Miras	848
Donald	Leyva	848
Elsie	Mathe	848
Vernon	Hannah	847
Joan	Beekman	847

```
proc sql;
select FirstName, LastName, CreditScore
        from sq.customer
        where CreditScore > 830
        order by CreditScore DESC, LastName;
quit;
```



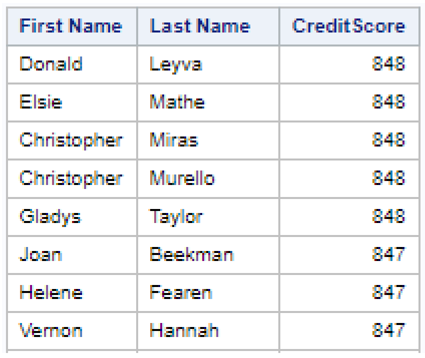
First Name	Last Name	CreditScore
Donald	Leyva	848
Elsie	Mathe	848
Christopher	Miras	848
Christopher	Murello	848
Gladys	Taylor	848
Joan	Beekman	847
Helene	Fearen	847

```
proc sql;
select FirstName, CreditScore
        from sq.customer
        where CreditScore > 830
        order by CreditScore DESC, LastName;
quit;
```



First Name	CreditScore
Donald	848
Elsie	848
Christopher	848
Christopher	848
Gladys	848
Joan	847
Helene	847

```
proc sql;
select FirstName, LastName, CreditScore
        from sq.customer
        where CreditScore > 830
        order by 3 DESC, 2;
quit;
```

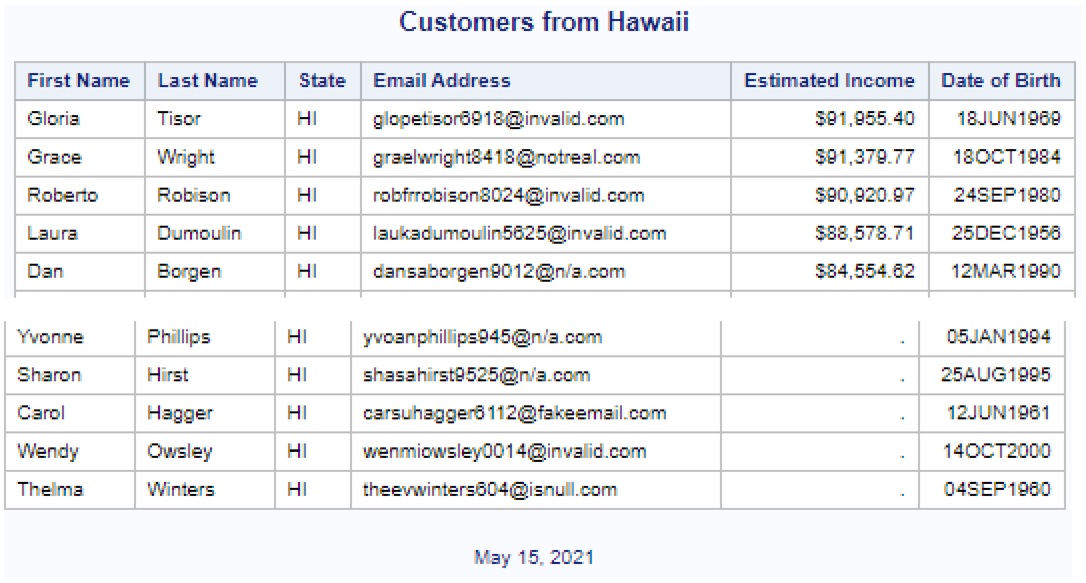


First Name	Last Name	CreditScore
Donald	Leyva	848
Elsie	Mathe	848
Christopher	Miras	848
Christopher	Murello	848
Gladys	Taylor	848
Joan	Beekman	847
Helene	Fearen	847
Vernon	Hannah	847

/* Enhancing Reports */

```
*******************************************************;
*  Activity 2.04                              *;
*  1) Examine the query. Add the title "Customers from    *;
*     Hawaii" and a footnote using today's date. Run the  *;
*     program and examine the new title and footnote in   *;
*     your report.                              *;
*  2) Apply LABEL="Email Address" to the UserID column    *;
*     and LABEL="Estimated Income" to the Income column.  *;
*  3) Apply FORMAT=DATE9. to the DOB column and           *;
*     FORMAT=DOLLAR16.2 to the Income column. Run the     *;
*     program and examine the report.               *;
*  4) Change the DOLLAR16.2 format to DOLLAR7.2. Run the  *;
*     program. What happens to the values in the Income   *;
*     column?                                   *;
*******************************************************;
```

```
/*Add a title*/

title "Customers from Hawaii";

/*Add a footnote*/

footnote "May 15, 2021";


proc sql;

select FirstName, LastName, State,

     UserID label="Email Address",

     Income "Estimated Income" format dollar16.2,

     DOB format date9.

          from sq.customer

          where State = "HI" and

       BankID is not null

          order by Income desc;

quit;


title; /*Clear title*/

footnote; /*Clear footnote*/
```



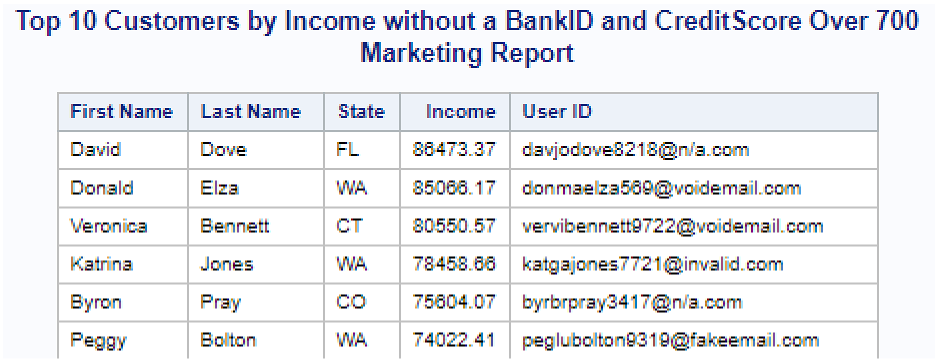
Customers from Hawaii

First Name	Last Name	State	Email Address	Estimated Income	Date of Birth
Gloria	Tisor	HI	glopetisor6918@invalid.com	$91,955.40	18JUN1969
Grace	Wright	HI	graelwright8418@notreal.com	$91,379.77	18OCT1984
Roberto	Robison	HI	robfrrobison8024@invalid.com	$90,920.97	24SEP1980
Laura	Dumoulin	HI	laukadumoulin5625@invalid.com	$88,578.71	25DEC1956
Dan	Borgen	HI	dansaborgen9012@n/a.com	$84,554.62	12MAR1990
Yvonne	Phillips	HI	yvoanphillips945@n/a.com	.	05JAN1994
Sharon	Hirst	HI	shasahirst9525@n/a.com	.	25AUG1995
Carol	Hagger	HI	carsuhagger6112@fakeemail.com	.	12JUN1961
Wendy	Owsley	HI	wenmiowsley0014@invalid.com	.	14OCT2000
Thelma	Winters	HI	theevwinters604@isnull.com	.	04SEP1960

May 15, 2021

```
*******************************************************;
*  Creating Simple Reports                             *;
*******************************************************;
*  Syntax                                              *;
*                                                      *;
*  TITLE<n> 'title-text';                              *;
*  PROC SQL OUTOBS=n;                                  *;
*  SELECT col-name <FORMAT=formatw.d> <LABEL='LABEL'>  *;
*     FROM input-table(OBS=n)                          *;
*     WHERE expression                                 *;
*     ORDER BY col-name <DESC>;                        *;
*  QUIT;                                               *;
*                                                      *;
*  TITLE;                                              *;
*******************************************************;

*******************************************************;
*  Demo                                                *;
*  1) Open the s102d01.sas program in the demos folder *;
*     and find the Demo section. Move to Report 1.     *;
*     Complete the query.                              *;
*     a) Complete the WHERE clause to filter for a     *;
*        missing BankID value and a value of CreditScore *;
*        greater than 700.                             *;
*     b) Complete the ORDER BY clause to arrange rows by *;
*        descending Income.                            *;
*     c) Add the column modifiers FORMAT=DOLLAR16. to the *;
*        Income column and LABEL='Email' to the UserID  *;
*        column. Remove the OBS= data set option and add *;
*      the OUTOBS=10 option in the PROC SQL statement.  *;
*  2) Move to Report 2. Complete the query.             *;
*    a) Complete the WHERE clause to filter DOB prior to *;
*      31DEC1940 and where Employed equals Y.          *;
*    b) Complete the ORDER BY clause to arrange rows by  *;
*      descending DOB. Run the query and view the       *;
*      results.                               *;
*    c) Add the column modifiers FORMAT= to the DOB and  *;
*      Zip columns. Remove the OBS= data set option and *;
*      highlight and run the query. Examine the log and *;
*      results.                               *;
*      Note: The Z format writes standard numeric data  *;
*           with leading 0s. Scroll in the results and *;
*           show ZIP codes with fewer than five digits.*;
**********************************************************;


*********************************;
*REPORT 1                     *;
* - No BankID                 *;
* - CreditScore > 700          *;
* - Top 10 customers by Income   *;
*********************************;
title "Top 10 Customers by Income without a BankID and CreditScore Over 700";
title2 "Marketing Report";
proc sql;
select FirstName, LastName, State,
      Income, UserID
   from sq.customer(obs=100)
   where BankID is null and CreditScore > 700
```

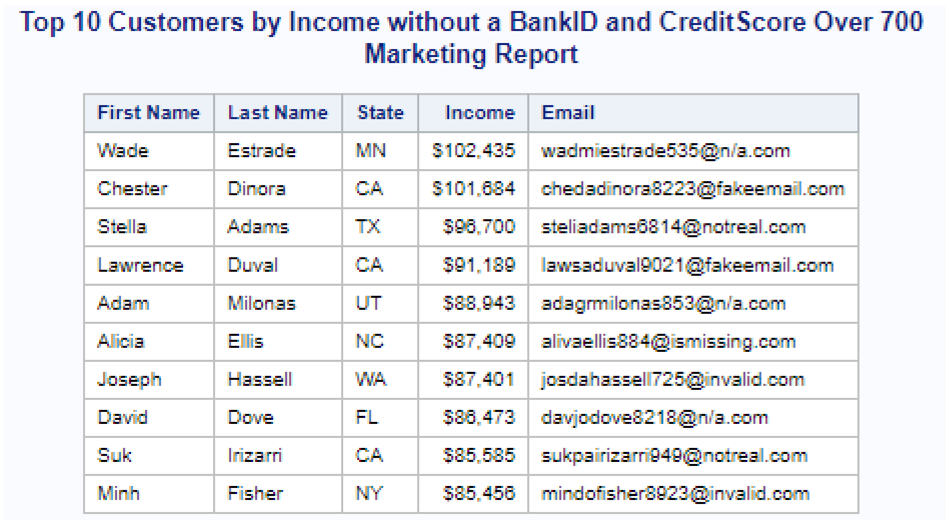
```
    order by Income Desc;
quit;
title;
```



Top 10 Customers by Income without a BankID and CreditScore Over 700
Marketing Report

First Name	Last Name	State	Income	User ID
David	Dove	FL	86473.37	davjodove8218@n/a.com
Donald	Elza	WA	85066.17	donmaelza569@voidemail.com
Veronica	Bennett	CT	80550.57	vervibennett9722@voidemail.com
Katrina	Jones	WA	78458.66	katgajones7721@invalid.com
Byron	Pray	CO	75604.07	byrbrpray3417@n/a.com
Peggy	Bolton	WA	74022.41	pegluboltonn9319@fakeemail.com

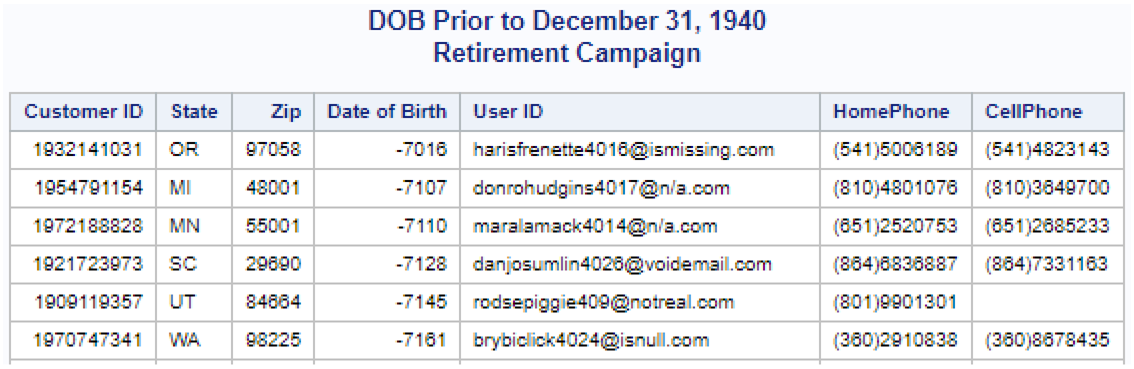
```
title "Top 10 Customers by Income without a BankID and CreditScore Over 700";
title2 "Marketing Report";
proc sql outobs=10;
select FirstName, LastName, State,
      Income format=dollar16., UserID "Email"
    from sq.customer
    where BankID is null and CreditScore > 700
    order by Income Desc;
quit;
title;
```



Top 10 Customers by Income without a BankID and CreditScore Over 700
Marketing Report

First Name	Last Name	State	Income	Email
Wade	Estrade	MN	$102,435	wadmiestrade535@n/a.com
Chester	Dinora	CA	$101,684	chedadinora8223@fakeemail.com
Stella	Adams	TX	$96,700	steliadams6814@notreal.com
Lawrence	Duval	CA	$91,189	lawsaduval9021@fakeemail.com
Adam	Milonas	UT	$88,943	adagrmilonas853@n/a.com
Alicia	Ellis	NC	$87,409	alivaellis884@ismissing.com
Joseph	Hassell	WA	$87,401	josdahassell725@invalid.com
David	Dove	FL	$86,473	davjodove8218@n/a.com
Suk	Irizarri	CA	$85,585	sukpairizarri949@notreal.com
Minh	Fisher	NY	$85,456	mindofisher8923@invalid.com

```
***********************************;
*REPORT 2                        *;
* - Born prior to December 31, 1940 *;
* - Employed                       *;
***********************************;
title "DOB Prior to December 31, 1940";
title2 "Retirement Campaign";
proc sql;
select CustomerID, State, Zip,
      DOB, UserID,
      HomePhone, CellPhone
   from sq.customer(obs=100)
   where DOB < '31DEC1940'd and Employed='Y'
         order by DOB DESC;
quit;
title;
```

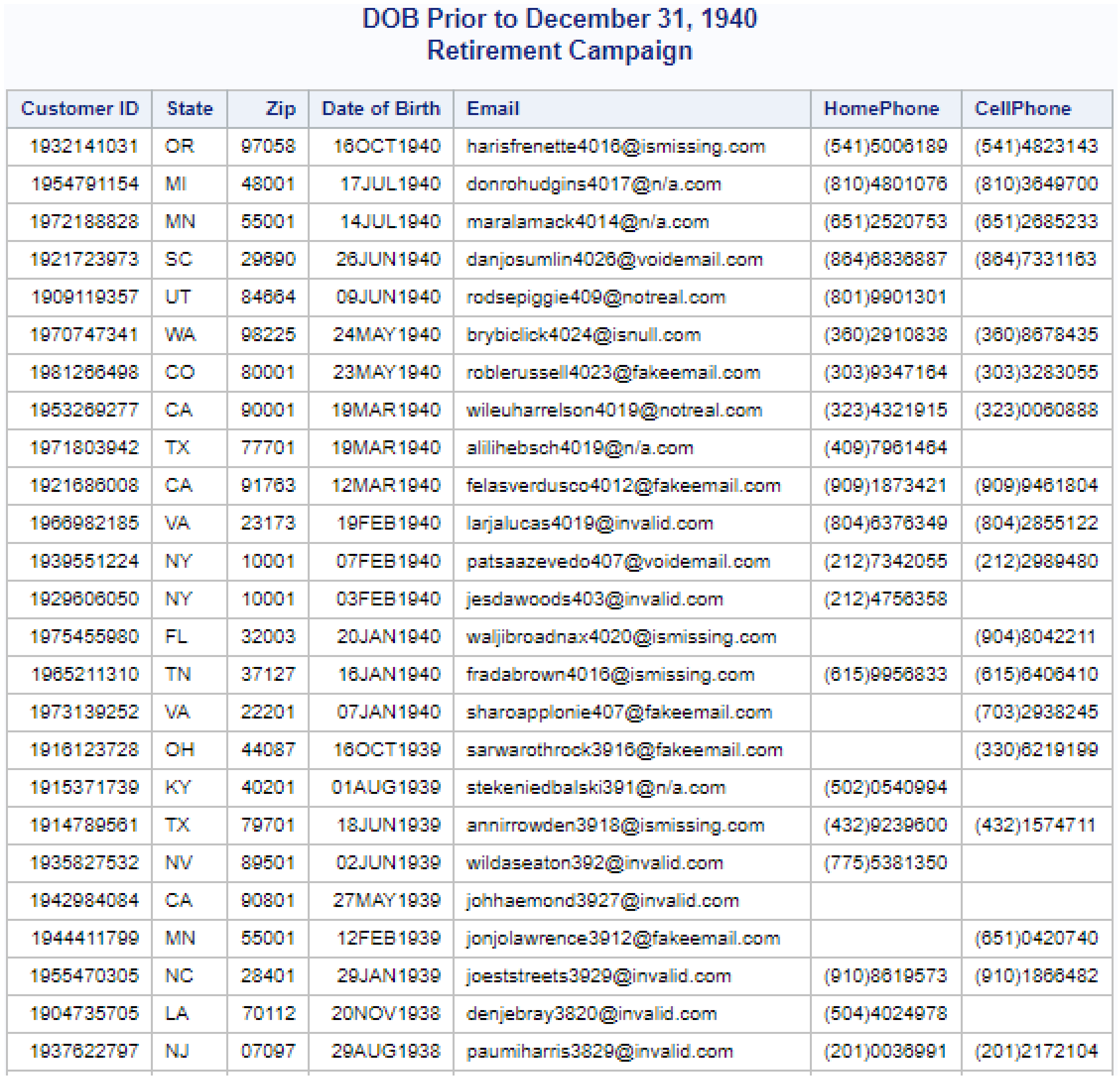


DOB Prior to December 31, 1940
Retirement Campaign

Customer ID	State	Zip	Date of Birth	User ID	HomePhone	CellPhone
1932141031	OR	97058	-7016	harisfrenette4016@ismissing.com	(541)5006189	(541)4823143
1954791154	MI	48001	-7107	donrohudgins4017@n/a.com	(810)4801076	(810)3649700
1972188828	MN	55001	-7110	maralamack4014@n/a.com	(651)2520753	(651)2685233
1921723973	SC	29690	-7128	danjosumlin4026@voidemail.com	(864)6836887	(864)7331163
1909119357	UT	84664	-7145	rodsepiggie409@notreal.com	(801)9901301	
1970747341	WA	98225	-7161	brybiclick4024@isnull.com	(360)2910838	(360)8678435

```
title "DOB Prior to December 31, 1940";
title2 "Retirement Campaign";
proc sql;
select CustomerID, State, Zip format=z5.,
      DOB format=date9., UserID 'Email',
      HomePhone, CellPhone
```

```
   from sq.customer(obs=100)
   where DOB < '31DEC1940'd and Employed='Y'
          order by DOB DESC;
quit;
title;
```

