

A Quick Comparison of SQL and SAS Coding Techniques for Basic Programming Tasks

A Comparative Study Using Examples
from SAS Programming I: Essentials

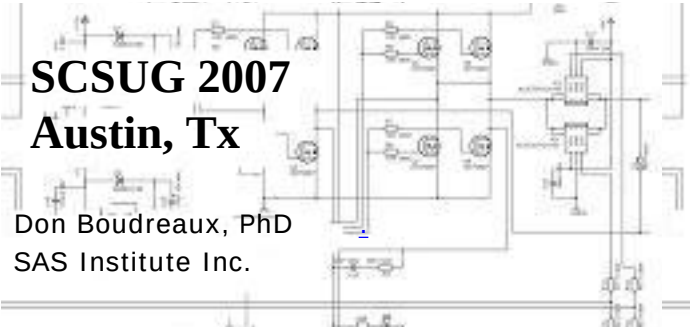
(c) 2007 - Don Boudreaux, PhD

A significant number of the Prog1 students coming to SAS for training will have some experience with SQL. Unlike the data step and proc step code shown in the course notes, SQL is a declarative language with a rather different type of syntax. In an effort to "ease the pain" for these students, I have taken a number of the syntax code segments shown in Prog1 and provided an SQL equivalent. The intent is to display SAS and SQL code "side by side" (see example below) in order to facilitate an SQL coder more quickly understanding and writing SAS code. This is not intended to be an SQL tutorial.

SAS Code	SQL Statement(s)
<pre>proc print data=work.staff ; run ;</pre>	<pre>proc SQL ; select * from work.staff ; quit ;</pre>

In most cases the exact same results will be generated, but not in every case. I did modify a few of the SAS code segments to keep the match between the SAS results and the SQL results as close as possible. In addition, I left off a number of the proc print steps after the first couple of segments - as repeatedly comparing them seemed redundant. Also, note that many of the code segments assume that a libname statement has already been submitted to define the ia library. The default required libname used in the course notes is:

```
libname ia "c:\workshop\winsas\prog1" ;
```



SCSUG 2007 Austin, Tx

Don Boudreaux, PhD
SAS Institute Inc.

[Intro](#)[Chapt2](#)[Chapt3](#)[Chapt4](#)[Chapt5](#)[Chapt6](#)[Chapt7](#)[Chapt8](#)[Chapt9](#)[Title Page](#)[Introduction](#)

For the most part, the basic listing, means, and frequency reports will "look and feel" similar. The two languages will have a different syntax, but the results will be the same for the specific examples in this chapter. However, the data step code does indicate a very different capability - the ability to read in text files. The SQL code examples in this section mimic reading in the same data using insert statements.

SAS Code	SQL Statement(s)
<pre>proc means data=work.staff ; var Salary ; class JobTitle ; run ;</pre>	<pre>proc SQL ; select JobTitle, count(*) as NObs, n(Salary) as n, avg(Salary) as mean, std(Salary) as stdDev, min(Salary) as minimum, max(Salary) as maximum from work.staff group by JobTitle ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>data work.staff ; infile 'emplist.dat' ; input LastName \$ 1-20 FirstName \$ 21-30 JobTitle \$ 36-43 Salary 54-59 ; run ;</pre>	<pre>proc SQL ; create table work.staff (LastName char(20), FirstName char(10), JobTitle char(8), Salary float) ; insert into work.staff values ('TORRES', 'JAN', 'Pilot', 50000) values ('LANGKAMM','SARAH', 'Mechanic',80000) values ('SMITH', 'MICHAEL', 'Mechanic',40000) values ('LEISTNER','COLIN', 'Mechanic',36000) values ('WADE', 'KIRSTEN', 'Pilot', 85000) ; quit ; ** note1: SQL does not read text files ; ** note2: example only inserting obs 1-5 ;</pre>

SAS Code	SQL Statement(s)
<pre>proc print data=work.staff ; run ;</pre>	<pre>proc SQL ; select * from work.staff ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>** shows descriptor portion ; proc contents data=work.staff ; run ;</pre>	<pre>proc SQL ; describe table work.staff ; quit ;</pre>
<pre>** shows data portion ; proc print data=work.staff ; run ;</pre>	<pre>proc SQL ; select * from work.staff ; quit ;</pre>

The libname statement is required to access folders that contain SAS datasets. For the purposes of comparing SQL and SAS code it will just be considered a requirement of the environment. It is also the case that both languages have a rather simple syntax for obtaining information about the descriptor portion of a single dataset. But, one advantage shown for the use of SAS code is the ability of proc contents to investigate a whole library at one time with use of the `_all_` keyword.