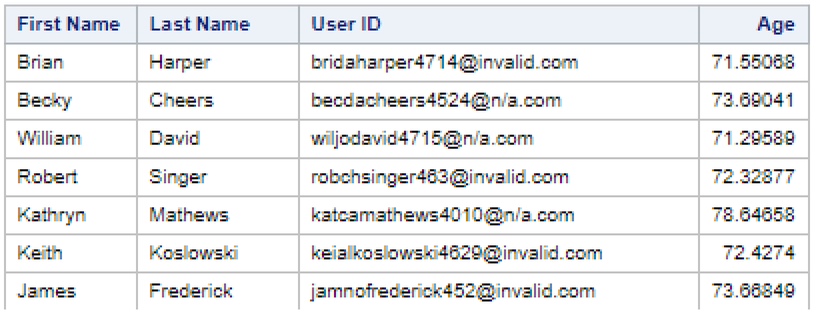


DOB Prior to December 31, 1940
Retirement Campaign

Customer ID	State	Zip	Date of Birth	Email	HomePhone	CellPhone
1932141031	OR	97058	16OCT1940	harisfrenette4016@ismissing.com	(541)5006189	(541)4823143
1954791154	MI	48001	17JUL1940	donrohudgins4017@n/a.com	(810)4801076	(810)3649700
1972188828	MN	55001	14JUL1940	maralamack4014@n/a.com	(651)2520753	(651)2685233
1921723973	SC	29690	26JUN1940	danjosumlin4026@voidemail.com	(864)6836887	(864)7331163
1909119357	UT	84664	09JUN1940	rodsepiggie409@notreal.com	(801)9901301	
1970747341	WA	98225	24MAY1940	brybiclick4024@isnull.com	(360)2910838	(360)8678435
1981266498	CO	80001	23MAY1940	roblerussell4023@fakeemail.com	(303)9347164	(303)3283055
1953269277	CA	90001	19MAR1940	wileuharrelson4019@notreal.com	(323)4321915	(323)0060888
1971803942	TX	77701	19MAR1940	alilihebsch4019@n/a.com	(409)7961464	
1921686008	CA	91763	12MAR1940	felasverdusco4012@fakeemail.com	(909)1873421	(909)9461804
1966982185	VA	23173	19FEB1940	larjalucas4019@invalid.com	(804)6376349	(804)2855122
1939551224	NY	10001	07FEB1940	patsaazevedo407@voidemail.com	(212)7342055	(212)2989480
1929606050	NY	10001	03FEB1940	jesdawoods403@invalid.com	(212)4756358	
1975455980	FL	32003	20JAN1940	waljibroadnax4020@ismissing.com		(904)8042211
1965211310	TN	37127	16JAN1940	fradabrown4016@ismissing.com	(615)9956833	(615)6406410
1973139252	VA	22201	07JAN1940	sharoapplonie407@fakeemail.com		(703)2938245
1916123728	OH	44087	16OCT1939	sarwarothrock3916@fakeemail.com		(330)6219199
1915371739	KY	40201	01AUG1939	stekeniedbalski391@n/a.com	(502)0540994	
1914789561	TX	79701	18JUN1939	annirrowden3918@ismissing.com	(432)9239600	(432)1574711
1935827532	NV	89501	02JUN1939	wildaseaton392@invalid.com	(775)5381350	
1942984084	CA	90801	27MAY1939	johhaemond3927@invalid.com		
1944411799	MN	55001	12FEB1939	jonjolawrence3912@fakeemail.com		(651)0420740
1955470305	NC	28401	29JAN1939	joeststreets3929@invalid.com	(910)8619573	(910)1866482
1904735705	LA	70112	20NOV1938	denjebray3820@invalid.com	(504)4024978	
1937622797	NJ	07097	29AUG1938	paumiharris3829@invalid.com	(201)0036991	(201)2172104

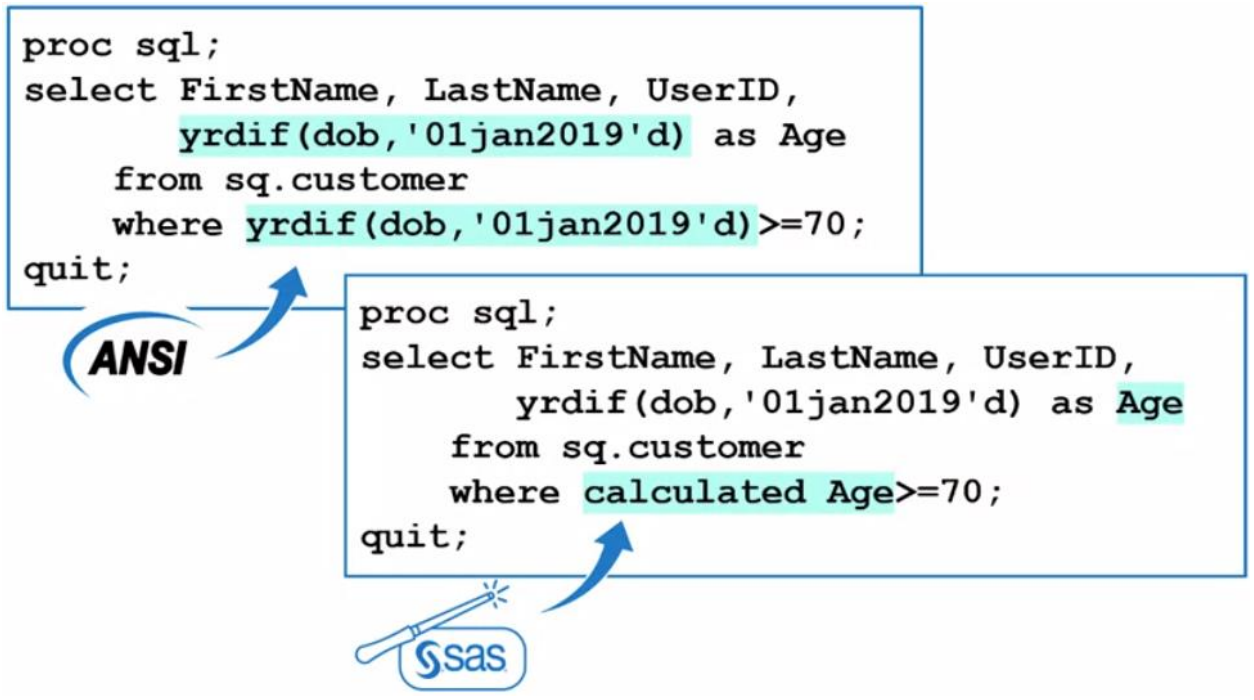
```
***********************************************************;
*  Activity 2.05                              *;
*  1) Examine and run the query. View the results.       *;
*  2) Add the expression yrdif(dob,'01jan2019'd) in the   *;
*     SELECT clause after UserID to create a new column.  *;
*    Run the query and examine the results. What is the  *;
*    name of the new column?                          *;
*  3) Add as Age after your function. Run the query and   *;
*    examine the results. What changes?                *;
*  4) Remove the OBS= data set option in the FROM clause  *;
*    and add a WHERE clause to return rows where Age is  *;
*    greater than or equal to 70. Run the query. Did the *;
*    query run successfully?                          *;
**********************************************************;
```

```
proc sql;
select FirstName, LastName, UserID,
           yrdif(dob, '01JAN2019'd) as Age
  from sq.customer
  where yrdif(dob, '01JAN2019'd) >= 70;
quit;
```



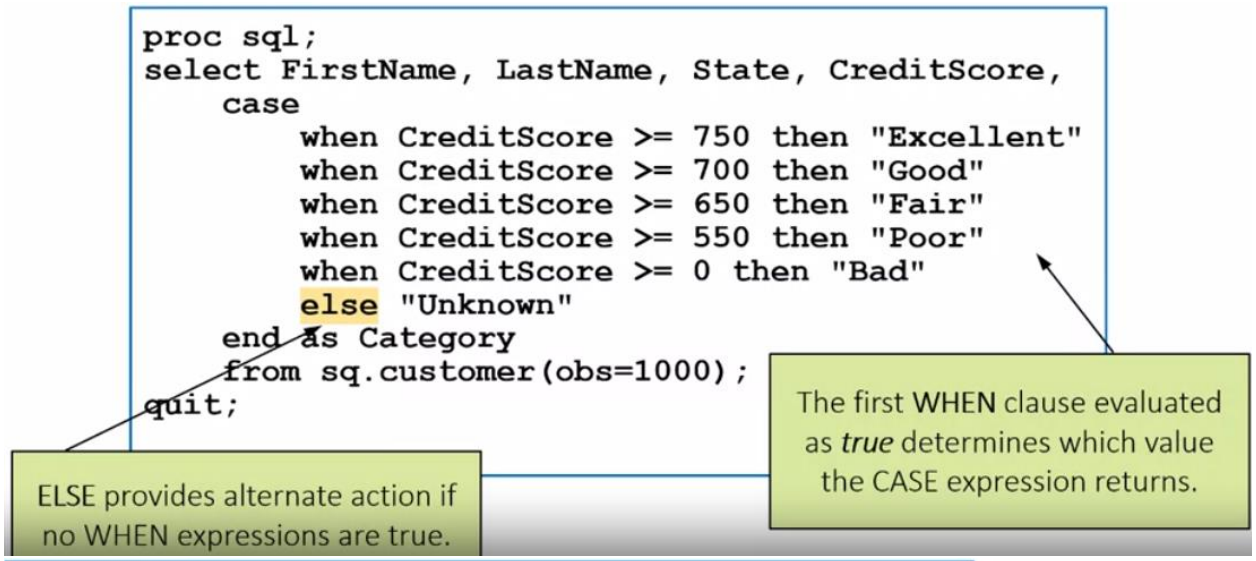
First Name	Last Name	User ID	Age
Brian	Harper	bridaharper4714@invalid.com	71.55068
Becky	Cheers	becdacheers4524@n/a.com	73.69041
William	David	wiljodavid4715@n/a.com	71.29589
Robert	Singer	robchsinger463@invalid.com	72.32877
Kathryn	Mathews	katcamathews4010@n/a.com	78.64658
Keith	Koslowski	keialkoslowski4629@invalid.com	72.4274
James	Frederick	jamnofrederick452@invalid.com	73.66849

```
proc sql;
select FirstName, LastName, UserID,
           yrdif(dob, '01JAN2019'd) as Age
  from sq.customer
  where calculated Age >= 70;
quit;
```



```
proc sql;
select FirstName, LastName, UserID,
       yrdif(dob,'01jan2019'd) as Age
    from sq.customer
    where yrdif(dob,'01jan2019'd)>=70;
quit;
```

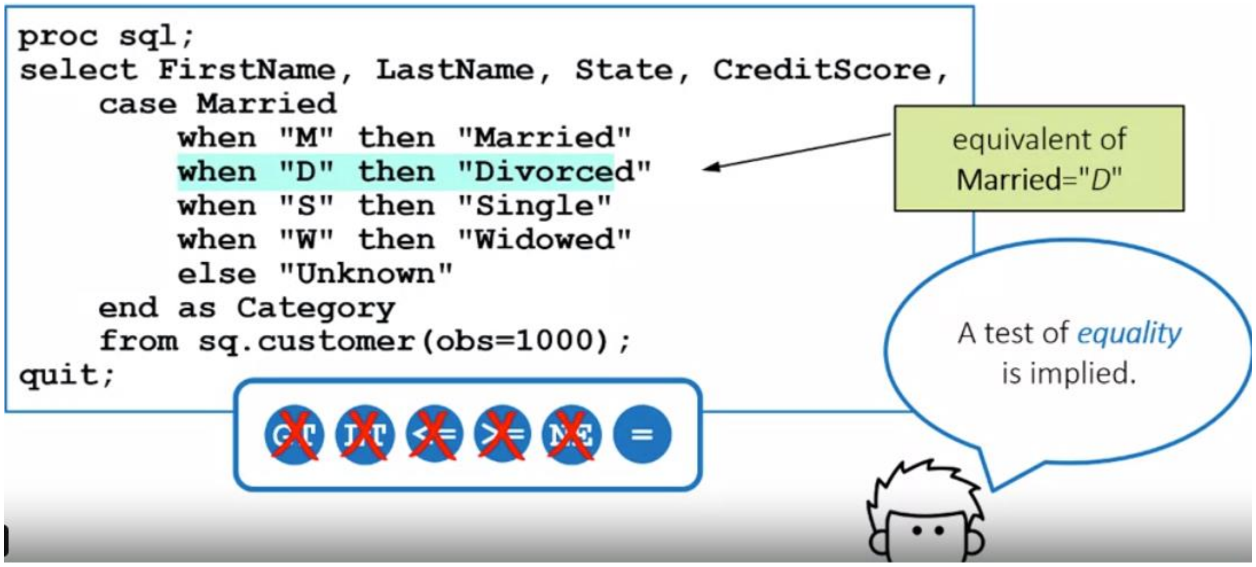
```
proc sql;
select FirstName, LastName, UserID,
       yrdif(dob,'01jan2019'd) as Age
    from sq.customer
    where calculated Age>=70;
quit;
```



```
proc sql;
select FirstName, LastName, State, CreditScore,
    case
        when CreditScore >= 750 then "Excellent"
        when CreditScore >= 700 then "Good"
        when CreditScore >= 650 then "Fair"
        when CreditScore >= 550 then "Poor"
        when CreditScore >= 0 then "Bad"
        else "Unknown"
    end as Category
    from sq.customer(obs=1000);
quit;
```

The first WHEN clause evaluated as *true* determines which value the CASE expression returns.

ELSE provides alternate action if no WHEN expressions are true.



```
proc sql;
select FirstName, LastName, State, CreditScore,
    case Married
        when "M" then "Married"
        when "D" then "Divorced"
        when "S" then "Single"
        when "W" then "Widowed"
        else "Unknown"
    end as Category
    from sq.customer(obs=1000);
quit;
```

equivalent of Married="*D*"

A test of *equality* is implied.

***;

* Assigning Values Conditionally   *;

***;

* Syntax   *;

*   *;

* PROC SQL;   *;

* SELECT col-name, <col-name>,   *;

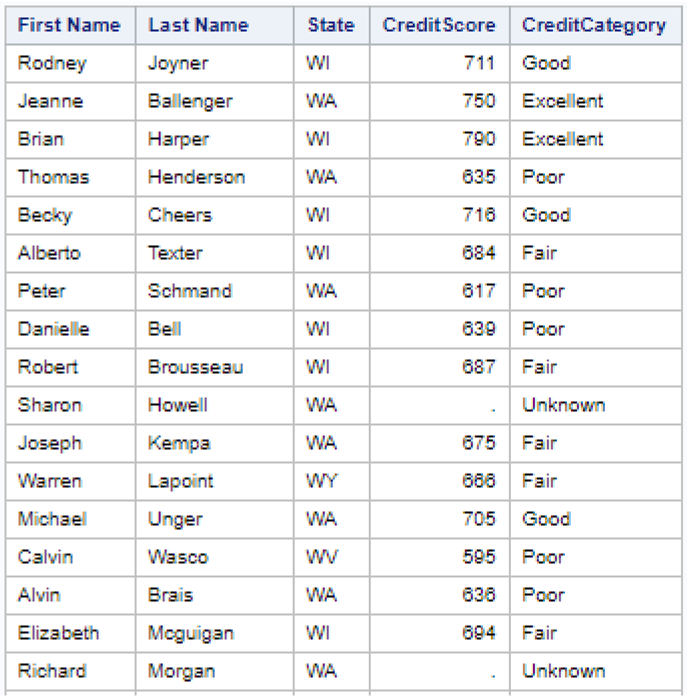
*   CASE <case-operand>   *;

*   WHEN condition THEN result-expression   *;

*   <WHEN condition THEN result-expression>   *;

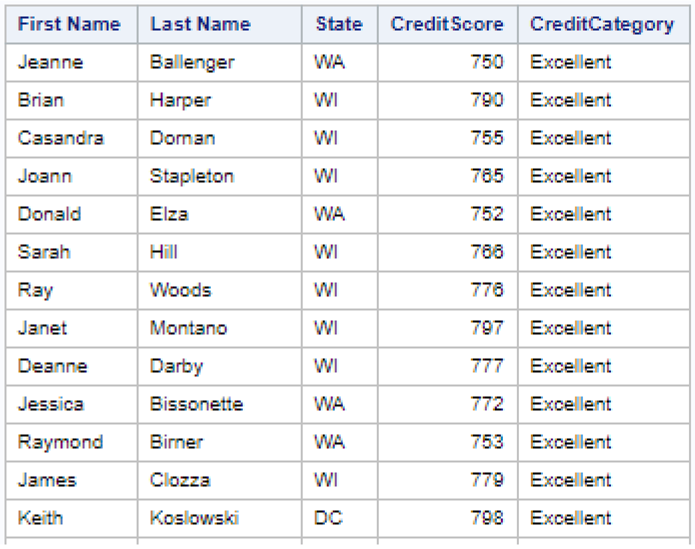
```
*          <ELSE result-expression>                  *;
*        END <AS column>                            *;
*     FROM input-table;                             *;
*  QUIT;                                            *;
*******************************************************;

*******************************************************;
*  Demo                                              *;
*  1) Open the s102d02.sas program in the demos folder    *;
*    and find the Demo section.                        *;
*  2) In the Simple Case Expression section:           *;
*    a) Highlight and run the query. Examine the log and *;
*       results.                                       *;
*    b) Complete the WHEN and ELSE expressions in the    *;
*       simple CASE expression. Highlight and run the    *;
*       query. Examine the log and results.             *;
*    c) Add the ELSE expression to change remaining      *;
*       values to Unknown. Highlight and run the query.  *;
*       Examine the log and results.                    *;
*    d) Add a WHERE clause to filter the table for       *;
*       customers with Excellent credit and remove the   *;
*       OBS=1000 data set option. Highlight and run the  *;
*       query. Examine the log and results.             *;
*  3) Move to the CASE-OPERAND FORM section.            *;
*    a) Highlight and run the query. Examine the log and *;
*       results.                                       *;
*    b) Complete the WHEN and ELSE expressions in the    *;
*       simple CASE-Operand expression. Highlight and    *;
*       run the query. Examine the log and results.      *;
```

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

****************************;
* SIMPLE CASE EXPRESSION    *;
****************************;
*Category        Range                  *;
*Excellent       750+                   *;
*Good            700 - 749              *;
*Fair            650 - 699              *;
*Poor            550 - 649              *;
*Bad             low - 549              *;
****************************;
proc sql;
select FirstName, LastName, State, CreditScore,
            case
              when CreditScore >=750 then "Excellent"
              when CreditScore >=700 then "Good"
              when CreditScore >=650 then "Fair"
              when CreditScore >=550 then "Poor"
              when CreditScore >=0 then "Bad"
              else "Unknown"
     end as CreditCategory
          from sq.customer(obs=1000);
quit;
```



First Name	Last Name	State	CreditScore	CreditCategory
Rodney	Joyner	WI	711	Good
Jeanne	Ballenger	WA	750	Excellent
Brian	Harper	WI	790	Excellent
Thomas	Henderson	WA	635	Poor
Becky	Cheers	WI	716	Good
Alberto	Texter	WI	684	Fair
Peter	Schmand	WA	617	Poor
Danielle	Bell	WI	639	Poor
Robert	Brousseau	WI	687	Fair
Sharon	Howell	WA	.	Unknown
Joseph	Kempa	WA	675	Fair
Warren	Lapoint	WY	666	Fair
Michael	Unger	WA	705	Good
Calvin	Wasco	WV	595	Poor
Alvin	Brais	WA	636	Poor
Elizabeth	Mcguigan	WI	694	Fair
Richard	Morgan	WA	.	Unknown

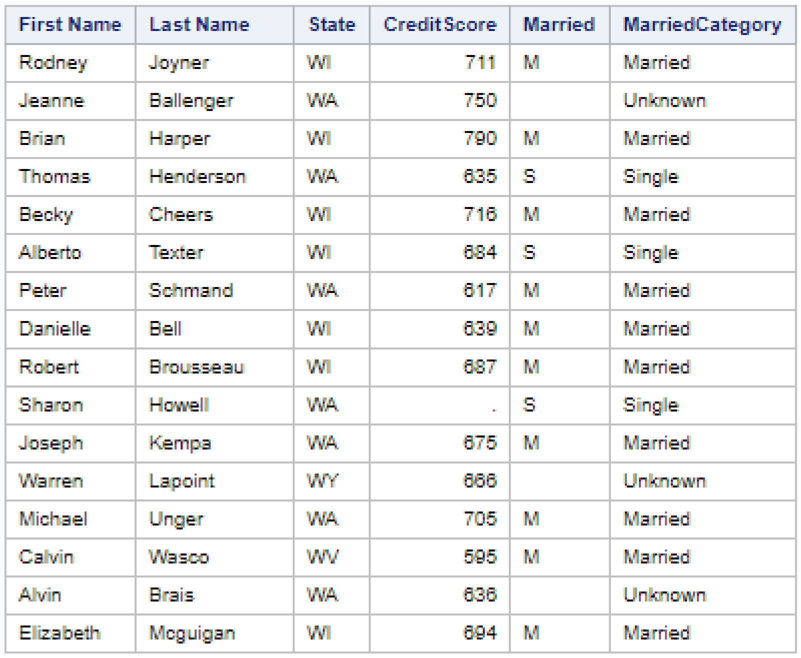
```
proc sql;
select FirstName, LastName, State, CreditScore,
            case
             when CreditScore >=750 then "Excellent"
             when CreditScore >=700 then "Good"
             when CreditScore >=650 then "Fair"
             when CreditScore >=550 then "Poor"
             when CreditScore >=0 then "Bad"
             else 'Unknown'
     end as CreditCategory
          from sq.customer
          where calculated CreditCategory = 'Excellent';
quit;
```



First Name	Last Name	State	CreditScore	CreditCategory
Jeanne	Ballenger	WA	750	Excellent
Brian	Harper	WI	790	Excellent
Casandra	Doman	WI	755	Excellent
Joann	Stapleton	WI	765	Excellent
Donald	Elza	WA	752	Excellent
Sarah	Hill	WI	766	Excellent
Ray	Woods	WI	776	Excellent
Janet	Montano	WI	797	Excellent
Deanne	Darby	WI	777	Excellent
Jessica	Bissonette	WA	772	Excellent
Raymond	Birner	WA	753	Excellent
James	Clozza	WI	779	Excellent
Keith	Koslowski	DC	798	Excellent

```
****************************;
* CASE-OPERAND FORM       *;
****************************;
* M = Married                        *;
* D = Divorced              *;
* S = Single                      *;
* W = Windowed          *;
* Else = Unknown                  *;
****************************;
proc sql;
select FirstName, LastName, State, CreditScore, Married,
    case Married
      when "M" then "Married"
      when "D" then "Divorced"
      when "S" then "Single"
      when "W" then "Widowed"
                  else 'Unknown'
    end as MarriedCategory
  from sq.customer(obs=1000);
```

quit;



First Name	Last Name	State	CreditScore	Married	MarriedCategory
Rodney	Joyner	WI	711	M	Married
Jeanne	Ballenger	WA	750		Unknown
Brian	Harper	WI	790	M	Married
Thomas	Henderson	WA	635	S	Single
Becky	Cheers	WI	716	M	Married
Alberto	Texter	WI	684	S	Single
Peter	Schmand	WA	617	M	Married
Danielle	Bell	WI	639	M	Married
Robert	Brousseau	WI	687	M	Married
Sharon	Howell	WA	.	S	Single
Joseph	Kempa	WA	675	M	Married
Warren	Lapoint	WY	666		Unknown
Michael	Unger	WA	705	M	Married
Calvin	Wasco	WV	595	M	Married
Alvin	Brais	WA	636		Unknown
Elizabeth	Mcguigan	WI	694	M	Married

/*Level 1 Practice: Querying a Table

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.transactionfull table contains customer and transaction information.

Using the sq.transactionfull table, write a PROC SQL step to generate a report for large transactions that

are not related to tuition payments to universities.

Use the following requirements as you generate the report:

Display the following columns in this order: CustomerName, MerchantName, Type, Service, and Amount from the sq.transactionfull table.

Select rows that have a transaction Amount value greater than $1,000 and a Service value not equal to University.

Order the rows such that the largest transaction is listed first.

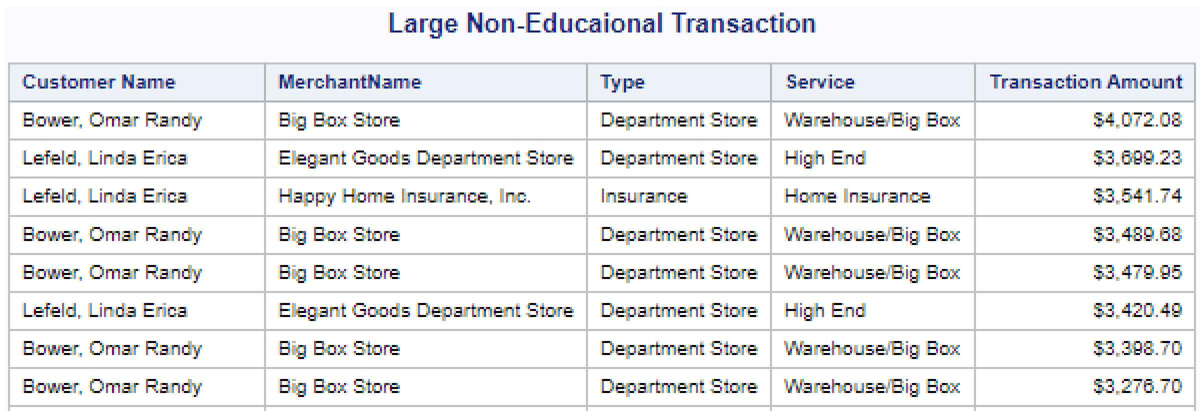
Format the Amount column with the DOLLAR10.2 format.

Label CustomerName as Customer Name, and Amount as Transaction Amount.

Add the title Large Non-Educational Transactions.

Run the program and view the results.

```
What is the first value for Transaction Amount?
*/

title "Large Non-Educaional Transaction";
proc sql;
Select CustomerName 'Customer Name',
        MerchantName, Type, Service,
        Amount 'Transaction Amount' format=dollar10.2
        from sq.transactionfull
        where Amount > 1000 and Service <> 'University'
        order by Amount desc;
quit;
title;

/*s102s01.sas*/

title 'Large Non-Educational Transactions';
proc sql;
select CustomerName label='Customer Name',
       MerchantName, Type, Service,
       Amount format=dollar10.2 label='Transaction Amount'
   from sq.transactionfull
   where Amount >1000 and
         Service ^= 'University'
   order by Amount desc;
quit;
title;
```


Large Non-Educaional Transaction

Customer Name	MerchantName	Type	Service	Transaction Amount
Bower, Omar Randy	Big Box Store	Department Store	Warehouse/Big Box	$4,072.08
Lefeld, Linda Erica	Elegant Goods Department Store	Department Store	High End	$3,699.23
Lefeld, Linda Erica	Happy Home Insurance, Inc.	Insurance	Home Insurance	$3,541.74
Bower, Omar Randy	Big Box Store	Department Store	Warehouse/Big Box	$3,489.68
Bower, Omar Randy	Big Box Store	Department Store	Warehouse/Big Box	$3,479.95
Lefeld, Linda Erica	Elegant Goods Department Store	Department Store	High End	$3,420.49
Bower, Omar Randy	Big Box Store	Department Store	Warehouse/Big Box	$3,398.70
Bower, Omar Randy	Big Box Store	Department Store	Warehouse/Big Box	$3,276.70

/*Level 2 Practice: Working with Datetime Values

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.transactionfull table contains a list of customer and transaction information.

Using the sq.transactionfull table, write a query to create a report displaying transactions that

took place in November and December of any year.

Use the following requirements as you generate the report:

Display the following columns in this order: CustomerName, MerchantName, and Amount.

Create a new column named TransactionDate by using the DATEPART function to extract the SAS date value from the DateTime column.

Format the new column using the DATE9. format. If necessary, you can review the DATEPART function here.

Filter the data to select rows where the month of the transaction date is November or December and

the Service value is not equal to University.

Order the report by the original DateTime column.

Format the Amount column with the DOLLAR10.2 format.

Label CustomerName as Customer Name, MerchantName as Merchant Name, Amount as Transaction Amount, and TransactionDate as Transaction Date.

Add the title November/December Transactions.

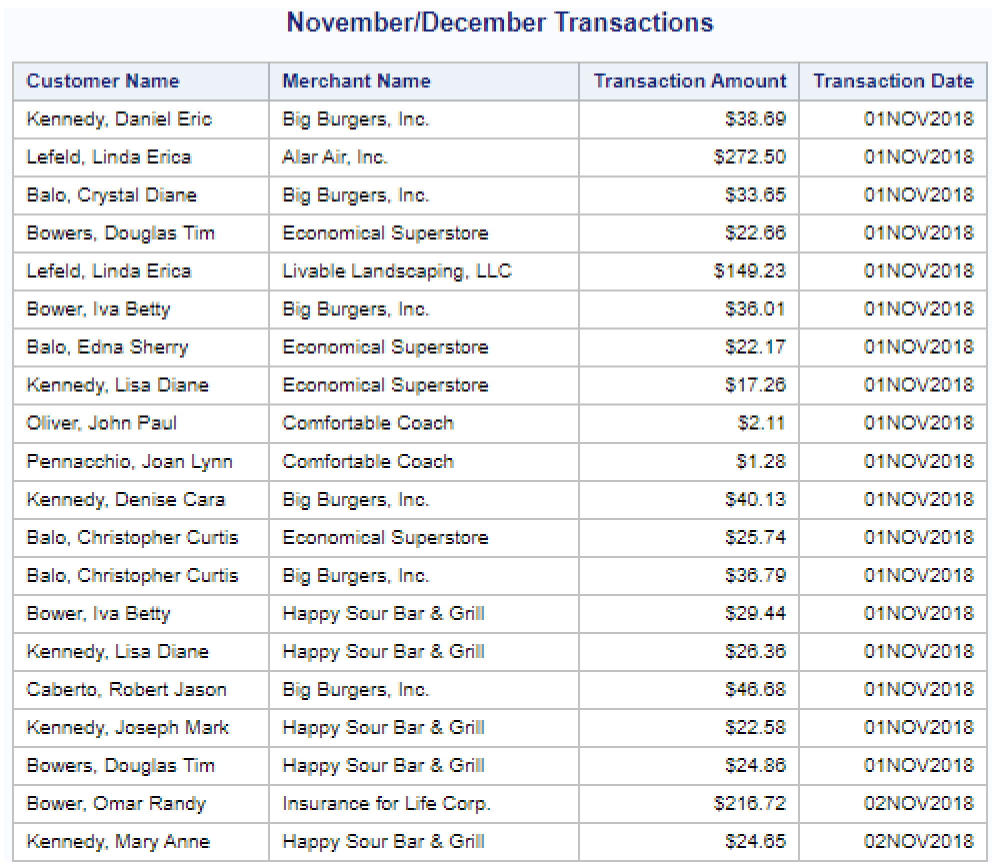
Run the program and view the results.

What value of MerchantName is on the first documented transaction in December?

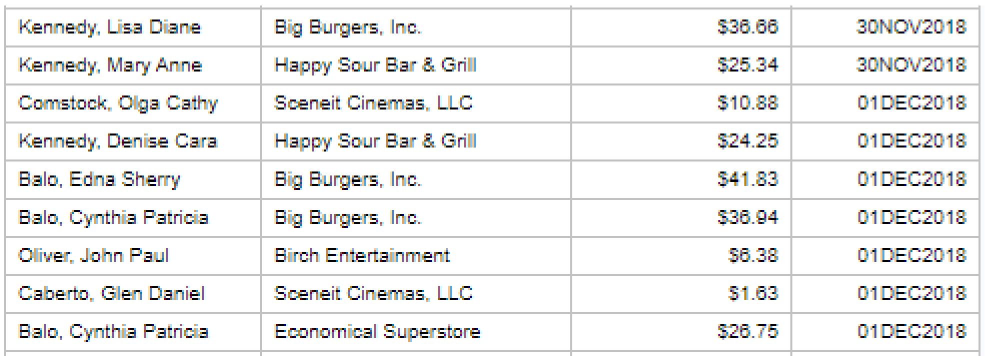
*/

title "November/December Transactions";

```
proc sql;
Select CustomerName 'Customer Name', MerchantName 'Merchant Name',
        Amount 'Transaction Amount' format=dollar10.2,
        datepart(DateTime) as TransactionDate label='Transaction Date' format=date9.
      from sq.transactionfull
      where month(calculated TransactionDate) IN (11, 12)
      and Service ^= 'University'
      order by DateTime;
quit;
title;
```


November/December Transactions

Customer Name	Merchant Name	Transaction Amount	Transaction Date
Kennedy, Daniel Eric	Big Burgers, Inc.	$38.69	01NOV2018
Lefeld, Linda Erica	Alar Air, Inc.	$272.50	01NOV2018
Balo, Crystal Diane	Big Burgers, Inc.	$33.65	01NOV2018
Bowers, Douglas Tim	Economical Superstore	$22.66	01NOV2018
Lefeld, Linda Erica	Livable Landscaping, LLC	$149.23	01NOV2018
Bower, Iva Betty	Big Burgers, Inc.	$36.01	01NOV2018
Balo, Edna Sherry	Economical Superstore	$22.17	01NOV2018
Kennedy, Lisa Diane	Economical Superstore	$17.26	01NOV2018
Oliver, John Paul	Comfortable Coach	$2.11	01NOV2018
Pennacchio, Joan Lynn	Comfortable Coach	$1.28	01NOV2018
Kennedy, Denise Cara	Big Burgers, Inc.	$40.13	01NOV2018
Balo, Christopher Curtis	Economical Superstore	$25.74	01NOV2018
Balo, Christopher Curtis	Big Burgers, Inc.	$36.79	01NOV2018
Bower, Iva Betty	Happy Sour Bar & Grill	$29.44	01NOV2018
Kennedy, Lisa Diane	Happy Sour Bar & Grill	$26.36	01NOV2018
Caberto, Robert Jason	Big Burgers, Inc.	$46.68	01NOV2018
Kennedy, Joseph Mark	Happy Sour Bar & Grill	$22.58	01NOV2018
Bowers, Douglas Tim	Happy Sour Bar & Grill	$24.86	01NOV2018
Bower, Omar Randy	Insurance for Life Corp.	$216.72	02NOV2018
Kennedy, Mary Anne	Happy Sour Bar & Grill	$24.65	02NOV2018



Kennedy, Lisa Diane	Big Burgers, Inc.	$36.66	30NOV2018
Kennedy, Mary Anne	Happy Sour Bar & Grill	$25.34	30NOV2018
Comstock, Olga Cathy	Sceneit Cinemas, LLC	$10.88	01DEC2018
Kennedy, Denise Cara	Happy Sour Bar & Grill	$24.25	01DEC2018
Balo, Edna Sherry	Big Burgers, Inc.	$41.83	01DEC2018
Balo, Cynthia Patricia	Big Burgers, Inc.	$36.94	01DEC2018
Oliver, John Paul	Birch Entertainment	$6.38	01DEC2018
Caberto, Glen Daniel	Sceneit Cinemas, LLC	$1.63	01DEC2018
Balo, Cynthia Patricia	Economical Superstore	$26.75	01DEC2018

Summarizing and Grouping Data

```
***********************************************************;
*  Activity 2.06                                *;
*  1) Examine and run the query. View the results.       *;
*  2) Change the State column in the SELECT clause to the *;
*     Employed column. Run the query. What does this     *;
*     query show?                                 *;
*  3) Add the Married column in the SELECT clause after   *;
*     the Employed column. Run the query. What does this  *;
*     query show?                                 *;
***********************************************************;


proc sql;
select distinct State
   from sq.customer;
quit;


proc sql;
select distinct Employed
   from sq.customer;
quit;
```

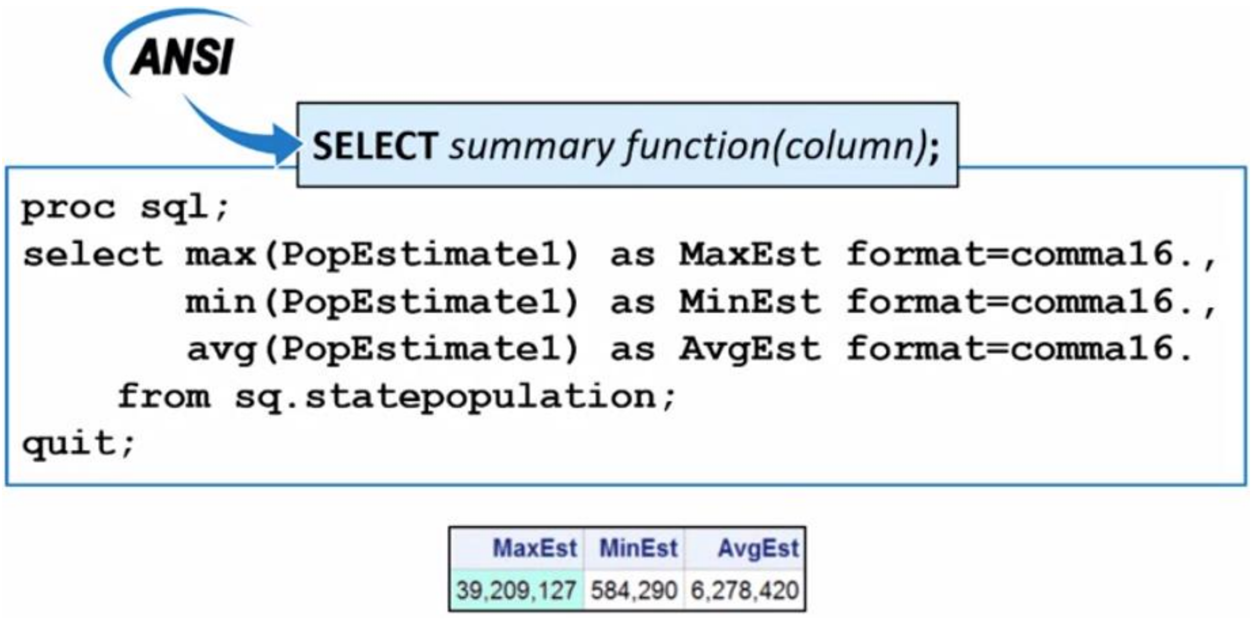
proc sql;

select distinct Employed, Married

from sq.customer;

quit;

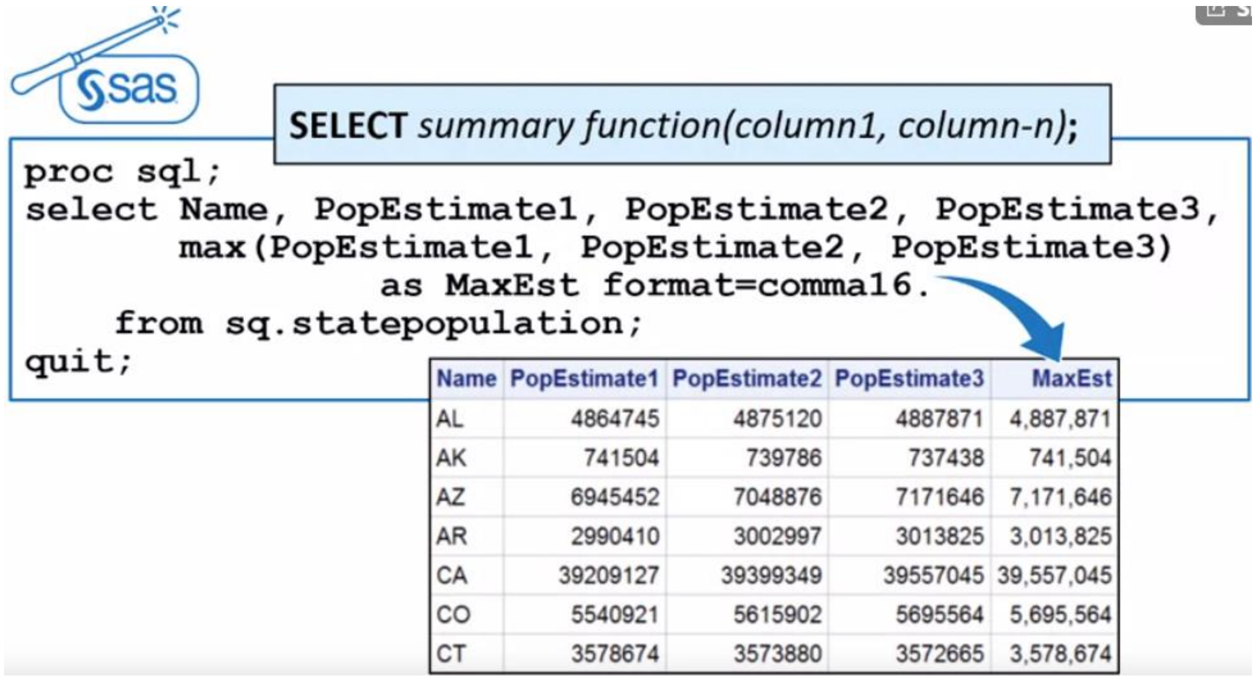
Employed	Married
N	
N	D
N	M
N	S
N	W
Y	
Y	D
Y	M
Y	S
Y	W



SELECT *summary function(column)*;

```
proc sql;
select max(PopEstimate1) as MaxEst format=comma16.,
       min(PopEstimate1) as MinEst format=comma16.,
       avg(PopEstimate1) as AvgEst format=comma16.
    from sq.statepopulation;
quit;
```

MaxEst	MinEst	AvgEst
39,209,127	584,290	6,278,420

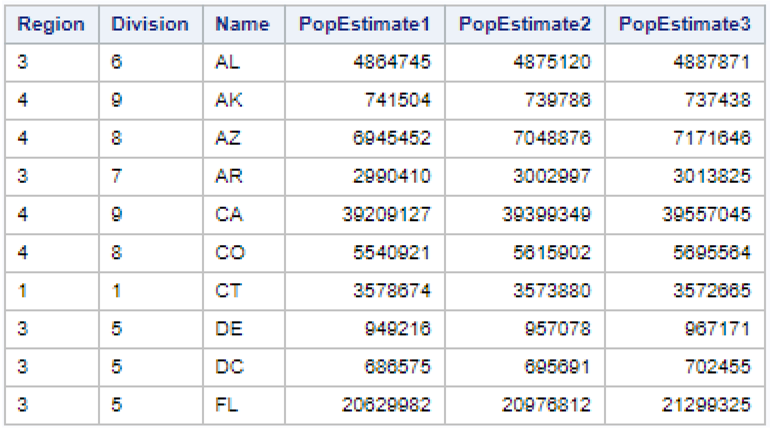


SELECT *summary function(column1, column-n)*;

```
proc sql;
select Name, PopEstimate1, PopEstimate2, PopEstimate3,
       max(PopEstimate1, PopEstimate2, PopEstimate3)
            as MaxEst format=comma16.
    from sq.statepopulation;
quit;
```

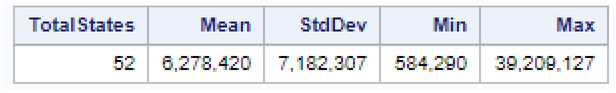
Name	PopEstimate1	PopEstimate2	PopEstimate3	MaxEst
AL	4864745	4875120	4887871	4,887,871
AK	741504	739786	737438	741,504
AZ	6945452	7048876	7171646	7,171,646
AR	2990410	3002997	3013825	3,013,825
CA	39209127	39399349	39557045	39,557,045
CO	5540921	5615902	5695564	5,695,564
CT	3578674	3573880	3572665	3,578,674