SAS Code	SQL Statement(s)
<pre>libname ia "c:\workshop\winsas\prog1" ;</pre>	
<pre>** to see all datasets in ia ; proc contents data=ia._all_ nods ; run ; ** to see just one dataset ; proc contents data=ia.crew ; run ;</pre>	<pre>proc SQL ; select memname format=\$16., nobs, nvar, crdate from dictionary.tables where libname='IA' ; describe table ia.crew ; quit ;</pre>

While basic listings are very similar for both languages, proc print will have an advantage in summing columns, by processing, and its pageby capability. The SQL examples can mimic the summed column totals and by processing with multiple selects and use of the outer union set operator. The pageby can be mimicked using separate SQL requests. But both of these cases involve "manually" working with the sums and classification groupings. Note, these are additional rows that are attached to a listing - they are not additional columns that could be generated using SQL group by clauses.

In typical usage the where processing is almost exactly the same for the two languages. The most notable exception is the addition, within SQL, of the ESCAPE keyword for use with the LIKE operator [1]. This keyword allows the user to specify a search for a literal underscore or percent sign. In the following example, the underscore is used simply as another text piece like the 'F' or the 'A' because it has been set as an escape character. The normal meaning of the underscore to take the place of a wildcard character is lost.

```
proc SQL ;
select    *
from      work.staff
where     jobcode like 'FA/_%' ESCAPE '/'
;
quit ;
```

Another advantage in SQL is the ability to list a variable multiple times using different format and/or label specifications for each occurrence.

```
proc SQL ;
select    startdate ,
          startdate format=mmddyy10. ,
          startdate format=date9.
from      work.look
;
```

```
quit ;
```

SAS Code	SQL Statement(s)
<pre>proc print data=ia.empdata noobs ; var JobCode EmpID Salary ; where JobCode='PILOT' ; run ;</pre>	<pre>proc SQL ; select JobCode, EmpID, Salary from ia.empdata where JobCode='PILOT' ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>proc print data=ia.empdata noobs ; var JobCode EmpID Salary ; sum Salary ; run ;</pre>	<pre>proc SQL ; select JobCode, EmpID, Salary from ia.empdata outer union corr select sum(Salary) as Salary from ia.empdata ; quit ; ** sums added in "by hand" ; ** with concatenation ;</pre>

SAS Code	SQL Statement(s)
<pre>proc print data=ia.empdata ; run ;</pre>	<pre>proc SQL number ; select * from ia.empdata ; quit ;</pre>
<pre>proc print data=ia.empdata ; var JobCode EmpID Salary ; run ;</pre>	<pre>proc SQL number ; select JobCode, EmpID, Salary from ia.empdata ; quit ;</pre>
<pre>proc print data=ia.empdata noobs ; var JobCode EmpID Salary ; run ;</pre>	<pre>proc SQL ; select JobCode, EmpID, Salary from ia.empdata ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> proc sort data=ia.empdata out=work.empdata ; by JobCode ; run ; proc print data=work.empdata ; var JobCode EmpID Salary ; by JobCode ; sum Salary ; run ; </pre>	<pre> proc SQL ; select JobCode, EmpID, Salary from ia.empdata where JobCode='FLTAT' outer union corr select sum(Salary) as Salary from ia.empdata where JobCode='FLTAT' outer union corr select JobCode, EmpID, Salary from ia.empdata where JobCode='PILOT' outer union corr select sum(Salary) as Salary from ia.empdata where JobCode='PILOT' outer union corr select sum(Salary) as Salary from ia.empdata ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> proc sort data=ia.empdata out=work.empdata ; by JobCode ; run ; proc print data=work.empdata ; var JobCode EmpID Salary ; by JobCode ; pageby JobCode ; sum Salary ; run ; </pre>	<pre> proc SQL ; select JobCode, EmpID, Salary from ia.empdata where JobCode='FLTAT' outer union corr select sum(Salary) as Salary from ia.empdata where JobCode='FLTAT' ; select JobCode, EmpID, Salary from ia.empdata where JobCode='PILOT' outer union corr select sum(Salary) as Salary from ia.empdata where JobCode='PILOT' ; quit ; </pre>