```
*********************************************************;
*  Summary Functions                        *;
*********************************************************;
*  Syntax                              *;
*                                      *;
*  SELECT summary function(column);                  *;
*  SELECT summary function(column1, column-n);          *;
*********************************************************;


*********************************************************;
*  Demo                                *;
*********************************************************;
*  1) Open the s102d03.sas program in the demos folder    *;
*     and find the Demo section. Highlight and run the    *;
*     DESCRIBE TABLE statement and query in the Explore   *;
*     the sq.statepopulation table section. Examine the   *;
*     log and results.                        *;
*  2) Move to Method 1 - Down a Column. In the query      *;
*    complete the following:                         *;
*    a) Run the query and examine the results.          *;
*    b) In the SELECT clause, add three columns to find  *;
*       the standard deviation, minimum, and maximum of  *;
*       PopEstimate1 using the STD, MIN, and MAX        *;
*       functions. Use the COMMA16. format for all new   *;
*       columns. Highlight and run the query. Examine    *;
*       the log and results.                         *;
*    c) Move to SAS Method - PROC MEANS below the query. *;
*       SAS has procedures to do similar summarization.  *;
*       Highlight and run the MEANS procedure. Examine   *;
*       the log and results.                         *;
*  3) Move to Method 2 - Across a Column. In the query,   *;
*    complete the following:                         *;
*    a) The new column named Mean generates the average  *;
*       population estimate for the next three years for *;
*       each state using the AVG function. In addition,  *;
*       the MIN and MAX functions are used to create new *;
*       columns that generate the minimum and maximum    *;
*       population estimate. Highlight and run the       *;
*       query. Examine the log and results.            *;
*       Note: When more than one argument is used within *;
*           an SQL aggregate function, the function is *;
*           no longer considered to be an SQL aggregate*;
*           or summary function. This causes an error. *;
*    b) Replace the AVG function with the MEAN function. *;
*       Highlight and run the query. Examine the log and *;
*       results.                                   *;
*    c) In the MAX function, change the arguments to of  *;
*      PopEstimate1-PopEstimate3. Highlight and run the *;
*      query. Examine and discuss the syntax error.     *;
*******************************************************;


***********************************;
*EXPLORE THE STATEPOPULATION TABLE *;
***********************************;
proc sql inobs=10;
describe table sq.statepopulation;
select Region, Division, Name, PopEstimate1, PopEstimate2, PopEstimate3
   from sq.statepopulation;
quit;
```



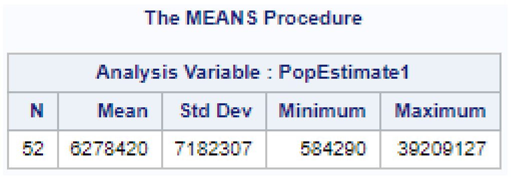
Region	Division	Name	PopEstimate1	PopEstimate2	PopEstimate3
3	6	AL	4864745	4875120	4887871
4	9	AK	741504	739786	737438
4	8	AZ	6945452	7048876	7171646
3	7	AR	2990410	3002997	3013825
4	9	CA	39209127	39399349	39557045
4	8	CO	5540921	5615902	5695564
1	1	CT	3578674	3573880	3572665
3	5	DE	949216	957078	967171
3	5	DC	686575	695691	702455
3	5	FL	20629982	20976812	21299325

```
********************************************************************************
*******;
*Method 1 - Down a Column: Find the count, mean, std, min and max of the PopEstimate1 column*;
********************************************************************************
*******;
proc sql;
select count(PopEstimate1) as TotalStates,
      mean(PopEstimate1) as Mean format=comma16.,
      std(PopEstimate1) as StdDev format=comma16.,
      min(PopEstimate1) as Min format=comma16.,
      max(PopEstimate1) as Max format=comma16.
   from sq.statepopulation;
quit;
```



Total States	Mean	StdDev	Min	Max
52	6,278,420	7,182,307	584,290	39,209,127

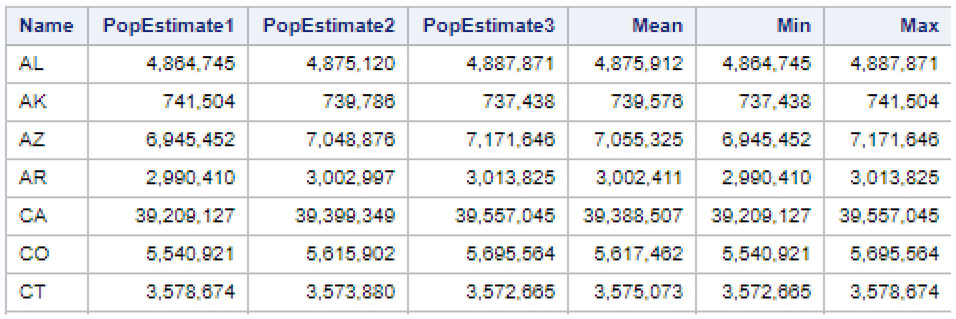
```
/*SAS Method - PROC MEANS*/
proc means data=sq.statepopulation maxdec=0;
            var PopEstimate1;
run;
```



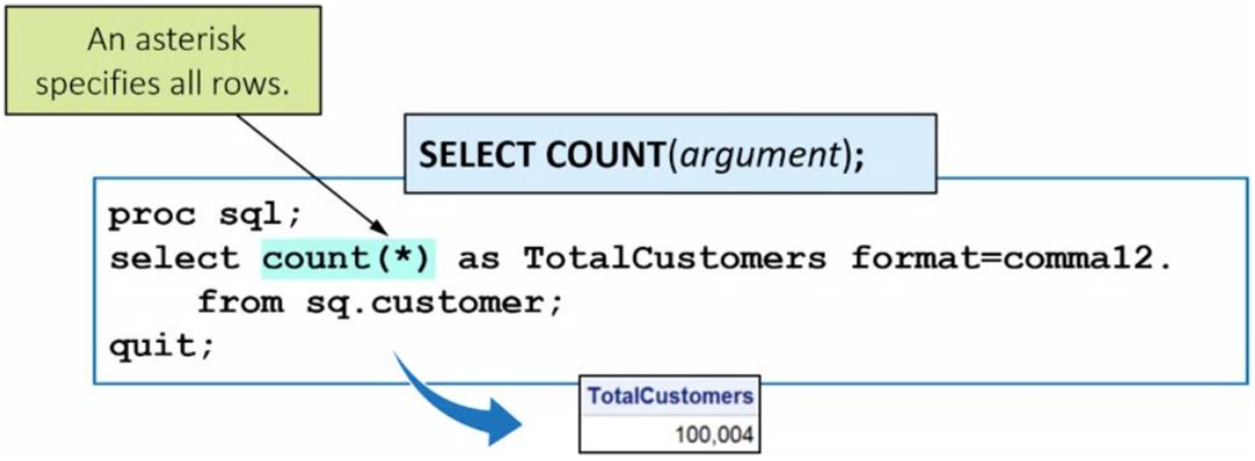
The MEANS Procedure

Analysis Variable : PopEstimate1				
N	Mean	Std Dev	Minimum	Maximum
52	6278420	7182307	584290	39209127

```
*************************************************************************************
**********;
*Method 2 - Across a Column: Find the mean, std, min and max of the PopEstimate1 column*
*************************************************************************************
**********;
proc sql;
select Name,
      PopEstimate1 format=comma16.,
      PopEstimate2 format=comma16.,
      PopEstimate3 format=comma16.,
               mean(PopEstimate1, PopEstimate2, PopEstimate3) as Mean format=comma16.,
      min(PopEstimate1, PopEstimate2, PopEstimate3) as Min format=comma16.,
      max(PopEstimate1, PopEstimate2, PopEstimate3) as Max format=comma16.
            from sq.statepopulation;
quit;
```



Name	PopEstimate1	PopEstimate2	PopEstimate3	Mean	Min	Max
AL	4,864,745	4,875,120	4,887,871	4,875,912	4,864,745	4,887,871
AK	741,504	739,786	737,438	739,576	737,438	741,504
AZ	6,945,452	7,048,876	7,171,646	7,055,325	6,945,452	7,171,646
AR	2,990,410	3,002,997	3,013,825	3,002,411	2,990,410	3,013,825
CA	39,209,127	39,399,349	39,557,045	39,388,507	39,209,127	39,557,045
CO	5,540,921	5,615,902	5,695,564	5,617,462	5,540,921	5,695,564
CT	3,578,674	3,573,880	3,572,665	3,575,073	3,572,665	3,578,674



```
proc sql;
select count(*) as TotalCustomers format=comma12.
    from sq.customer;
quit;
```

```
TotalCustomers
       100,004
```

***;

* Activity 2.07   *;

* 1) Examine and run the query. View the results. Why is *;

*   the value of MaritalStatus different from the value *;

*   of TotalRows?   *;

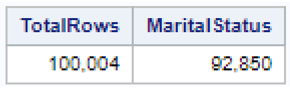
* 2) Inside the COUNT function, add the DISTINCT keyword *;

*   in front of the Married column and run the query.   *;

*   What does the new report show?   *;

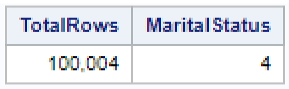
***;

```
proc sql;
select count(*) as TotalRows format=comma10.,
       count(Married) as MaritalStatus format=comma10.
             from sq.customer;
quit;
```

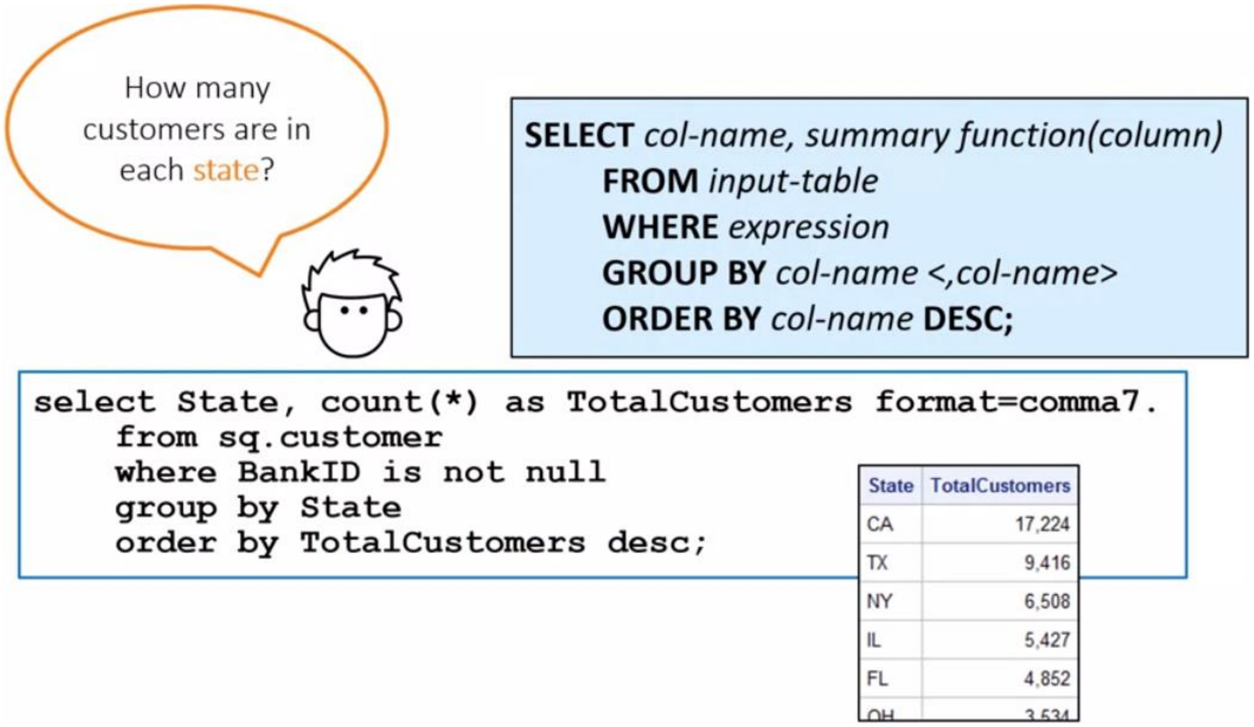


TotalRows	MaritalStatus
100,004	92,850

```
proc sql;

select count(*) as TotalRows format=comma10.,

    count(distinct Married) as MaritalStatus format=comma10.

        from sq.customer;

quit;
```



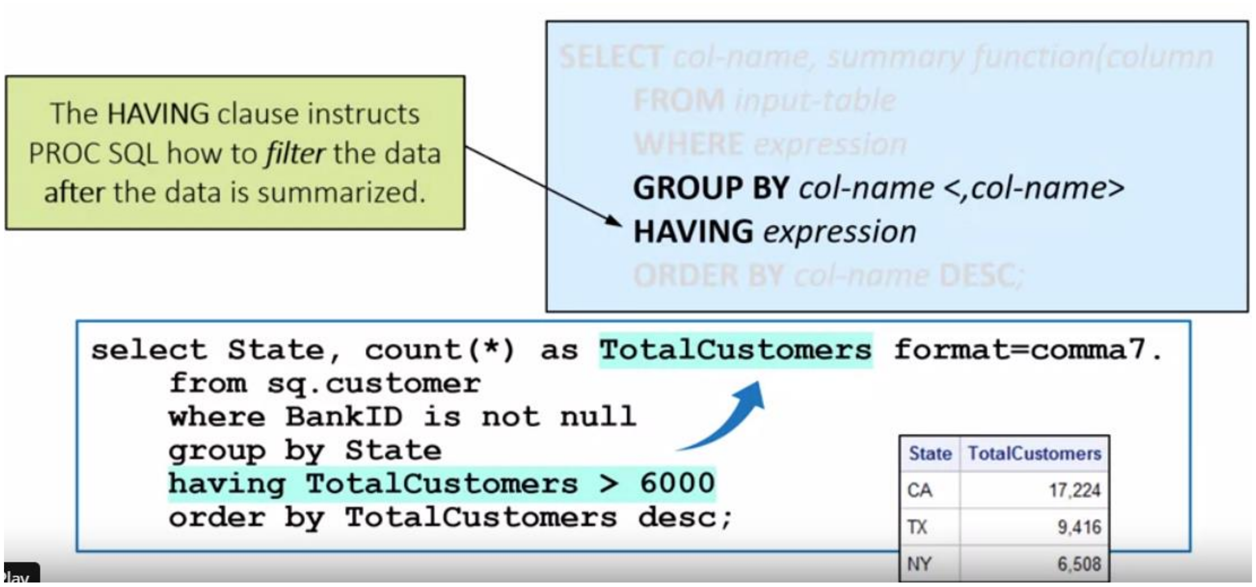
TotalRows	MaritalStatus
100,004	4



SELECT *col-name, summary function(column)*
FROM *input-table*
WHERE *expression*
GROUP BY *col-name <,col-name>*
ORDER BY *col-name* **DESC;**

```
select State, count(*) as TotalCustomers format=comma7.
    from sq.customer
    where BankID is not null
    group by State
    order by TotalCustomers desc;
```

State	TotalCustomers
CA	17,224
TX	9,416
NY	6,508
IL	5,427
FL	4,852
OH	3,534



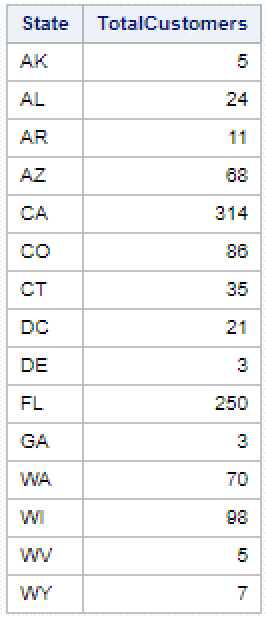
The HAVING clause instructs PROC SQL how to *filter* the data after the data is summarized.

SELECT *col-name, summary function(column*
FROM *input-table*
WHERE *expression*
GROUP BY *col-name <,col-name>*
HAVING *expression*
ORDER BY *col-name* DESC;

```
select State, count(*) as TotalCustomers format=comma7.
    from sq.customer
    where BankID is not null
    group by State
    having TotalCustomers > 6000
    order by TotalCustomers desc;
```

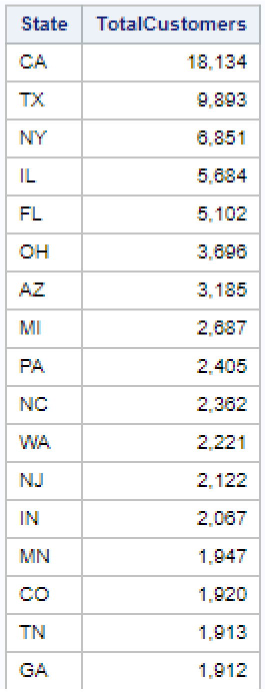
State	TotalCustomers
CA	17,224
TX	9,416
NY	6,508

```
*********************************************************;
*  Analyzing Groups of Data                   *;
*********************************************************;
*  Syntax                              *;
*                                  *;
*  PROC SQL;                               *;
*  SELECT col-name, col-name                   *;
*     FROM input-table                     *;
*     WHERE expression                      *;
*     GROUP BY col-name                      *;
*     HAVING expression                     *;
*     ORDER BY col-name <DESC>;                 *;
*  QUIT;                               *;
*********************************************************;


*********************************************************;
*  Demo                               *;
*  1) Open the s102d04.sas program in the demos folder   *;
*     and find the Demo section. Notice that the query    *;
*    creates a report of the State column in the        *;
*    customer table and limits the output to 1,000 rows. *;
*    Highlight and run the query. Examine the log and    *;
*    results.                                            *;
*    Note: The table is limited for development purposes *;
*         because the customer table has more than       *;
*         100,000 rows. After we finalize our query, we *;
*         can run it on the entire table.                *;
*    Note: When you use a GROUP BY clause without an     *;
*         aggregate function, PROC SQL treats the        *;
*         GROUP BY clause as if it were an ORDER BY      *;
*         clause, displaying a corresponding message in *;
*         the log.                                       *;
*  2) Modify the query to count the total number of      *;
*    customers in each state by using the COUNT          *;
*    function. Name the column TotalCustomers. Highlight *;
*    and run the query. Examine the log and results.     *;
*  3) Add an ORDER BY clause to the query to sort the    *;
*    report by descending TotalCustomers. Remove the     *;
*    OBS=1000 data set option and run the final query.   *;
*    Examine the log and results.                        *;
*  4) Replace State in the SELECT and GROUP BY clauses   *;
*    with BankID. Highlight and run the query. Examine   *;
*    the log and the results.                            *;
*    Note: Missing values are grouped and summarized.    *;
*  5) Add the Employed column after BankID in the SELECT  *;
*    and GROUP BY clauses. Highlight and run the query.  *;
*    Examine the log and the results.                    *;
*  6) Add a WHERE clause to filter for TotalCustomers    *;
*    greater than 10,000. Highlight and run the query.   *;
*    Examine the log and the results.                  *;
*    Note: Because the WHERE clause is evaluated before  *;
*         a row is available for processing and        *;
*         determines which individual rows are         *;
*         available for grouping, you cannot use a      *;
*         WHERE clause to subset grouped rows by       *;
*         referring to the calculated summary column.   *;
*  7) Remove the WHERE clause and insert a HAVING clause  *;
*    below the GROUP BY clause. Highlight and run the    *;
*    query. Examine the log and the results.            *;
*    Note: The order of the clauses is required.        *;
*******************************************************;


proc sql;
select State, count(*) as TotalCustomers format=comma7.
   from sq.customer(obs=1000)
   group by State;
quit;
```



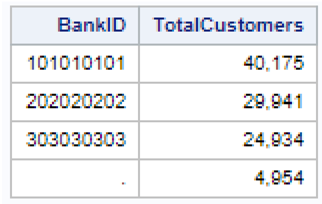
State	TotalCustomers
AK	5
AL	24
AR	11
AZ	68
CA	314
CO	86
CT	35
DC	21
DE	3
FL	250
GA	3
WA	70
WI	98
WV	5
WY	7

```
proc sql;
select State, count(*) as TotalCustomers format=comma7.
   from sq.customer
   group by State
   order by TotalCustomers desc;
quit;
```



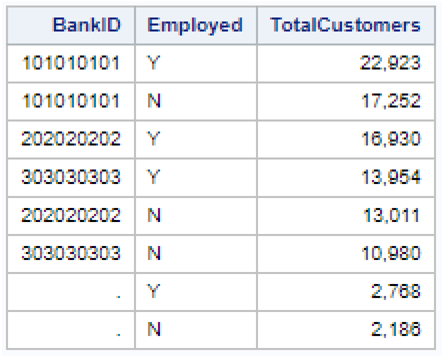
State	TotalCustomers
CA	18,134
TX	9,893
NY	6,851
IL	5,684
FL	5,102
OH	3,696
AZ	3,185
MI	2,687
PA	2,405
NC	2,362
WA	2,221
NJ	2,122
IN	2,067
MN	1,947
CO	1,920
TN	1,913
GA	1,912

```
proc sql;
select BankID, count(*) as TotalCustomers format=comma7.
   from sq.customer
   group by BankID
   order by TotalCustomers desc;
quit;
```



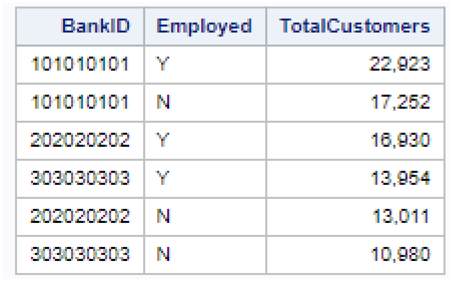
BankID	TotalCustomers
101010101	40,175
202020202	29,941
303030303	24,934
.	4,954

```
proc sql;
select BankID, Employed, count(*) as TotalCustomers format=comma7.
   from sq.customer
   group by BankID, Employed
   order by TotalCustomers desc;
quit;
```

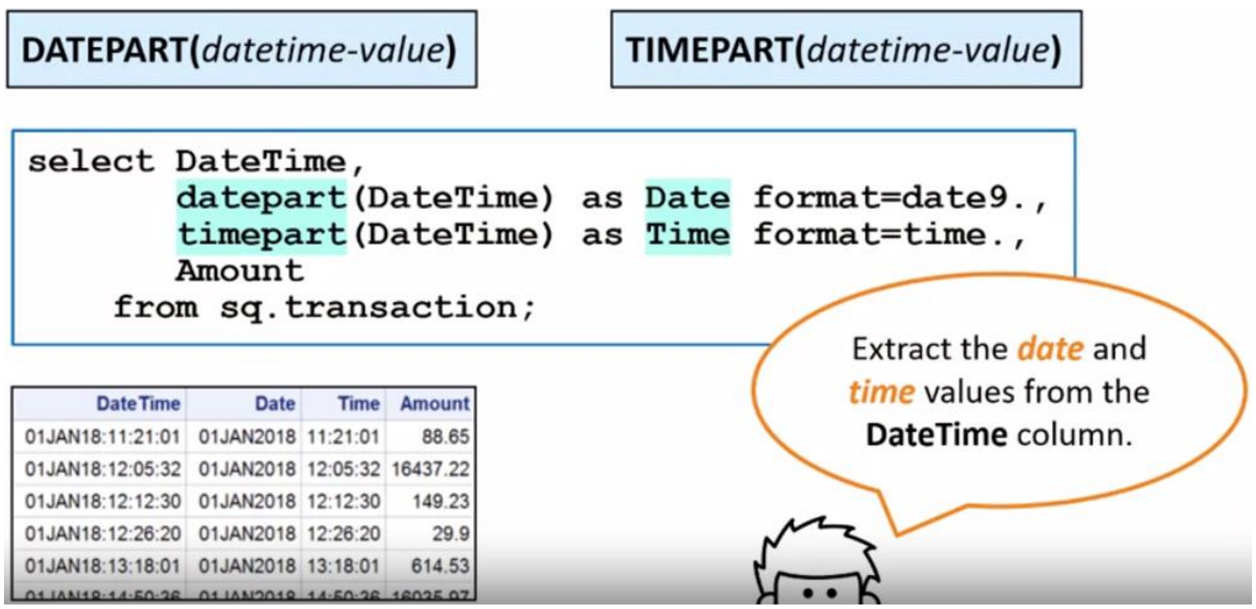


BankID	Employed	TotalCustomers
101010101	Y	22,923
101010101	N	17,252
202020202	Y	16,930
303030303	Y	13,954
202020202	N	13,011
303030303	N	10,980
.	Y	2,768
.	N	2,186

```
proc sql;
select BankID, Employed, count(*) as TotalCustomers format=comma7.
   from sq.customer
   group by BankID, Employed
   having calculated TotalCustomers > 10000
   order by TotalCustomers desc;
quit;
```



BankID	Employed	TotalCustomers
101010101	Y	22,923
101010101	N	17,252
202020202	Y	16,930
303030303	Y	13,954
202020202	N	13,011
303030303	N	10,980



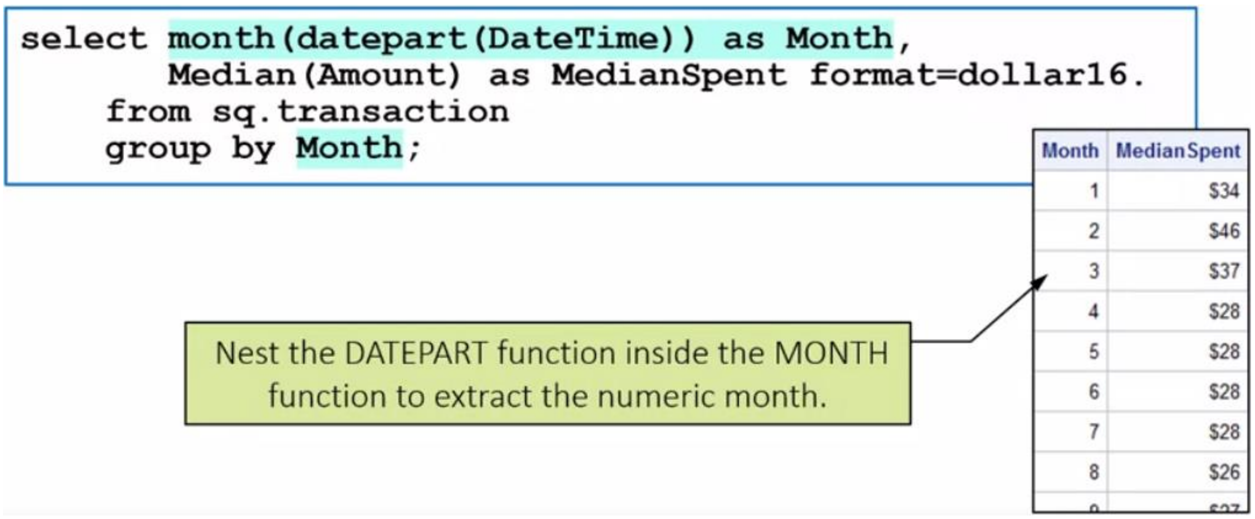
DATEPART(*datetime-value*)

TIMEPART(*datetime-value*)

```
select DateTime,
       datepart(DateTime) as Date format=date9.,
       timepart(DateTime) as Time format=time.,
       Amount
   from sq.transaction;
```

DateTime	Date	Time	Amount
01JAN18:11:21:01	01JAN2018	11:21:01	88.65
01JAN18:12:05:32	01JAN2018	12:05:32	16437.22
01JAN18:12:12:30	01JAN2018	12:12:30	149.23
01JAN18:12:26:20	01JAN2018	12:26:20	29.9
01JAN18:13:18:01	01JAN2018	13:18:01	614.53

Extract the *date* and *time* values from the **DateTime** column.

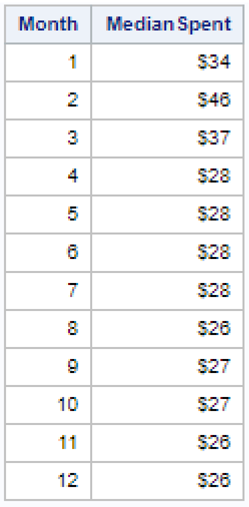


```
select month(datepart(DateTime)) as Month,
       Median(Amount) as MedianSpent format=dollar16.
   from sq.transaction
   group by Month;
```

Month	MedianSpent
1	$34
2	$46
3	$37
4	$28
5	$28
6	$28
7	$28
8	$26

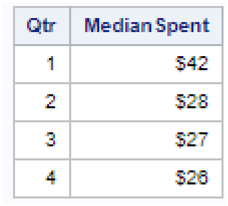
Nest the DATEPART function inside the MONTH function to extract the numeric month.

```
**************************************************************;
*  Activity 2.08                              *;
*  1) Examine and run the query. View the results. Which  *;
*     month has the highest value for MedianSpent?        *;
*  2) Replace the MONTH function with the QTR function.   *;
*     Change the name of the Month column to Qtr. Run the *;
*     query. What is the error?                        *;
*  3) Replace Month in the GROUP BY clause with Qtr. Run  *;
*     the query. Which quarter has the highest value for  *;
*     MedianSpent?                                  *;
*********************************************************;

proc sql;
select month(datepart(DateTime)) as Month,
       Median(Amount) as MedianSpent format=dollar16.
   from sq.transaction
   group by Month;
quit;
```



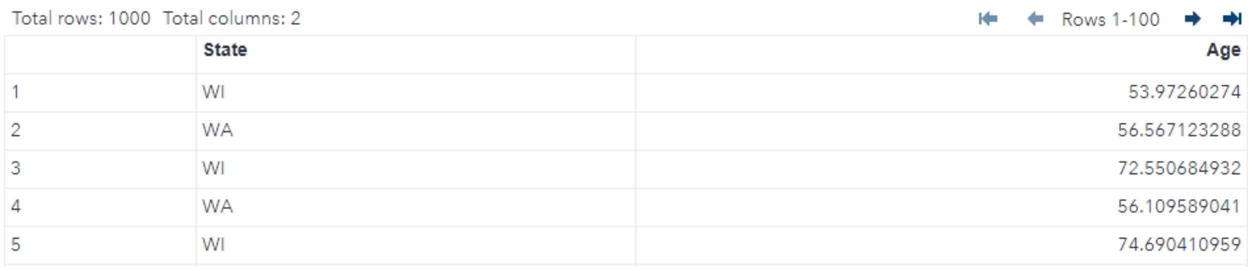
Month	Median Spent
1	$34
2	$46
3	$37
4	$28
5	$28
6	$28
7	$28
8	$26
9	$27
10	$27
11	$26
12	$26

```
proc sql;
select Qtr(datepart(DateTime)) as Qtr,
       Median(Amount) as MedianSpent format=dollar16.
   from sq.transaction
   group by Qtr;
quit;
```



Qtr	Median Spent
1	$42
2	$28
3	$27
4	$26

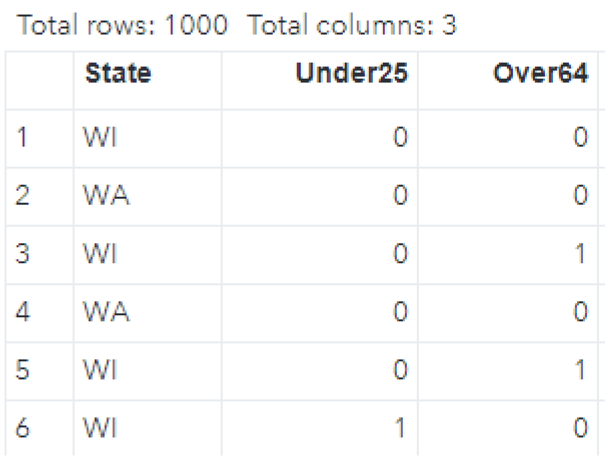
```
***********************************************************;
*  Summarizing Data Using a Boolean                    *;
***********************************************************;
*  Syntax                                  *;
*
              *;
*  YRDIF(start-date, end-date <,'basis'>)               *;
***********************************************************;


***********************************************************;
*  Demo                                    *;
*  1) Open the s102d05.sas program in the demos folder    *;
*     and find the Demo section. Run the query and       *;
*     examine the results.                          *;
*  2) Add the less than operator after the YRDIF function *;
*     test if the row is under 25 years old. Rename the   *;
*     column Under25. Run the query and examine the       *;
*     results.                                   *;
*  3) Copy the expression. Replace the < comparison       *;
*     operator with the > comparison operator. Change the *;
*     value from 25 to 64 and the name from Under25 to    *;
*     Over64. Run the query and examine the results.     *;
*  4) Summarize the data by wrapping each new column with *;
*     the SUM function to add all the values of 1 to      *;
*     count the number of customers. Add a GROUP BY       *;
*     clause with the State column. Remove the INOBS=     *;
*     option. Run the query and examine the results.      *;
**********************************************************;

proc sql inobs=1000;
create table CustomerCount as
select State,
      yrdif(DOB,"01JAN2020"d,'age') as Age
   from sq.customer;
quit;
```



Total rows: 1000 Total columns: 2   Rows 1-100

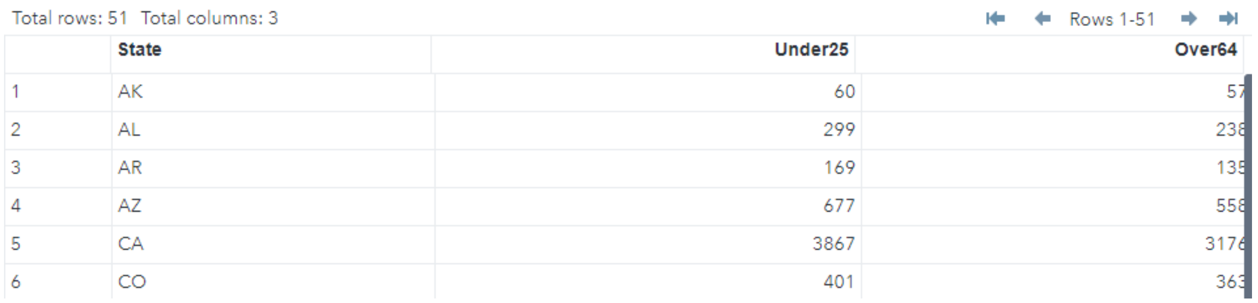
	State	Age
1	WI	53.97260274
2	WA	56.567123288
3	WI	72.550684932
4	WA	56.109589041
5	WI	74.690410959

```
proc sql inobs=1000;
create table CustomerCount as
select State,
      yrdif(DOB,"01JAN2020"d,'age') < 25 as Under25,
      yrdif(DOB,"01JAN2020"d,'age') > 64 as Over64
   from sq.customer;
quit;
```


Total rows: 1000 Total columns: 3

	State	Under25	Over64
1	WI	0	0
2	WA	0	0
3	WI	0	1
4	WA	0	0
5	WI	0	1
6	WI	1	0

```
proc sql;
create table CustomerCount as
select State,
    sum(yrdif(DOB,"01JAN2020"d,'age') < 25) as Under25,
    sum(yrdif(DOB,"01JAN2020"d,'age') > 64) as Over64
  from sq.customer
  group by State;
quit;
```


Total rows: 51 Total columns: 3

Rows 1-51

	State	Under25	Over64
1	AK	60	57
2	AL	299	238
3	AR	169	135
4	AZ	677	558
5	CA	3867	3176
6	CO	401	363

/*Level 1 Practice: Eliminating Duplicates

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.globalfull table contains estimated financial information by geographic region and country for the population age 15 years and older.

Using the sq.globalfull table, write a query to generate a report that displays the unique CountryCode values.

Use the following requirements as you generate the report:

Order the rows by the CountryCode.

Add the title Unique Country Codes.

Run the program and view the results.

Do you see any duplicate country codes?

Modify the query to produce a count of the unique CountryCode values.

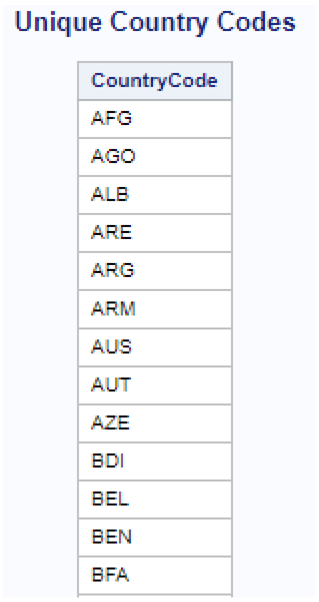
Name the result of the count CountryCount.

Add the title Count of Unique Country Codes.

Run the program and view the results.

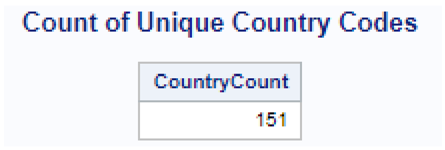
How many unique country codes are in the sq.globalfull table?

```
*/
title "Unique Country Codes";
proc sql;
Select distinct CountryCode
        from sq.globalfull
        order by CountryCode;
quit;
title;
```


Unique Country Codes

CountryCode
AFG
AGO
ALB
ARE
ARG
ARM
AUS
AUT
AZE
BDI
BEL
BEN
BFA

```
title "Count of Unique Country Codes";

proc sql;

Select count(distinct CountryCode) as CountryCount

        from sq.globalfull;

quit;

title;
```



Count of Unique Country Codes

CountryCount
151

/*Level 2 Practice: Grouping and Summarizing Data

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.transactionfull table contains a list of customers with transaction information.

Using the sq.transactionfull table, write a query to generate a report that identifies which

customers have the greatest percentage of suspiciously large transactions (over $500).

Use the following requirements as you generate the report:

Select and group the report by CustomerID.

Create the column TotalTransactions using the COUNT(*) function to count the number of transactions for each value of CustomerID.

Create the column SuspiciousTransactions as SUM(Amount >= 500) to count the number of transactions greater than 500.

Create the column PCTSuspicious by dividing SuspiciousTransactions by TotalTransactions. Format the new column with PERCENT8.2.

Select only transactions where the Service value is not equal to University.

Filter the output to display only summary rows where PCTSuspicious > .05.

Order the report by descending PCTSuspicious.

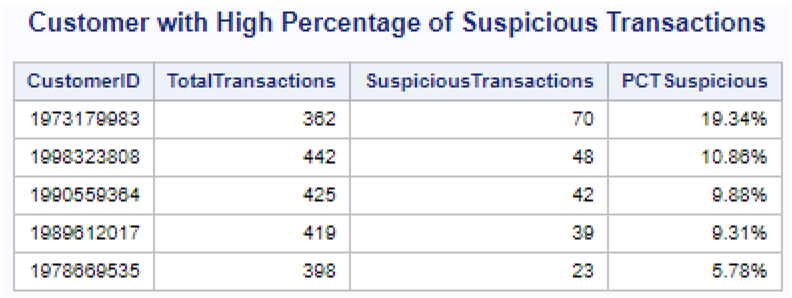
Add the title Customers with High Percentage of Suspicious Transactions.

Run the program and view the results.

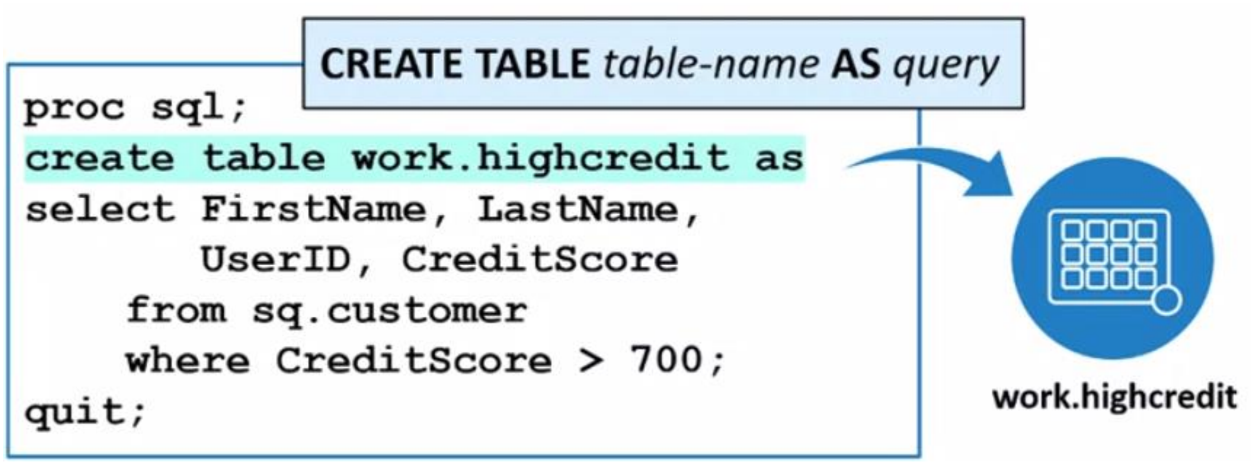
Which CustomerID value had the highest percentage of suspicious transactions?

*/

```
title "Customer with High Percentage of Suspicious Transactions";

proc sql;

Select CustomerID, count(*) as TotalTransactions,

            sum(Amount >=500) as SuspiciousTransactions,

            (calculated SuspiciousTransactions/calculated TotalTransactions) as PCTSuspicious
format=percent8.2

          from sq.transactionfull

          where Service <> 'University'

          group by CustomerID

          having calculated PCTSuspicious > .05

          order by PCTSuspicious desc;

quit;

title;