The labels and formats used within proc SQL are all standard SAS code capabilities and are not specified with the ANSI standard for SQL. This would also apply to any of the other global statements in SAS - title, footnote, filename, libname, and options.

SAS Code	SQL Statement(s)
<pre>title1 'Salary Report' ; proc print data=ia.empdata split=' ' ; var EmpID LastName FirstName JobCode Salary ; format Salary dollar11.2 ; label LastName='Last Name' FirstName='First Name' Salary='Annual Salary' ; run ;</pre>	<pre>proc SQL ; title1 'Salary Report' ; select EmpID, LastName label='Last Name' , FirstName label='First Name' , JobCode, Salary label='Annual Salary' format=dollar11.2 from ia.empdata ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>ods pdf file='c05s3d1m.pdf' ; title1 'Salary Report' ; proc print data=ia.empdata label noobs ; var EmpID JobCode Salary ; label Salary='Annual Salary' ; format Salary dollar11.2 ; run ; ods pdf close ;</pre>	<pre>ods pdf file='c05s3d1m.pdf' ; proc SQL ; title1 'Salary Report' ; select EmpID, JobCode, Salary label='Annual Salary' format=dollar11.2 from ia.empdata ; quit ; ods pdf close ;</pre>

SAS Code	SQL Statement(s)
<pre>ods html file='c05s3d1m.html' ; title1 'Salary Report' ; proc print data=ia.empdata label noobs ; var EmpID JobCode Salary ; label Salary='Annual Salary' ; format Salary dollar11.2 ; run ; ods html close ;</pre>	<pre>ods html file='c05s3d1m.html' ; proc SQL ; title1 'Salary Report' ; select EmpID, JobCode, Salary label='Annual Salary' format=dollar11.2 from ia.empdata ; quit ; ods html close ;</pre>

As mentioned in the Chapter 2 section, SQL can not read text files. It is only mimicked here using values or set clauses within insert statements. This would be a real limitation in dealing with alternate data sources.

As mentioned in the Chapter 5 section, the labels and formats used within proc SQL are all standard SAS code capabilities and are not specified with the ANSI standard for SQL. This would also apply to any of the other global statements in SAS - title, footnote, filename, libname, and options.