/*s102s06.sas*/


title 'Customers with High Percentage of Suspicious Transactions';

proc sql;

select CustomerID,

     count(*) as TotalTransactions,

     sum(Amount >= 500) as SuspiciousTransactions,

     calculated SuspiciousTransactions/calculated TotalTransactions as PCTSuspicious format=percent8.2

   from sq.transactionfull

   where Service ^= 'University'

   group by CustomerID

   having PCTSuspicious>.05

   order by PCTSuspicious desc;

quit;

title;
```


Customer with High Percentage of Suspicious Transactions

CustomerID	TotalTransactions	SuspiciousTransactions	PCT Suspicious
1973179983	362	70	19.34%
1998323808	442	48	10.86%
1990559364	425	42	9.88%
1989612017	419	39	9.31%
1978669535	398	23	5.78%


CREATE TABLE *table-name* AS *query*

```
proc sql;
create table work.highcredit as
select FirstName, LastName,
       UserID, CreditScore
    from sq.customer
    where CreditScore > 700;
quit;
```

```
**********************************************************;
*  Activity 2.09                            *;
*  1) Examine and run the query in the Create a Table     *;
*     from a Query section. View the results.           *;
*  2) Add the CREATE TABLE statement and create a table   *;
*     named Top5States. Run the query and confirm that    *;
*     the table was created successfully.                *;
*  3) Run the code below your SQL query. What did the     *;
*     code produce?                                     *;
**********************************************************;


********************************;
*Create a Table from a Query *;
********************************;
proc sql outobs=5;
```

```
/*Add a CREATE TABLE Statement*/

Create table Top5States as

select Name label="State Name",

      PopEstimate1 format=comma14. label="Population Estimate"

   from sq.statepopulation

            order by PopEstimate1 desc;

quit;
```

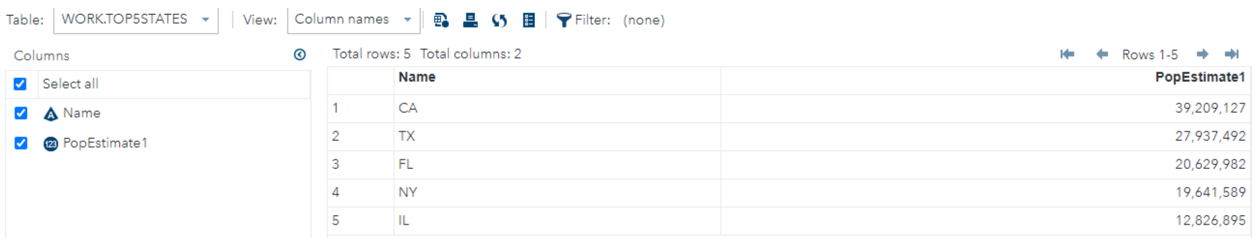


Table: WORK.TOP5STATES View: Column names Filter: (none)

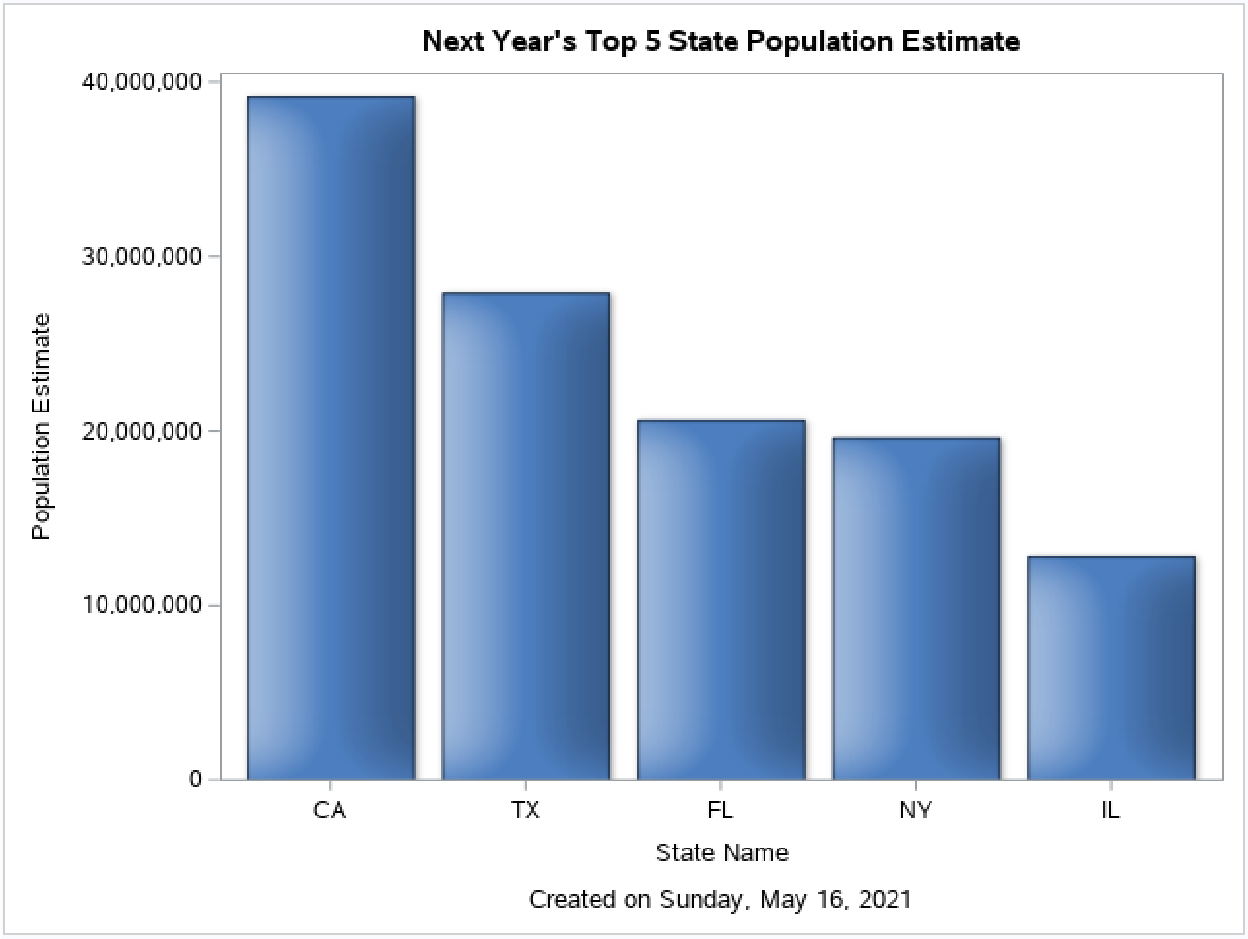
Columns

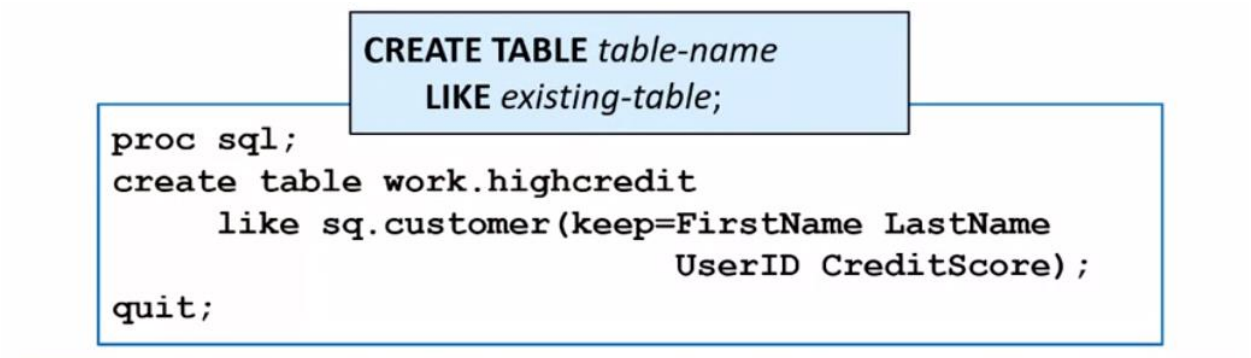
- ✓ Select all
- ✓ Name
- ✓ PopEstimate1

Total rows: 5 Total columns: 2 Rows 1-5

	Name	PopEstimate1
1	CA	39,209,127
2	TX	27,937,492
3	FL	20,629,982
4	NY	19,641,589
5	IL	12,826,895

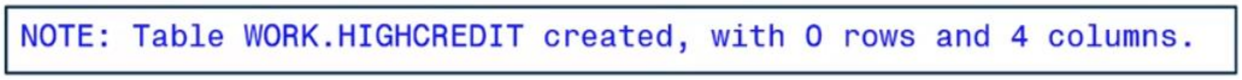
```
*******************************************************;

*          USE THE TABLE IN A VISUALIZATION          *;

*                                   DO NOT EDIT CODE BELOW            *;

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

title "Next Year's Top 5 State Population Estimate";

footnote "Created on %left(%qsysfunc(today(),weekdate.))";/*<-----Automatically adds the current date as the footnote*/

proc sgplot data=Top5States; /*<------------Top5States table from above*/

  vbar Name / response=PopEstimate1 /*<----Specifies the numeric response value*/

                                categoryorder=respdesc /*<---Specify the order in which the columns are arranged*/

            dataskin=matte /*<-----------Specifies a special effect to be used on the bars*/

                                fillattrs=(color=bigb);/*<---Specifies the fill color*/

run;

title;

footnote;
```



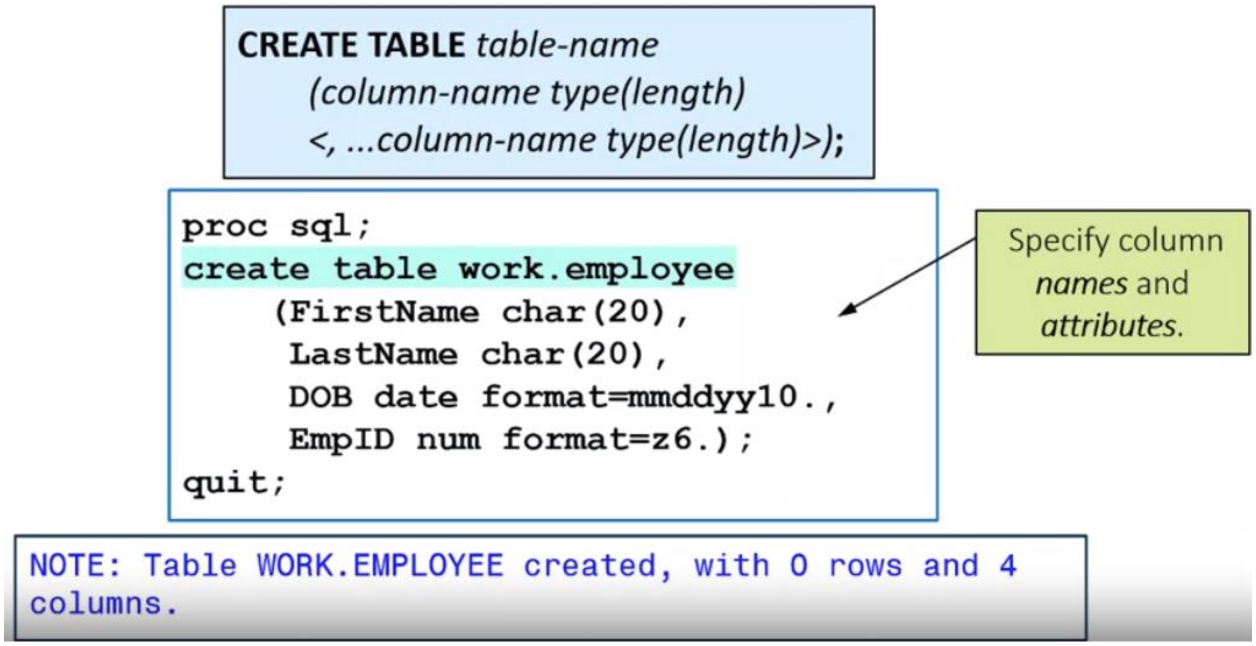


CREATE TABLE *table-name*
LIKE *existing-table*;

```
proc sql;
create table work.highcredit
     like sq.customer(keep=FirstName LastName
                                        UserID CreditScore);
quit;
```



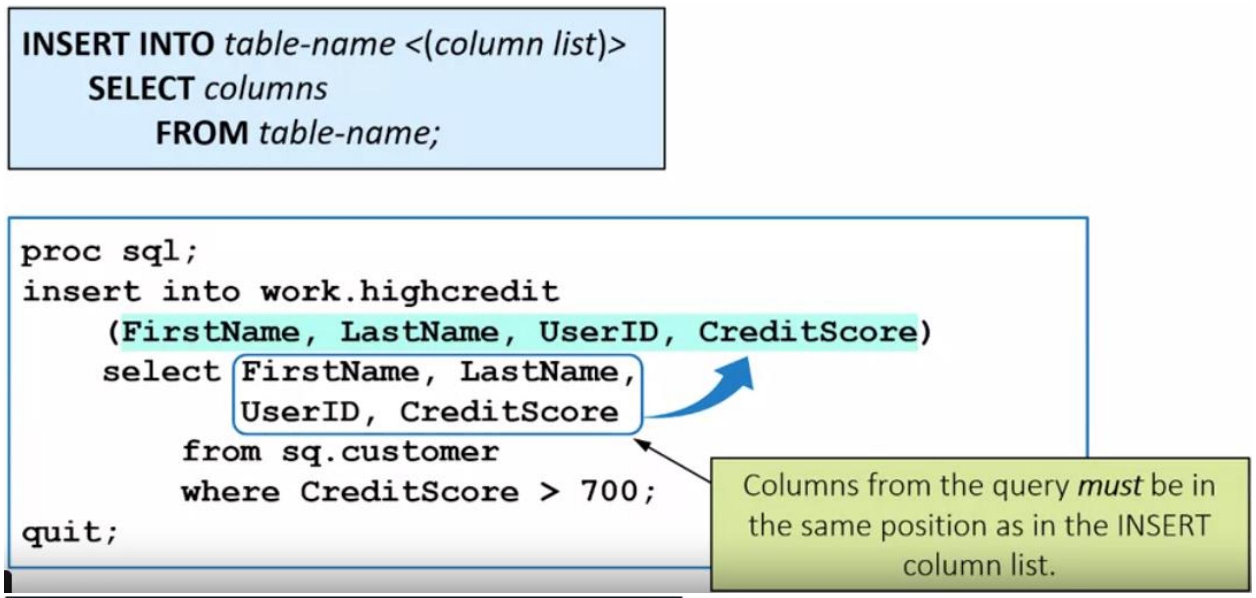
```
NOTE: Table WORK.HIGHCREDIT created, with 0 rows and 4 columns.
```



CREATE TABLE *table-name*
(column-name type(length)
<, ...column-name type(length)>);

```
proc sql;
create table work.employee
    (FirstName char(20),
     LastName char(20),
     DOB date format=mmddyy10.,
     EmpID num format=z6.);
quit;
```

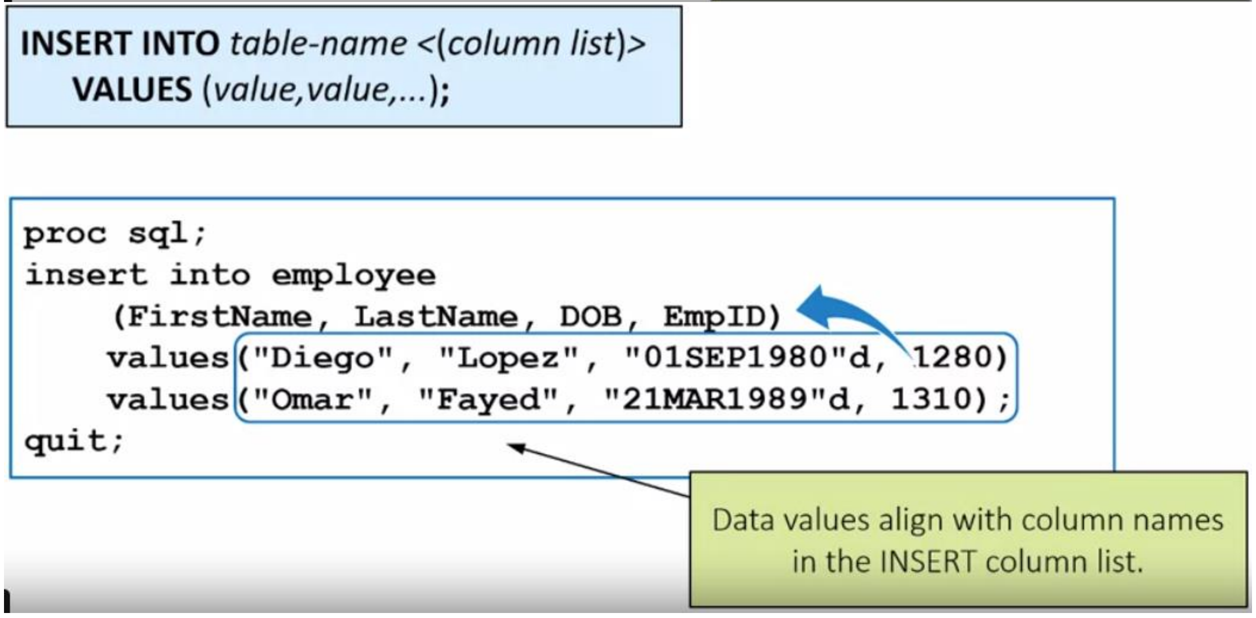
```
NOTE: Table WORK.EMPLOYEE created, with 0 rows and 4
columns.
```



INSERT INTO *table-name* <(*column list*)>
    SELECT *columns*
        FROM *table-name;*

```
proc sql;
insert into work.highcredit
    (FirstName, LastName, UserID, CreditScore)
    select FirstName, LastName,
           UserID, CreditScore
        from sq.customer
        where CreditScore > 700;
quit;
```

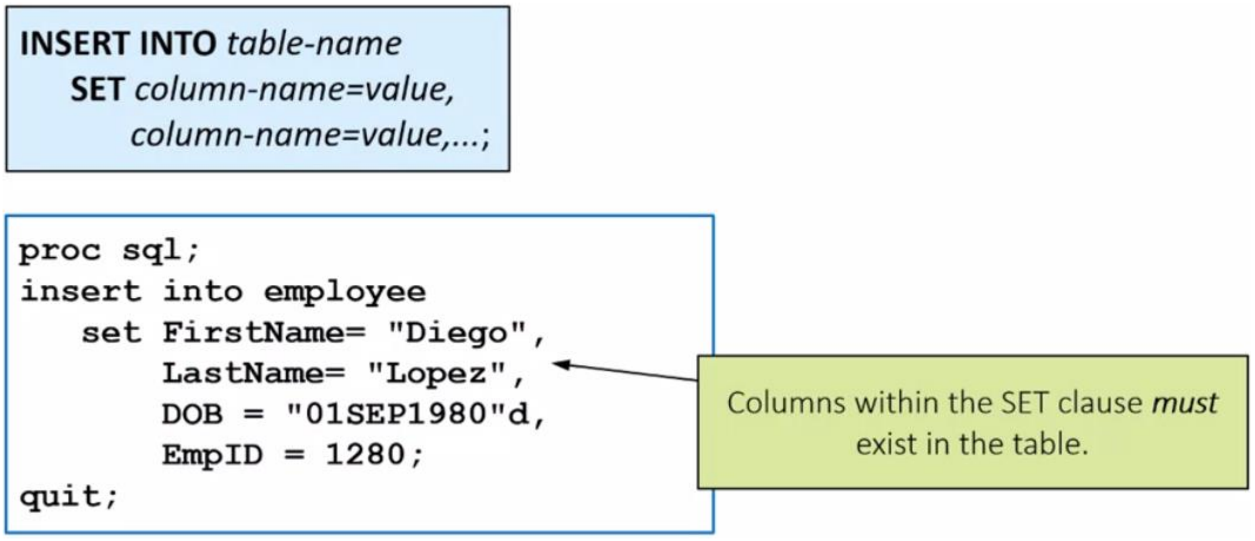
Columns from the query *must* be in the same position as in the INSERT column list.



INSERT INTO *table-name* <(*column list*)>
    VALUES (*value,value,...*)**;**