SAS Code	SQL Statement(s)
<pre>data work.dfwlax ; infile 'dfwlax.dat' ; input Flight \$ 1-3 Date \$ 4-11 Dest \$ 12-14 FirstClass 15-17 Economy 18-20 ; run ; proc print data=work.dfwlax ; run ;</pre>	<pre>proc SQL ; create table work.dfwlax (Flight char(3), Date char(8), Dest char(3), FirstClass float, Economy float) ; insert into work.dfwlax values ('439', '12/11/00', 'LAX', 20, 137) values ('921', '12/11/00', 'DFW', 20, 131) values ('114', '12/12/00', 'LAX', 15, 170) values ('982', '12/12/00', 'dfw', 5, 85) values ('439', '12/12/00', 'LAX', 14, 196) ; select * from work.dfwlax ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>data ia.dfwlax ; infile 'dfwlax.dat' ; input Flight \$ 1-3 Date \$ 4-11 Dest \$ 12-14 FirstClass 15-17 Economy 18-20 ; run ; proc print data=ia.dfwlax ; run ;</pre>	<pre>proc SQL ; create table ia.dfwlax (Flight char(3), Date char(8), Dest char(3), FirstClass float, Economy float) ; insert into ia.dfwlax values ('439', '12/11/00', 'LAX', 20, 137) values ('921', '12/11/00', 'DFW', 20, 131) values ('114', '12/12/00', 'LAX', 15, 170) values ('982', '12/12/00', 'dfw', 5, 85) values ('439', '12/12/00', 'LAX', 14, 196) ; select * from ia.dfwlax ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data work.dfwlax ; infile 'dfwlax.dat' ; input @1 Flight \$3. @4 Date mmddyy8. @12 Dest \$3. @15 FirstClass 3. @18 Economy 3.; run ; proc print data=work.dfwlax ; run ; </pre>	<pre> proc SQL ; create table work.dfwlax (Flight char(3), Date date, Dest char(3), FirstClass float, Economy float) ; insert into work.dfwlax values ('439', '11DEC2000'd, 'LAX', 20, 137) values ('921', '11DEC2000'd, 'DFW', 20, 131) values ('114', '12DEC2000'd, 'LAX', 15, 170) values ('982', '12DEC2000'd, 'dfw', 5, 85) values ('439', '12DEC2000'd, 'LAX', 14, 196) ; select * from work.dfwlax ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre>proc print data=work.dfwlax ; format Date date9. ; run ;</pre>	<pre>proc SQL ; select Flight , Date format=date9. , Dest , FirstClass , Economy from work.dfwlax ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data ia.dfwlax ; infile 'dfwlax.dat' ; input @1 Flight \$3. @4 Date mmddyy8. @12 Dest \$3. @15 FirstClass 3. @18 Economy 3.; run ; proc contents data=ia.dfwlax ; run ; *** dataset created with just ; *** name, type, length attributes ; </pre>	<pre> proc SQL ; create table ia.dfwlax (Flight char(3), Date date, Dest char(3), FirstClass float, Economy float) ; insert into ia.dfwlax values ('439', '11DEC2000'd, 'LAX', 20, 137) values ('921', '11DEC2000'd, 'DFW', 20, 131) values ('114', '12DEC2000'd, 'LAX', 15, 170) values ('982', '12DEC2000'd, 'dfw', 5, 85) values ('439', '12DEC2000'd, 'LAX', 14, 196) ; describe table ia.dfwlax ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre>proc print data=ia.dfwlax label ; var Flight Date Dest FirstClass Economy ; format Date mmddyy10. ; label Dest='Destination' FirstClass='First Class Passengers' Economy='Economy Passengers' ; run;</pre>	<pre>proc SQL ; select Flight, Date format=mmddyy10. , Dest label='Destination' , FirstClass label='First Class Passengers' , Economy label='Economy Passengers' from ia.dfwlax ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data ia.dfwlax ; infile 'dfwlax.dat' ; input @1 Flight \$3. @4 Date mmddyy8. @12 Dest \$3. @15 FirstClass 3. @18 Economy 3.; format Date mmddyy10. ; label Dest='Destination' FirstClass='First Class Passengers' Economy='Economy Passengers' ; run ; </pre>	<pre> proc SQL ; create table ia.dfwlax (Flight char(3), Date date format=mmddyy10. , Dest char(3) label='Destination' , FirstClass float label='First Class Passengers' , Economy float label='Economy Passengers') ; insert into ia.dfwlax values ('439', '11DEC2000'd, 'LAX', 20, 137) values ('921', '11DEC2000'd, 'DFW', 20, 131) values ('114', '12DEC2000'd, 'LAX', 15, 170) values ('982', '12DEC2000'd, 'dfw', 5, 85) values ('439', '12DEC2000'd, 'LAX', 14, 196) ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre>* check before ; proc contents data=ia.dfwlax ; run ; proc datasets library=ia ; modify dfwlax ; rename Dest=Destination ; quit ; * check after ; proc contents data=ia.dfwlax ; run ;</pre>	<pre>proc SQL ; describe table ia.dfwlax ; create table ia.dfwlax as select Flight, Date, Dest as Destination , FirstClass, Economy from ia.dfwlax ; describe table ia.dfwlax ; quit ;</pre>

A major syntax difference does however show up then conditional processing is needed. In SQL conditional processing is accomplished using a case/when syntax as opposed to the if/then/else syntax seen in the datastep. There is a major advantage in the SAS code in that do/end blocks can be inserted to process multiple statements at the same time. SQL does not possess this capability.

SQL, as a declarative language, also lacks internal do loops, subsetting ifs, first-dot/last-dot processing, the ability to create multiple datasets, or the ability to write text files (although these topics are normally not covered within the scope of the examples shown here from Programming 1).

SAS Code	SQL Statement(s)
<pre>data onboard ; set ia.dfwlax ; Total = sum(FirstClass, Economy) ; drop FirstClass Economy ; run ;</pre>	<pre>proc SQL ; create table work.onboard as select Flight, Date, Dest, sum(FirstClass, Economy) as Total from ia.dfwlax ; quit ; ** varaibales can be used in calculations ; ** without being put into the table ;</pre>

SAS Code	SQL Statement(s)
<pre>data onboard ; set ia.dfwlax ; Total = FirstClass+Economy ; run ;</pre>	<pre>proc SQL ; create table work.onboard as select *, FirstClass+Economy as Total from ia.dfwlax ; quit ;</pre>
<pre>data onboard ; set ia.dfwlax ; Total = sum(FirstClass, Economy) ; run ;</pre>	<pre>proc SQL ; create table work.onboard as select *, sum(FirstClass, Economy) as Total from ia.dfwlax ; quit ;</pre>
<pre>data onboard ; set ia.dfwlax ; Total = sum(FirstClass, Economy) ; DayOfWeek = weekday(Date) ; run ;</pre>	<pre>proc SQL ; create table work.onboard as select *, sum(FirstClass, Economy) as Total, weekday(Date) as DayOfWeek from ia.dfwlax ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>data flightrev ; set ia.dfwlax ; Total=sum(FirstClass,Economy) ; if Dest='LAX' then Revenue=sum(2000*FirstClass,1200*Economy) ; else if Dest='DFW' then Revenue=sum(1500*FirstClass,900*Economy) ; run ; proc print data=flightrev ; format Date date9. ; run ;</pre>	<pre>proc SQL ; create table work.flightrev as select *, sum(FirstClass, Economy) as Total, case Dest when 'LAX' then sum(2000*FirstClass,1200*Economy) when 'DFW' then sum(1500*FirstClass, 900*Economy) else . end as Revenue from ia.dfwlax ; select * from work.flightrev ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data flightrev ; set ia.dfwlax ; Total=sum(FirstClass,Economy) ; if upcase(Dest)='LAX' then Revenue=sum(2000*FirstClass,1200*Economy) ; else if upcase(Dest)='DFW' then Revenue=sum(1500*FirstClass,900*Economy) ; run ; proc print data=flightrev ; format Date date9. ; run ; </pre>	<pre> proc SQL ; create table work.flightrev as select *, sum(FirstClass, Economy) as Total, case upcase(Dest) when 'LAX' then sum(2000*FirstClass,1200*Economy) when 'DFW' then sum(1500*FirstClass, 900*Economy) else . end as Revenue from ia.dfwlax ; select * from work.flightrev ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> data flightrev ; set ia.dfwlax ; Total=sum(FirstClass,Economy) ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.flightrev as select *, sum(FirstClass, Economy) as Total, case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue , case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax ; quit ; ** there is no do...end construct in SQL ; ** must use multiple case requests ; </pre>

SAS Code	SQL Statement(s)
<pre> data flightrev ; set ia.dfwlax ; length City \$ 11 ; Total=sum(FirstClass,Economy) ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.flightrev as select *, sum(FirstClass, Economy) as Total, case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue, case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax ; quit ; ** no need to preset the length ; ** for fields created in SQL ; </pre>