```
proc sql;
insert into employee
    (FirstName, LastName, DOB, EmpID)
    values("Diego", "Lopez", "01SEP1980"d, 1280)
    values("Omar", "Fayed", "21MAR1989"d, 1310);
quit;
```

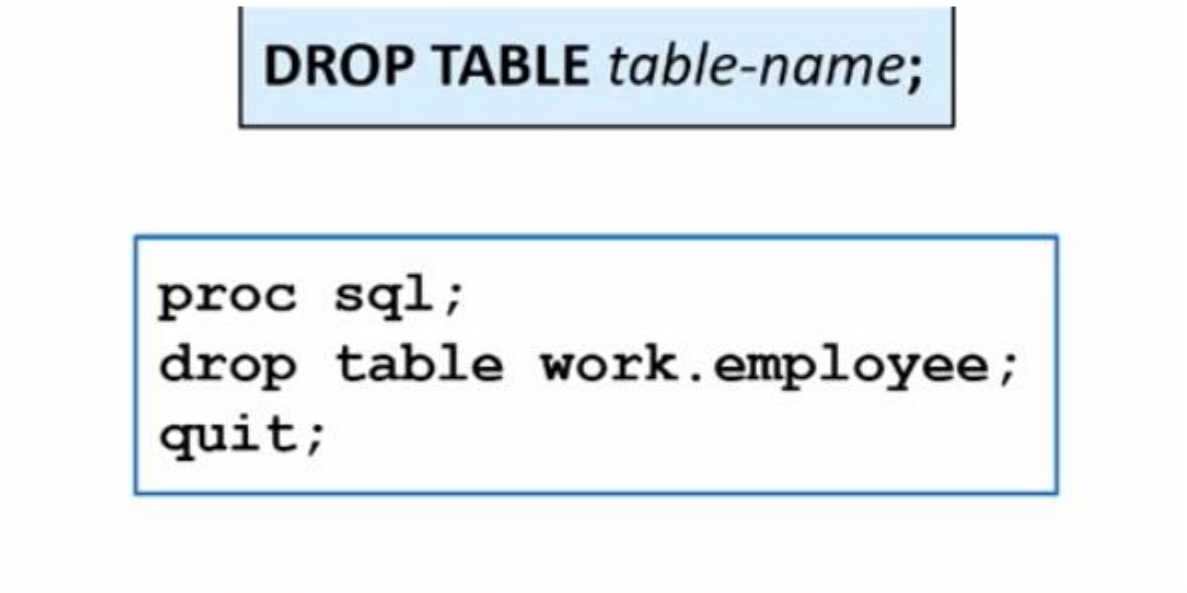
Data values align with column names in the INSERT column list.



INSERT INTO *table-name*
  SET *column-name=value,*
    column-name=value,...;

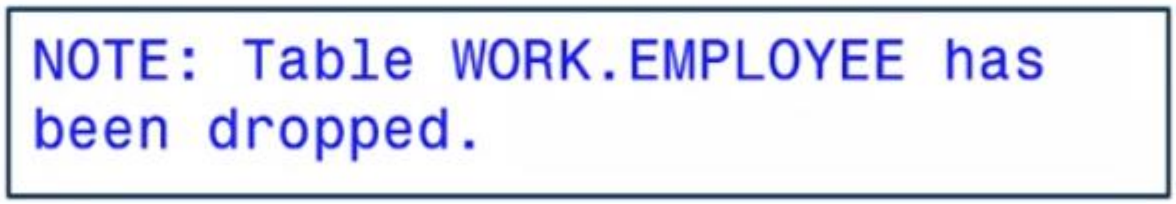
```
proc sql;
insert into employee
   set FirstName= "Diego",
       LastName= "Lopez",
       DOB = "01SEP1980"d,
       EmpID = 1280;
quit;
```

Columns within the SET clause *must* exist in the table.



DROP TABLE *table-name***;**

```
proc sql;
drop table work.employee;
quit;
```



```
NOTE: Table WORK.EMPLOYEE has
been dropped.
```

```
**************************************************************;
*  Activity 2.10                                          *;
*  1) Examine the CREATE TABLE statement and run the      *;
*     query only. Confirm an empty table was created.     *;
*  2) In the Inserting Rows with a Query section, enter   *;
*     the correct column names to complete the INSERT     *;
*    INTO statement. Run the query. How many rows were   *;
*    inserted into the table highcredit?                 *;
*  3) In the Inserting Rows with the SET Clause section,  *;
*    complete the INSERT INTO statement with the SET     *;
*    clause and insert yourself as a customer into the   *;
*    highcredit table. Run the query. What does the note *;
*    in the log say?                                     *;
*  4) Complete the code to drop the highcredit table.     *;
**********************************************************;

*******************************************;
*Creating Tables like an Existing Table    *;
*******************************************;
proc sql;
create table work.highcredit
    like sq.customer(keep=FirstName LastName
                     UserID CreditScore);
quit;

*******************************************;
*Inserting Rows with a Query               *;
*******************************************;
proc sql;
insert into work.highcredit(FirstName, LastName, UserID, CreditSCore)
select FirstName, LastName,
      UserID, CreditScore
    from sq.customer
    where CreditScore > 700;
quit;
```

```
        ******************************************;
        *Inserting Rows with a Query             *;
        ******************************************;
        proc sql;
        insert into work.highcredit(FirstName, LastName, UserID, CreditSCore)
        select FirstName, LastName,
               UserID, CreditScore
             from sq.customer
             where CreditScore > 700;
NOTE: 26006 rows were inserted into WORK.HIGHCREDIT.
```

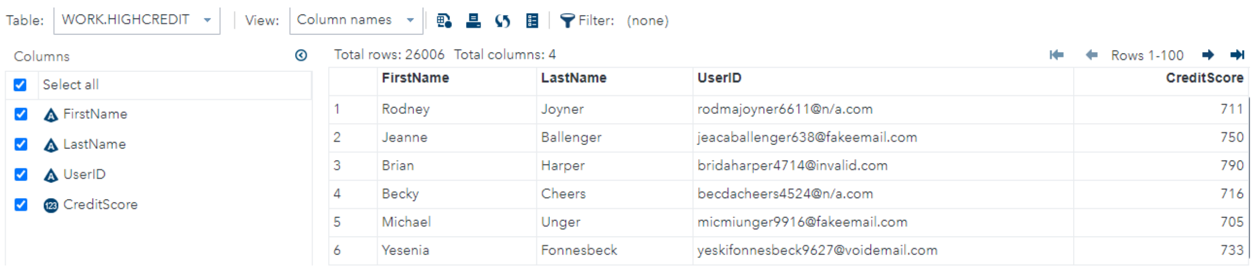


Table: WORK.HIGHCREDIT | View: Column names | Filter: (none)

Columns
- ☑ Select all
- ☑ FirstName
- ☑ LastName
- ☑ UserID
- ☑ CreditScore

Total rows: 26006 Total columns: 4 | Rows 1-100

	FirstName	LastName	UserID	CreditScore
1	Rodney	Joyner	rodmajoyner6611@n/a.com	711
2	Jeanne	Ballenger	jeacaballenger638@fakeemail.com	750
3	Brian	Harper	bridaharper4714@invalid.com	790
4	Becky	Cheers	becdacheers4524@n/a.com	716
5	Michael	Unger	micmiunger9916@fakeemail.com	705
6	Yesenia	Fonnesbeck	yeskifonnesbeck9627@voidemail.com	733

**;

*Inserting Rows with the SET Clause *;

**;

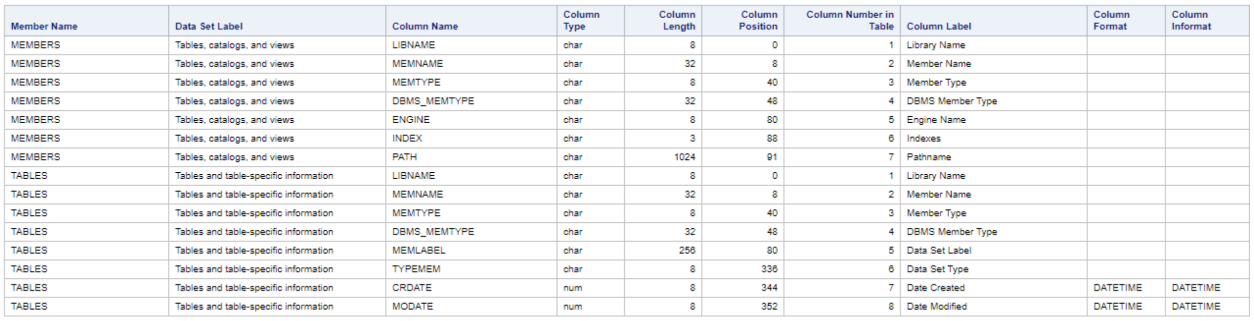
```
proc sql;

insert into highcredit

   set FirstName="William",  /*<-----Add your first name*/

            LastName="Chan",   /*<-----Add your last name*/

                  UserID="9861",   /*<-----Add your first initial followed by your last name*/

                  CreditScore=818; /*<-----Add any number from 701 - 850*/

quit;
```

```
        proc sql;
        insert into highcredit
            set FirstName="William",   /*<-----Add your first name*/
            LastName="Chan",    /*<-----Add your last name*/
        UserID="9861",    /*<-----Add your first initial followed by your last name*/
        CreditScore=818;
NOTE: 1 row was inserted into WORK.HIGHCREDIT.
```

**;

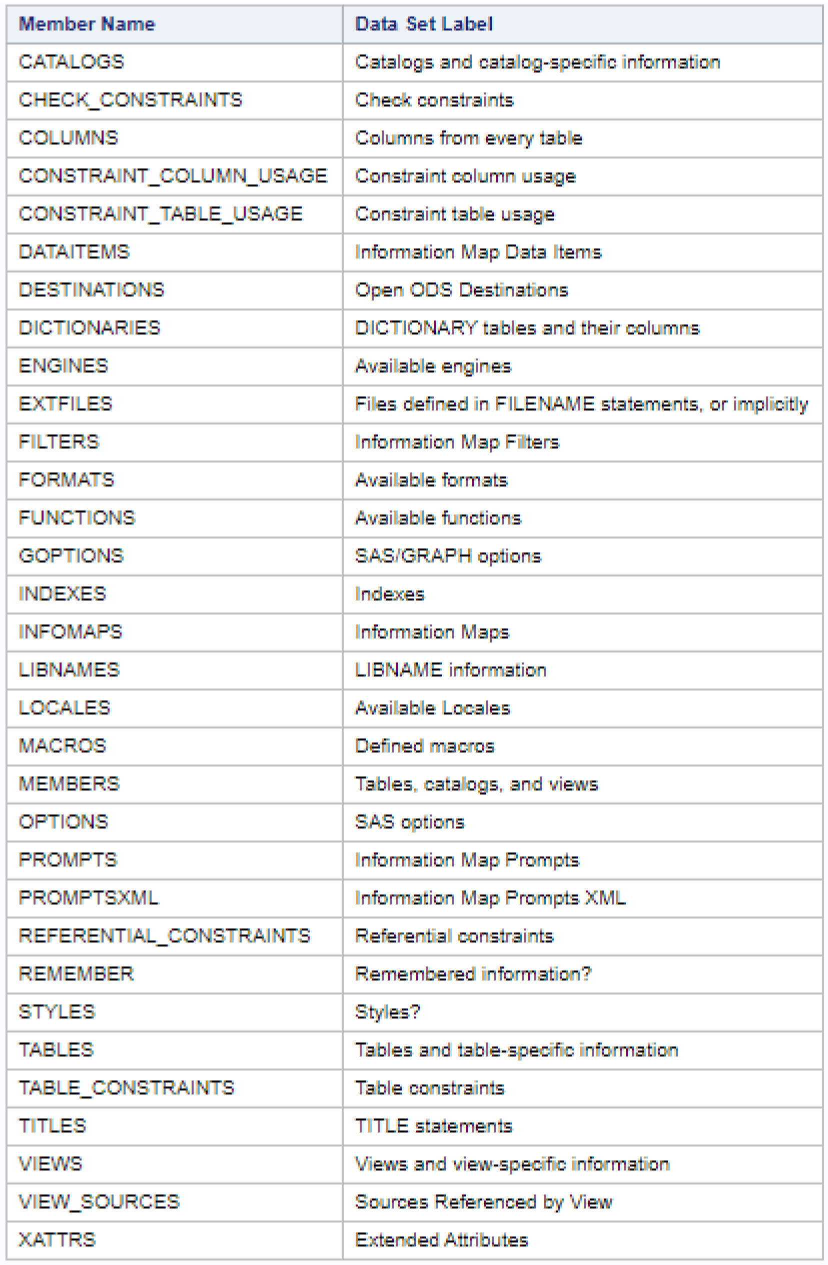
*DROP the highcredit Table *;

```
*******************************************;
proc sql;
        drop table work.highcredit;
quit;

*******************************************************;
*  Activity 2.11                          *;
*  1) Examine and run the program. View the log and      *;
*    results.                             *;
*  2) Note the column labels for the first two columns:   *;
*    Member Name is the DICTIONARY table, and Data Set   *;
*    Label is the description of that table.          *;
*  3) Replace the asterisk in the SELECT clause and      *;
*    select the DISTINCT memname and memlabel columns.   *;
*    Run the query and examine all the available        *;
*    DICTIONARY tables in your SAS session.            *;
*  4) What is the data set label of the members          *;
*    DICTIONARY table?                          *;
*******************************************************;

proc sql;
describe table dictionary.dictionaries;
select *
        from dictionary.dictionaries;
quit;
```



Member Name	Data Set Label	Column Name	Column Type	Column Length	Column Position	Column Number in Table	Column Label	Column Format	Column Informat
MEMBERS	Tables, catalogs, and views	LIBNAME	char	8	0	1	Library Name		
MEMBERS	Tables, catalogs, and views	MEMNAME	char	32	8	2	Member Name		
MEMBERS	Tables, catalogs, and views	MEMTYPE	char	8	40	3	Member Type		
MEMBERS	Tables, catalogs, and views	DBMS_MEMTYPE	char	32	48	4	DBMS Member Type		
MEMBERS	Tables, catalogs, and views	ENGINE	char	8	80	5	Engine Name		
MEMBERS	Tables, catalogs, and views	INDEX	char	3	88	6	Indexes		
MEMBERS	Tables, catalogs, and views	PATH	char	1024	91	7	Pathname		
TABLES	Tables and table-specific information	LIBNAME	char	8	0	1	Library Name		
TABLES	Tables and table-specific information	MEMNAME	char	32	8	2	Member Name		
TABLES	Tables and table-specific information	MEMTYPE	char	8	40	3	Member Type		
TABLES	Tables and table-specific information	DBMS_MEMTYPE	char	32	48	4	DBMS Member Type		
TABLES	Tables and table-specific information	MEMLABEL	char	256	80	5	Data Set Label		
TABLES	Tables and table-specific information	TYPEMEM	char	8	336	6	Data Set Type		
TABLES	Tables and table-specific information	CRDATE	num	8	344	7	Date Created	DATETIME	DATETIME
TABLES	Tables and table-specific information	MODATE	num	8	352	8	Date Modified	DATETIME	DATETIME

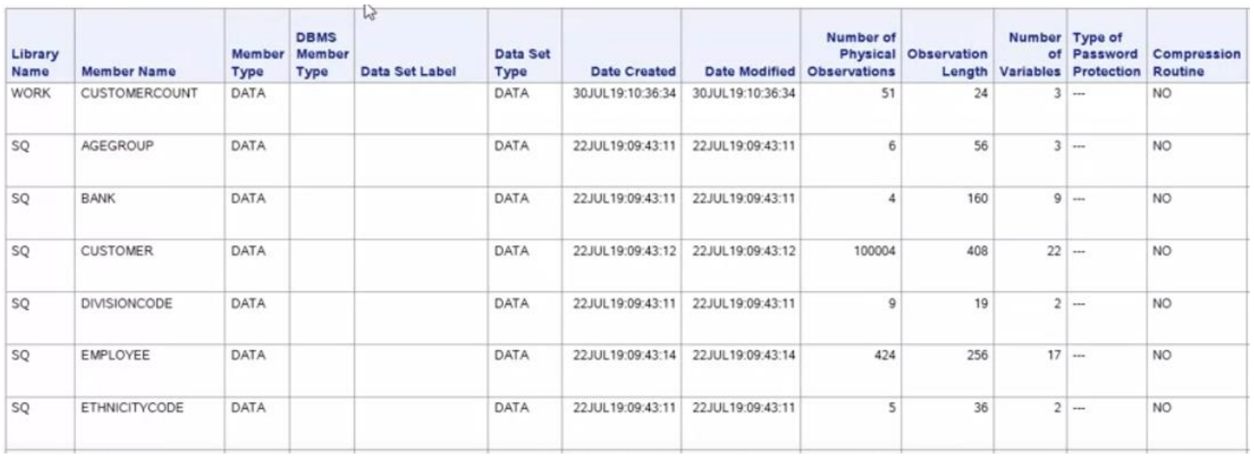
```
proc sql;

describe table dictionary.dictionaries;

select distinct memname, memlabel

            from dictionary.dictionaries;

quit;
```



Member Name	Data Set Label
CATALOGS	Catalogs and catalog-specific information
CHECK_CONSTRAINTS	Check constraints
COLUMNS	Columns from every table
CONSTRAINT_COLUMN_USAGE	Constraint column usage
CONSTRAINT_TABLE_USAGE	Constraint table usage
DATAITEMS	Information Map Data Items
DESTINATIONS	Open ODS Destinations
DICTIONARIES	DICTIONARY tables and their columns
ENGINES	Available engines
EXTFILES	Files defined in FILENAME statements, or implicitly
FILTERS	Information Map Filters
FORMATS	Available formats
FUNCTIONS	Available functions
GOPTIONS	SAS/GRAPH options
INDEXES	Indexes
INFOMAPS	Information Maps
LIBNAMES	LIBNAME information
LOCALES	Available Locales
MACROS	Defined macros
MEMBERS	Tables, catalogs, and views
OPTIONS	SAS options
PROMPTS	Information Map Prompts
PROMPTSXML	Information Map Prompts XML
REFERENTIAL_CONSTRAINTS	Referential constraints
REMEMBER	Remembered information?
STYLES	Styles?
TABLES	Tables and table-specific information
TABLE_CONSTRAINTS	Table constraints
TITLES	TITLE statements
VIEWS	Views and view-specific information
VIEW_SOURCES	Sources Referenced by View
XATTRS	Extended Attributes

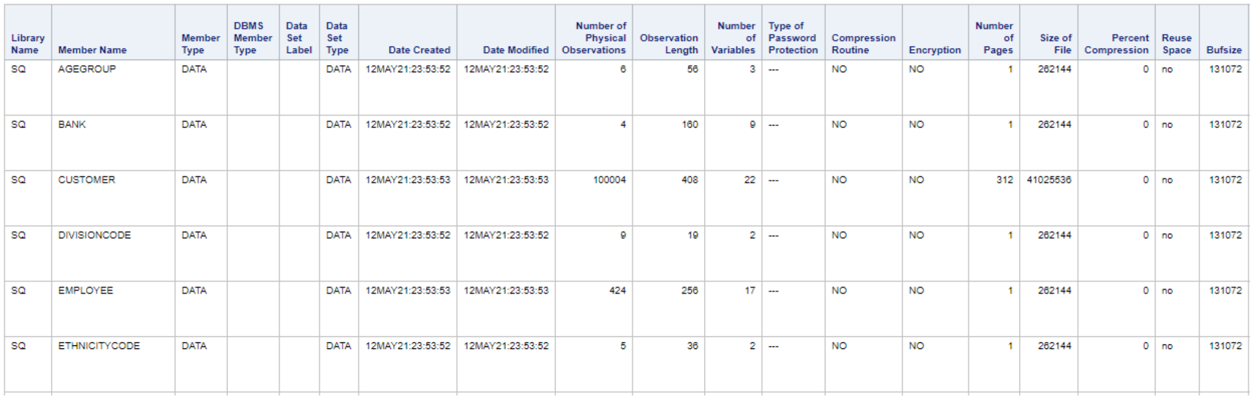
```
*******************************************************;
*  Using DICTIONARY Tables                        *;
*******************************************************;
*  Syntax                                 *;
*                                     *;
*  PROC SQL;                               *;
*  DESCRIBE TABLE DICTIONARY.<input-table>;        *;
*  SELECT *                                  *;
*    FROM DICTIONARY.<input-table>                    *;
*    WHERE expression                           *;
*    ORDER BY col-name <DESC>;                       *;
*  QUIT;                                      *;
*******************************************************;


*******************************************************;
*  Demo                                       *;
*  1) Open the s102d06.sas program in the demos folder    *;
*    and find the Demo section.                       *;
*  2) In the Explore dictionary.tables section:          *;
*    a) Highlight and run the procedure. Examine the log *;
*      and the results.                          *;
*    b) Add a WHERE clause to subset the Libname column  *;
*      for libraries named SQ and remove the INOBS=     *;
*      option. Highlight and run the procedure. Examine *;
*      the log and the results.                    *;
*      Note: The Libname and Memname columns are stored *;
*           in all uppercase.                     *;
*    c) Discuss the code for the SAS equivalent of      *;
*      DICTIONARY.tables. Highlight and run the        *;
*      procedure. Examine the log and the results.      *;
*  3) Move to the Explore dictionary.columns section.     *;
*    a) Highlight and run the procedure. Examine the log *;
*      and the results.                          *;
*    b) Modify the WHERE clause to subset the Name       *;
*      column by BankID and use the UPCASE function on  *;
*      the Name column. Highlight and run the          *;
*       procedure. Examine the log and the results.      *;
*     c) Discuss the code for the SAS equivalent of      *;
*       DICTIONARY.columns. Highlight and run the        *;
*       procedure. Examine the log and the results.      *;
*  4) Move to the Explore dictionary.libnames section.    *;
*     a) Highlight and run the procedure. Examine the log *;
*       and the results.                                 *;
*     b) Modify the SELECT clause by replacing the        *;
*       asterisks and adding the DISTINCT keyword on the *;
*       Libname column. Highlight and run the procedure. *;
*       Examine the log and the results.                 *;
*     c) Modify the SELECT clause by replacing distinct   *;
*       libname with an asterisk. Add a WHERE clause to  *;
*       subset the Libname column for the SQ library.    *;
*       Highlight and run the procedure. Examine the log *;
*       and the results.                                 *;
*     d) Discuss the code for the SAS equivalent of      *;
*       DICTIONARY.libnames. Highlight and run the       *;
*       procedure. Examine the log and the results.      *;
**********************************************************;