SAS Code	SQL Statement(s)
<pre> data over175 ; set ia.dfwlax ; length City \$ 11 ; Total=sum(FirstClass,Economy) ; if Total LE 175 then delete ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.over175 as select *, sum(FirstClass, Economy) as Total , case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue, case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax where CALCULATED Total > 175 ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> data over175 ; set ia.dfwlax ; length City \$ 11 ; Total=sum(FirstClass,Economy) ; if Total GT 175 ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.over175 as select *, sum(FirstClass, Economy) as Total , case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue, case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax where CALCULATED Total > 175 ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> data over175 ; set ia.dfwlax ; length City \$ 11 ; Total=sum(FirstClass,Economy) ; if Total GT 175 AND Date LE '14dec2000'd ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.over175 as select *, sum(FirstClass, Economy) as Total , case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue, case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax where CALCULATED Total > 175 AND Date LE '14dec2000'd ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> data over175 ; set ia.dfwlax ; where Date LE '14dec2000'd ; length City \$ 11 ; Total=sum(FirstClass,Economy) ; if Total GT 175 ; if upcase(Dest)='DFW' then do ; Revenue=sum(1500*FirstClass,900*Economy) ; City='Dallas' ; end ; else if upcase(Dest)='LAX' then do ; Revenue=sum(2000*FirstClass,1200*Economy) ; City='Los Angeles' ; end ; run ; </pre>	<pre> proc SQL ; create table work.over175 as select *, sum(FirstClass, Economy) as Total , case upcase(Dest) when 'DFW' then sum(1500*FirstClass, 900*Economy) when 'LAX' then sum(2000*FirstClass,1200*Economy) else . end as Revenue, case upcase(Dest) when 'DFW' then 'Dallas' when 'LAX' then 'Los Angeles' else ' ' end as City from ia.dfwlax where CALCULATED Total > 175 AND Date LE '14dec2000'd ; quit ; </pre>

Concatenating datasets using SQL and SAS only involves a different syntax and terminology. A set with multiple dataset names in SAS code is exactly the same as a SQL request using the outer union corr set operator. What is lost with SAS code is the use of the other SQL set operators - intersect, except, and union. What is lost with the SQL code is the ability to effectively control interleaving.

Merging is the same for both languages on simple 1-to-none, 1-to-1, and 1-to-many processing using a merge with a by within a dataset versus using a full join in SQL. SQL however also allows for a very simple inner join, left join, and right join. Which can be mimicked in a datstep using if/then processing with the in= option. Unfortunately, when many-to-many relationships are present SQL will always generate a Cartesian product of the matching rows - a result that a simple dataset merge will not generate.

SQL does however allow the user one huge advantage over the dataset merge in that it does not require the matching to be an equivalence on the same variable. SQL joins are done with a where clause not a matching by criteria. Consider the following example:

```
proc SQL ;
select    ds1.x,
          ds1.y,
          ds2.z
from      work.ds1, work.ds2
where     ds1.transaction_date < ds2.check_date and
          ds1.id = ds2.id
;
quit ;
```

SAS Code	SQL Statement(s)
<pre>data newhires ; set na1 na2 ; run ; proc print data=newhires ; run ;</pre>	<pre>proc SQL ; create table work.newhires as select * from work.na1 outer union corr select * from work.na2 ; select * from work.newhires ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data na2 ; length Name \$ 6 Gender \$ 1 JCode \$ 3 ; input Name \$ Gender \$ JCode ; cards ; LISTER M NA2 TORRES F NA2 ; ** name JCode replaces JobCode in na2 ; ** but JobCode still in dataset na1 ; </pre>	<pre> proc SQL ; create table work.na2 (Name char(6), Gender char(1), JCode char(3)) ; insert into work.na2 values ('LISTER', 'M', 'NA2') values ('TORRES', 'F', 'NA2') ; quit ; </pre>
<pre> data newhires ; set na1 na2 ; run ; </pre>	<pre> proc SQL ; create table work.newhires as select * from work.na1 outer union corr select * from work.na2 ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre>data newhires ; set na1 na2(rename=(JCode=JobCode)) ; run ; proc print data=newhires ; run ;</pre>	<pre>proc SQL ; create table work.newhires as select Name, Gender, JobCode from work.na1 outer union corr select Name, Gender, JCode as JobCode from work.na2 ; select * from work.newhires ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>proc sort data=ia.miamiemp ; by ID ; run ; proc sort data=ia.parisemp ; by ID ; run ; proc sort data=ia.romeemp ; by ID ; run ; data work.allemp ; set ia.miamiemp ia.parisemp ia.romeemp ; by ID ; run ;</pre>	<pre>proc SQL ; create table work.allemp as select * from ia.miamiemp outer union corr select * from ia.parisemp outer union corr select * from ia.romeemp order by ID ; quit ; ** the results will match but there is no ; ** way to specify using interleaving in SQL ;</pre>

SAS Code	SQL Statement(s)
<pre>proc sort data=fa1 ; by Name ; run ; proc sort data=fa2 ; by Name ; run ; data newfa ; set fa1 fa2(rename=(JCode=JobCode)) ; by Name ; run ;</pre>	<pre>proc SQL ; create table work.newfa as select Name, Gender, JobCode from ia.fa1 outer union corr select Name, Gender, JCode as JobCode from ia.fa2 order by Name ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre> data fa1 ; length Name \$ 6 Gender \$ 1 JobCode \$ 3 ; input Name \$ Gender \$ JobCode ; cards ; KENT F FA1 PATEL M FA1 JONES F FA1 ; data fa2; length Name \$ 6 JCode \$ 3 Gender \$ 1 ; input Name \$ JCode \$ Gender ; cards; LISTER FA2 F TORRES FA2 F ; </pre>	<pre> proc SQL ; create table work.fa1 (Name char(6), Gender char(1), JobCode char(3)) ; insert into work.fa1 values ('KENT' , 'F', 'FA1') values ('PATEL', 'M', 'FA1') values ('JONES', 'F', 'FA1') ; create table work.fa2 (Name char(6), JCode char(3), Gender char(1)) ; insert into work.fa2 values ('LISTER', 'FA2', 'F') values ('TORRES', 'FA2', 'F') ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> data na1 ; length Name \$ 6 Gender \$ 1 JobCode \$ 3 ; input Name \$ Gender \$ JobCode ; cards ; TORRES M NA1 LANG F NA1 SMITH F NA1 ; data na2; length Name \$ 6 Gender \$ 1 JobCode \$ 3; input Name \$ Gender \$ JobCode; cards; LISTER M NA2 TORRES F NA2 ; </pre>	<pre> proc SQL ; create table work.na1 (Name char(6), Gender char(1), JobCode char(3)) ; insert into work.na1 values ('TORRES', 'M', 'NA1') values ('LANG' , 'F', 'NA1') values ('SMITH' , 'F', 'NA1') ; create table work.na2 (Name char(6), Gender char(1), JobCode char(3)) ; insert into work.na2 values ('LISTER', 'M', 'NA2') values ('TORRES', 'F', 'NA2') ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre> proc sort data=ia.performance ; by Month ; run ; proc sort data=ia.goals ; by Month ; run ; data ia.compare ; merge ia.performance ia.goals ; by Month ; Difference=Sales-Goal ; run ; title 'Sales Performance' ; proc print data=ia.compare ; run ; </pre>	<pre> proc SQL ; create table work.compare as select performance.Month, Sales, Goal, Sales-Goal as Difference from ia.performance full join ia.goals on performance.Month = goals.Month ; title 'Sales Performance' ; select * from work.compare ; quit ; </pre>

SAS Code	SQL Statement(s)
<pre>proc sort data=ia.gersched out=work.gersched ; by EmpID ; run ; data work.nextweek ; merge ia.gercrew work.gersched ; by EmpID ; run ; ** it is assumed that ia.gercrew ; ** is already in EmpID order ;</pre>	<pre>proc SQL ; create table work.nextweek as select gercrew.EmpID , LastName, FlightNum from ia.gercrew full join ia.gersched on gercrew.EmpID = gersched.EmpID ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>proc sort data=ia.gersched out=work.gersched ; by EmpID ; run ; data work.nextweek ; merge ia.gercrew work.gersched(in=InSched) ; by EmpID ; if InSched ; run ;</pre>	<pre>proc SQL ; create table work.nextweek as select gercrew.EmpID , LastName, FlightNum from ia.gercrew, ia.gersched where gercrew.EmpID = gersched.EmpID ; quit ; ** inner joins in SQL automatically ; ** remove non-matches ;</pre>