*******************************;
*EXPLORE DICTIONARY.TABLES     *;
*******************************;
proc sql inobs=100;
describe table dictionary.tables;
select *
        from dictionary.tables;
quit;
```

```
73          proc sql inobs=100;
74          describe table dictionary.tables;
NOTE: SQL table DICTIONARY.TABLES was created like:

create table DICTIONARY.TABLES
  (
   libname char(8) label='Library Name',
   memname char(32) label='Member Name',
   memtype char(8) label='Member Type',
   dbms_memtype char(32) label='DBMS Member Type',
   memlabel char(256) label='Data Set Label',
   typemem char(8) label='Data Set Type',
   crdate num format=DATETIME informat=DATETIME label='Date Created',
   modate num format=DATETIME informat=DATETIME label='Date Modified',
   nobs num label='Number of Physical Observations',
   obslen num label='Observation Length',
   nvar num label='Number of Variables',
   protect char(3) label='Type of Password Protection',
   compress char(8) label='Compression Routine',
   encrypt char(8) label='Encryption',
   npage num label='Number of Pages',
   filesize num label='Size of File',
   pcompress num label='Percent Compression',
   reuse char(3) label='Reuse Space',
   bufsize num label='Bufsize',
   delobs num label='Number of Deleted Observations',
   nlobs num label='Number of Logical Observations',
   maxvar num label='Longest variable name',
   maxlabel num label='Longest label',
   maxgen num label='Maximum number of generations',
```



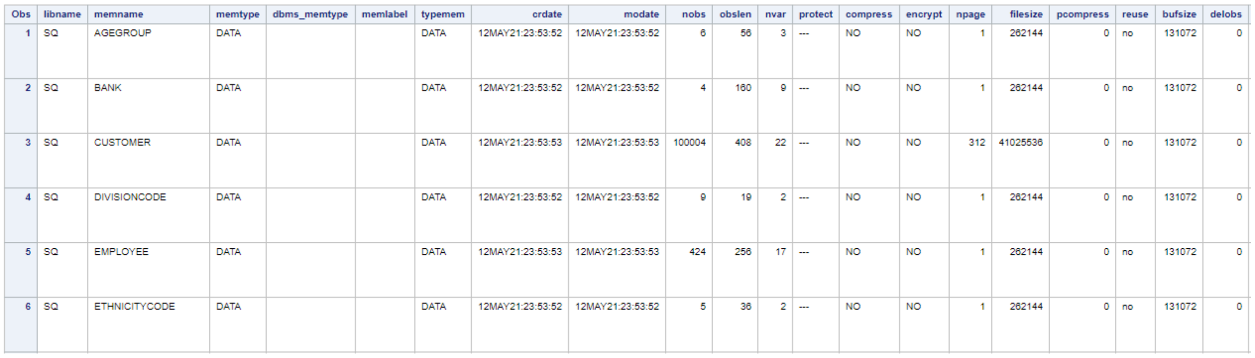
Library Name	Member Name	Member Type	DBMS Member Type	Data Set Label	Data Set Type	Date Created	Date Modified	Number of Physical Observations	Observation Length	Number of Variables	Type of Password Protection	Compression Routine
WORK	CUSTOMERCOUNT	DATA			DATA	30JUL19:10:36:34	30JUL19:10:36:34	51	24	3	---	NO
SQ	AGEGROUP	DATA			DATA	22JUL19:09:43:11	22JUL19:09:43:11	6	56	3	---	NO
SQ	BANK	DATA			DATA	22JUL19:09:43:11	22JUL19:09:43:11	4	160	9	---	NO
SQ	CUSTOMER	DATA			DATA	22JUL19:09:43:12	22JUL19:09:43:12	100004	408	22	---	NO
SQ	DIVISIONCODE	DATA			DATA	22JUL19:09:43:11	22JUL19:09:43:11	9	19	2	---	NO
SQ	EMPLOYEE	DATA			DATA	22JUL19:09:43:14	22JUL19:09:43:14	424	256	17	---	NO
SQ	ETHNICITYCODE	DATA			DATA	22JUL19:09:43:11	22JUL19:09:43:11	5	36	2	---	NO

```
proc sql;
select *
        from dictionary.tables
        where libname='SQ';
quit;
```



Library Name	Member Name	Member Type	DBMS Member Type	Data Set Label	Data Set Type	Date Created	Date Modified	Number of Physical Observations	Observation Length	Number of Variables	Type of Password Protection	Compression Routine	Encryption	Number of Pages	Size of File	Percent Compression	Reuse Space	Bufsize
SQ	AGEGROUP	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	6	56	3	---	NO	NO	1	262144	0	no	131072
SQ	BANK	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	4	160	9	---	NO	NO	1	262144	0	no	131072
SQ	CUSTOMER	DATA			DATA	12MAY21:23:53:53	12MAY21:23:53:53	100004	408	22	---	NO	NO	312	41025536	0	no	131072
SQ	DIVISIONCODE	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	9	19	2	---	NO	NO	1	262144	0	no	131072
SQ	EMPLOYEE	DATA			DATA	12MAY21:23:53:53	12MAY21:23:53:53	424	256	17	---	NO	NO	1	262144	0	no	131072
SQ	ETHNICITYCODE	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	5	36	2	---	NO	NO	1	262144	0	no	131072

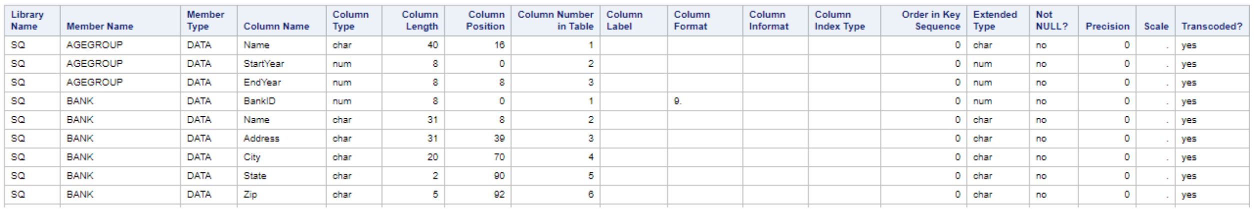
```
/*SAS Equivalent of dictionary.tables*/
proc print data=sashelp.vtable;
        where Libname = "SQ";
run;
```



Obs	libname	memname	memtype	dbms_memtype	memlabel	typemem	crdate	modate	nobs	obslen	nvar	protect	compress	encrypt	npage	filesize	pcompress	reuse	bufsize	delobs
1	SQ	AGEGROUP	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	6	56	3	---	NO	NO	1	262144	0	no	131072	0
2	SQ	BANK	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	4	160	9	---	NO	NO	1	262144	0	no	131072	0
3	SQ	CUSTOMER	DATA			DATA	12MAY21:23:53:53	12MAY21:23:53:53	100004	408	22	---	NO	NO	312	41025536	0	no	131072	0
4	SQ	DIVISIONCODE	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	9	19	2	---	NO	NO	1	262144	0	no	131072	0
5	SQ	EMPLOYEE	DATA			DATA	12MAY21:23:53:53	12MAY21:23:53:53	424	256	17	---	NO	NO	1	262144	0	no	131072	0
6	SQ	ETHNICITYCODE	DATA			DATA	12MAY21:23:53:52	12MAY21:23:53:52	5	36	2	---	NO	NO	1	262144	0	no	131072	0

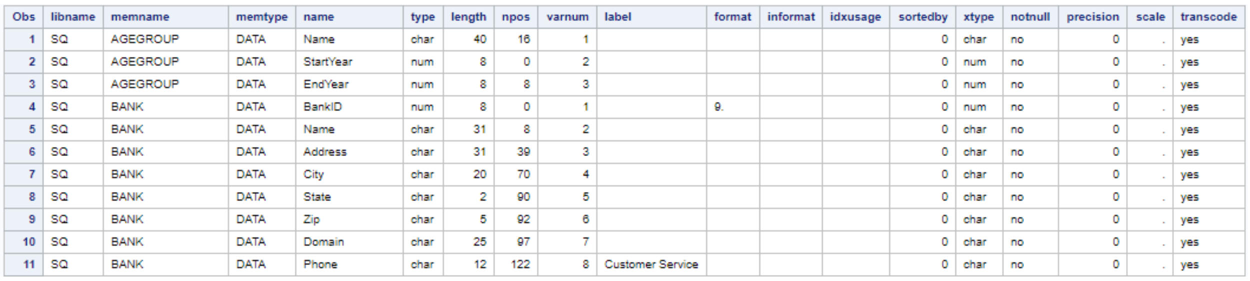
```
******************************;
*EXPLORE DICTIONARY.COLUMNS    *;
******************************;
proc sql;
describe table dictionary.columns;
select *
        from dictionary.columns
        where Libname = "SQ";
quit;
```

```
 73          proc sql;
 74          describe table dictionary.columns;
 NOTE: SQL table DICTIONARY.COLUMNS was created like:

 create table DICTIONARY.COLUMNS
   (
    libname char(8) label='Library Name',
    memname char(32) label='Member Name',
    memtype char(8) label='Member Type',
    name char(32) label='Column Name',
    type char(4) label='Column Type',
    length num label='Column Length',
    npos num label='Column Position',
    varnum num label='Column Number in Table',
    label char(256) label='Column Label',
    format char(49) label='Column Format',
    informat char(49) label='Column Informat',
    idxusage char(9) label='Column Index Type',
    sortedby num label='Order in Key Sequence',
    xtype char(12) label='Extended Type',
    notnull char(3) label='Not NULL?',
    precision num label='Precision',
    scale num label='Scale',
    transcode char(3) label='Transcoded?',
    diagnostic char(256) label='Diagnostic Message from File Open Attempt'
   );
```



Library Name	Member Name	Member Type	Column Name	Column Type	Column Length	Column Position	Column Number in Table	Column Label	Column Format	Column Informat	Column Index Type	Order in Key Sequence	Extended Type	Not NULL?	Precision	Scale	Transcoded?
SQ	AGEGROUP	DATA	Name	char	40	16	1					0	char	no	0	.	yes
SQ	AGEGROUP	DATA	StartYear	num	8	0	2					0	num	no	0	.	yes
SQ	AGEGROUP	DATA	EndYear	num	8	8	3					0	num	no	0	.	yes
SQ	BANK	DATA	BankID	num	8	0	1		9.			0	num	no	0	.	yes
SQ	BANK	DATA	Name	char	31	8	2					0	char	no	0	.	yes
SQ	BANK	DATA	Address	char	31	39	3					0	char	no	0	.	yes
SQ	BANK	DATA	City	char	20	70	4					0	char	no	0	.	yes
SQ	BANK	DATA	State	char	2	90	5					0	char	no	0	.	yes
SQ	BANK	DATA	Zip	char	5	92	6					0	char	no	0	.	yes

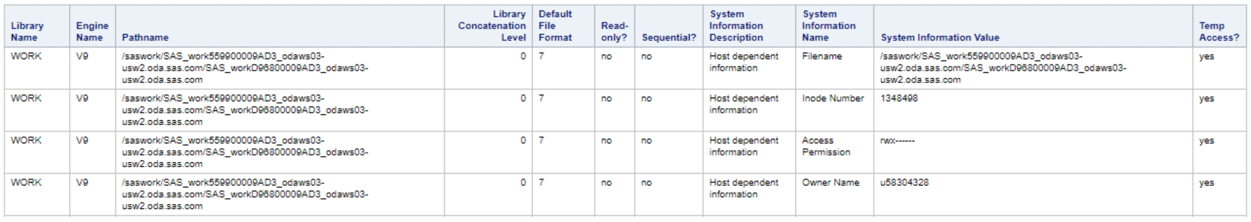
```
/*SAS Equivalent of dictionary.columns*/

proc print data=sashelp.vcolumn(obs=100);

        where Libname = "SQ";

run;
```



Obs	libname	memname	memtype	name	type	length	npos	varnum	label	format	informat	idxusage	sortedby	xtype	notnull	precision	scale	transcode
1	SQ	AGEGROUP	DATA	Name	char	40	16	1					0	char	no	0	.	yes
2	SQ	AGEGROUP	DATA	StartYear	num	8	0	2					0	num	no	0	.	yes
3	SQ	AGEGROUP	DATA	EndYear	num	8	8	3					0	num	no	0	.	yes
4	SQ	BANK	DATA	BankID	num	8	0	1		9.			0	num	no	0	.	yes
5	SQ	BANK	DATA	Name	char	31	8	2					0	char	no	0	.	yes
6	SQ	BANK	DATA	Address	char	31	39	3					0	char	no	0	.	yes
7	SQ	BANK	DATA	City	char	20	70	4					0	char	no	0	.	yes
8	SQ	BANK	DATA	State	char	2	90	5					0	char	no	0	.	yes
9	SQ	BANK	DATA	Zip	char	5	92	6					0	char	no	0	.	yes
10	SQ	BANK	DATA	Domain	char	25	97	7					0	char	no	0	.	yes
11	SQ	BANK	DATA	Phone	char	12	122	8	Customer Service				0	char	no	0	.	yes

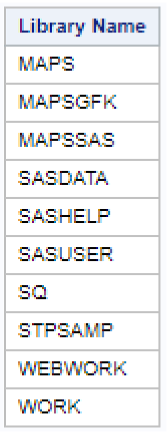
```
********************************;

*EXPLORE DICTIONARY.LIBNAMES   *;

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

proc sql;

describe table dictionary.libnames;

select *

        from dictionary.libnames;

quit;
```

```
 76          proc sql;
 77          describe table dictionary.libnames;
 NOTE: SQL table DICTIONARY.LIBNAMES was created like:
 
 create table DICTIONARY.LIBNAMES
   (
    libname char(8) label='Library Name',
    engine char(8) label='Engine Name',
    path char(1024) label='Pathname',
    level num label='Library Concatenation Level',
    fileformat char(8) label='Default File Format',
    readonly char(3) label='Read-only?',
    sequential char(3) label='Sequential?',
    sysdesc char(1024) label='System Information Description',
    sysname char(1024) label='System Information Name',
    sysvalue char(1024) label='System Information Value',
    temp char(3) label='Temp Access?'
   );
```



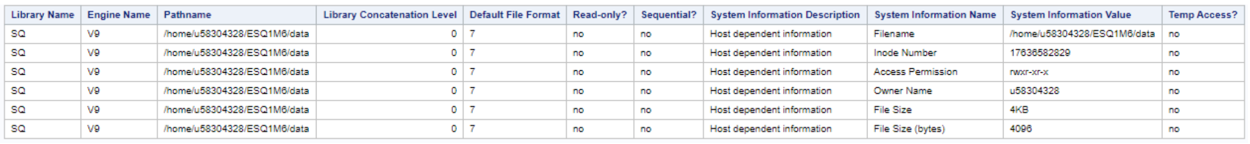
Library Name	Engine Name	Pathname	Library Concatenation Level	Default File Format	Read-only?	Sequential?	System Information Description	System Information Name	System Information Value	Temp Access?
WORK	V9	/saswork/SAS_work559900009AD3_odaws03-usw2.oda.sas.com/SAS_workD9680009AD3_odaws03-usw2.oda.sas.com	0	7	no	no	Host dependent information	Filename	/saswork/SAS_work559900009AD3_odaws03-usw2.oda.sas.com/SAS_workD9680009AD3_odaws03-usw2.oda.sas.com	yes
WORK	V9	/saswork/SAS_work559900009AD3_odaws03-usw2.oda.sas.com/SAS_workD9680009AD3_odaws03-usw2.oda.sas.com	0	7	no	no	Host dependent information	Inode Number	1348498	yes
WORK	V9	/saswork/SAS_work559900009AD3_odaws03-usw2.oda.sas.com/SAS_workD9680009AD3_odaws03-usw2.oda.sas.com	0	7	no	no	Host dependent information	Access Permission	rwx------	yes
WORK	V9	/saswork/SAS_work559900009AD3_odaws03-usw2.oda.sas.com/SAS_workD9680009AD3_odaws03-usw2.oda.sas.com	0	7	no	no	Host dependent information	Owner Name	u58304328	yes

```
proc sql;

describe table dictionary.libnames;

select distinct libname

        from dictionary.libnames;

quit;
```



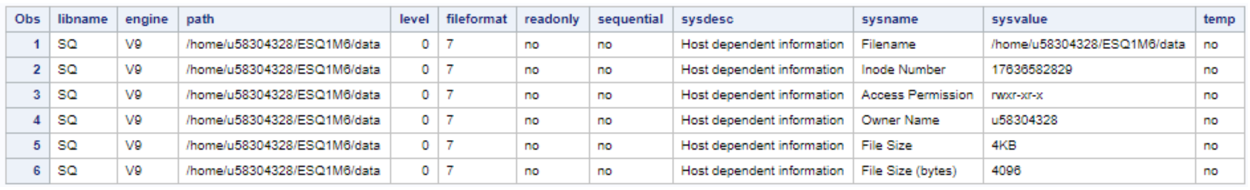
Library Name
MAPS
MAPSGFK
MAPSSAS
SASDATA
SASHELP
SASUSER
SQ
STPSAMP
WEBWORK
WORK

```
proc sql;

describe table dictionary.libnames;

select *

        from dictionary.libnames

        where libname='SQ';

quit;
```



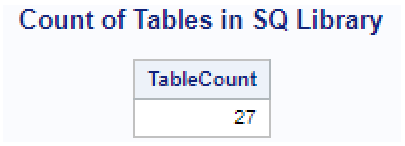
Library Name	Engine Name	Pathname	Library Concatenation Level	Default File Format	Read-only?	Sequential?	System Information Description	System Information Name	System Information Value	Temp Access?
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Filename	/home/u58304328/ESQ1M6/data	no
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Inode Number	17636582829	no
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Access Permission	rwxr-xr-x	no
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Owner Name	u58304328	no
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	File Size	4KB	no
SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	File Size (bytes)	4096	no

```
/*SAS Equivalent of dictionary.members*/

proc print data=sashelp.vlibnam;

   where Libname = "SQ";

run;
```



Obs	libname	engine	path	level	fileformat	readonly	sequential	sysdesc	sysname	sysvalue	temp
1	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Filename	/home/u58304328/ESQ1M6/data	no
2	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Inode Number	17636582829	no
3	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Access Permission	rwxr-xr-x	no
4	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	Owner Name	u58304328	no
5	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	File Size	4KB	no
6	SQ	V9	/home/u58304328/ESQ1M6/data	0	7	no	no	Host dependent information	File Size (bytes)	4096	no

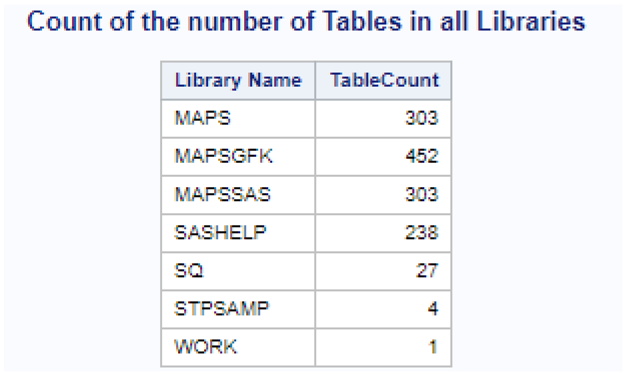
```
/*Practice Level 1: Counting the Number of Tables in a Library

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 query to create a report that displays the count of the number of tables in the SQ library.

Use DICTIONARY.tables as input.

Name the calculated column TableCount.

Add an appropriate title.

Run the program and view the results.

How many tables are in the sq library?

*/

title "Count of Tables in SQ Library";

proc sql;

describe table dictionary.tables;

select count(*) as TableCount

            from dictionary.tables

            where libname='SQ';


quit;

title;
```



```
Count of Tables in SQ Library

TableCount
        27
```

```
/*s102s08.sas*/


title 'Count of SQ Tables';

proc sql;

select count(*) as TableCount

   from dictionary.tables

   where libname='SQ';
```

```
quit;

title;


/*Practice Level 2: Counting the Number of Tables in All Libraries

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 query to create a report that displays the count of the number of tables in all libraries.

Use DICTIONARY.tables as input.

Name the calculated column TableCount.

Group the results by the library name.

Add an appropriate title and display the library name and table count.

Run the program and view the results.

Which library has the most tables?

Note: This is a free response question and all attempts receive credit. Type your response and compare
your answer to the answer provided.

*/

title "Count of the number of Tables in all Libraries";

proc sql;

describe table dictionary.tables;

select libname, count(*) as TableCount

        from dictionary.tables

        group by libname;

quit;

title;
```



Count of the number of Tables in all Libraries

Library Name	TableCount
MAPS	303
MAPSGFK	452
MAPSSAS	303
SASHELP	238
SQ	27
STPSAMP	4
WORK	1