If the proc means request are using simple statistics (n, min, max, sum, mean std) the SQL select statements can generate similar results. Complex statistics that can be created with the simple statistics (like the range = max - min) can also be generated within an SQL request. Unfortunately, complex statistics are not available within SQL requests. The best example of this is the median. It is easy to request in proc means but not available within proc SQL (or any other ANSI standard version of SQL without vendor specific extensions).

A similar situation exists with proc freq. SQL can easily handle simple frequency counts and cross-tabulations. The more complex a proc freq request becomes the less likely SQL will be able to match it within a simple or single select. This is especially true when other statistics (expected values, chi-square tests, or other measures of association) are requested .

SAS Code	SQL Statement(s)
<pre>title 'Distribution of Job code Values' ; proc freq data=ia.crew ; tables JobCode / nocum ; run ; ** nocum option added to allow for ; ** a closer match with SQL results ;</pre>	<pre>proc SQL ; title 'Distribution of Job code Values' ; select *, (100*Frequency)/sum(Frequency) as Percent from (select JobCode, count(JobCode) as Frequency from ia.crew group by Jobcode) ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>title 'Number of Levels for Job Code' ; proc freq data=ia.crew nlevels ; tables JobCode / noprint ; run ; ** use _all_ keyword on table request ; ** for all levels of all variables ;</pre>	<pre>proc SQL ; title 'Number of Levels for Job Code' ; select 'JobCode' as Variable, count(*) as Levels from (select distinct JobCode from ia.crew) ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>proc format ; value \$codefmt 'FLTAT1'-'FLTAT3'='Flight Attendant' 'PILOT1'-'PILOT3'='Pilot' ; run ;</pre>	
<pre>proc freq data=ia.crew ; tables JobCode / nocum ; format JobCode \$codefmt. ; run ;</pre>	<pre>proc SQL ; select *, (100*Frequency)/sum(Frequency) as Percent from (select put(JobCode,\$codefmt.) as jobcodex , count(*) as Frequency from ia.crew group by 1) ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>proc format ; value \$codefmt 'FLTAT1'-'FLTAT3'='Flight Attendant' 'PILOT1'-'PILOT3'='Pilot' ; value money low-<25000 ='Less than 25,000' 25000-50000='25,000 to 50,000' 50000<-high='More than 50,000' ; run ;</pre>	
<pre>title 'Salary Distribution by Job Codes' ; proc freq data=ia.crew ; tables JobCode*Salary / nocum list ; format JobCode \$codefmt. Salary money. ; run ;</pre>	<pre>proc SQL ; title 'Salary Distribution by Job Codes' ; select *, (100*Frequency)/sum(Frequency) as Percent from (select put(JobCode,\$codefmt.) as JobCode_fmt , put(Salary ,money.) as Salary_fmt , count(*) as Frequency from ia.crew group by 1, 2) ; quit ;</pre>

SAS Code	SQL Statement(s)
<pre>title 'Salary Analysis' ; proc means data=ia.crew ; var Salary ; run ;</pre>	<pre>proc SQL ; title 'Salary Analysis' ; select n(Salary) as n, avg(Salary) as Mean, std(Salary) as StdDev, min(Salary) as Minimum, max(Salary) as Maximum from ia.crew ; quit ; ** requesting the same five ; ** proc means default statistics ;</pre>

SAS Code	SQL Statement(s)
<pre>title 'Salary Analysis by JobCode' ; proc means data=ia.crew ; var Salary ; class JobCode ; run ;</pre>	<pre>proc SQL ; title 'Salary Analysis by JobCode' ; select JobCode, count(*) as NObs, n(Salary) as n, avg(Salary) as Mean, std(Salary) as StdDev, min(Salary) as Minimum, max(Salary) as Maximum from ia.crew group by JobCode ; quit ;</pre>