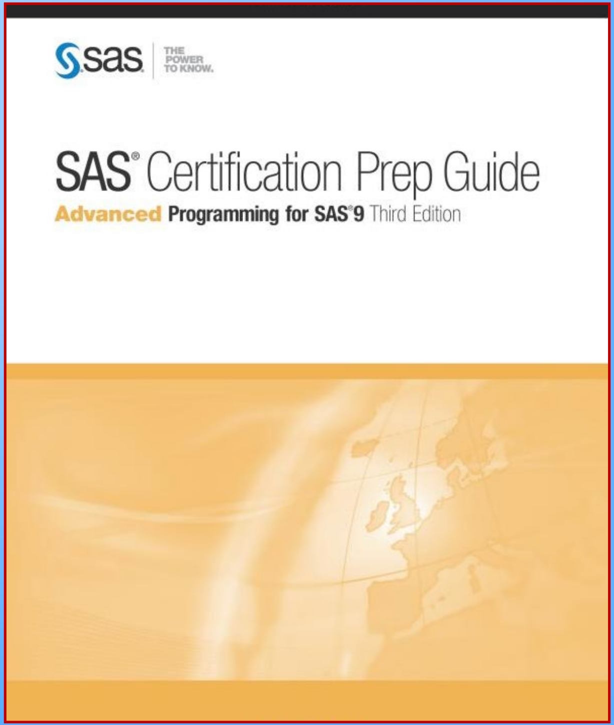
Part 2

SQL Processing with SAS

Reference:

SAS Certification Prep Guide

Advanced Programming for SAS 9 (3rd Edition)

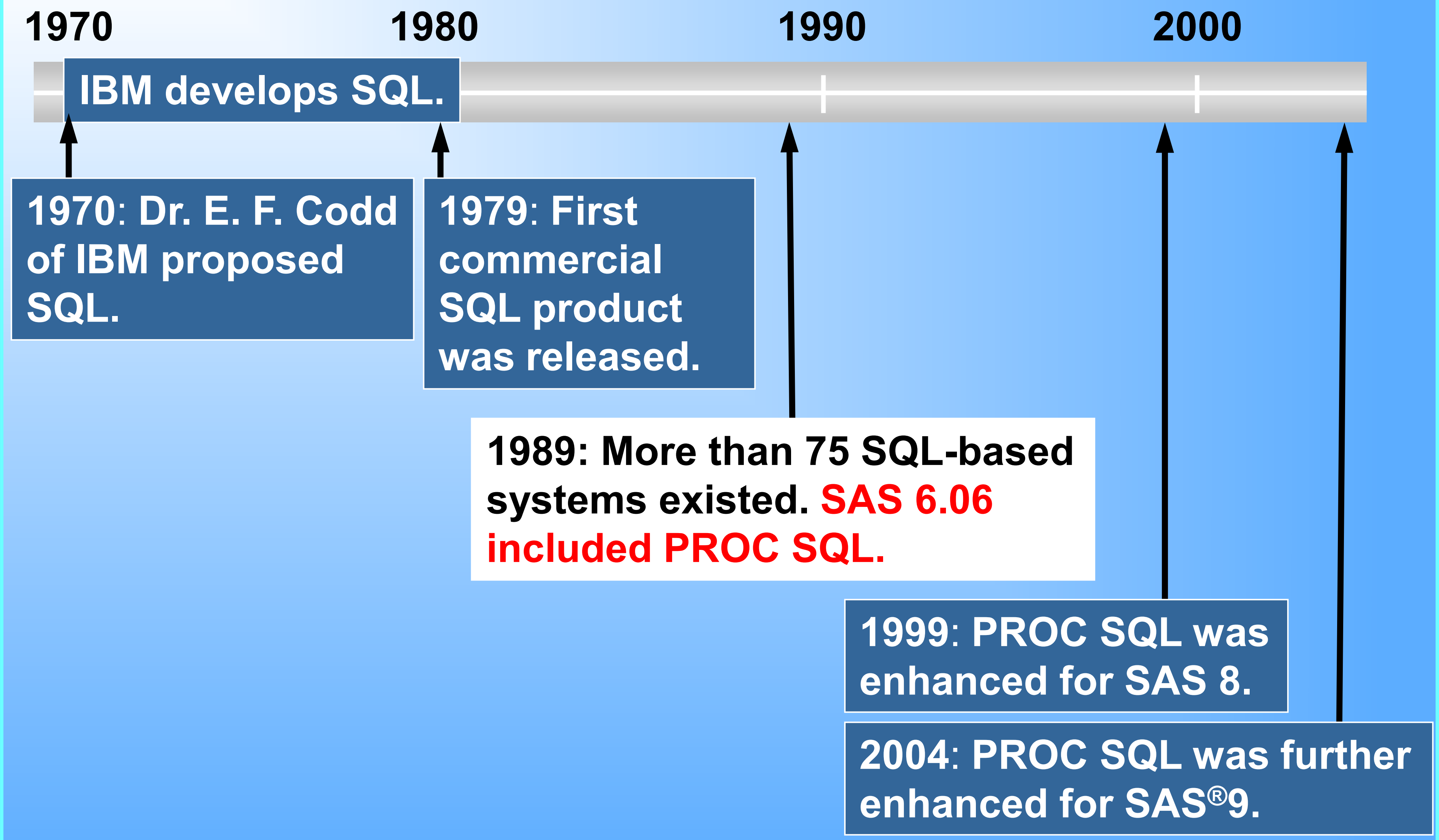

sas THE POWER TO KNOW.
SAS® Certification Prep Guide
Advanced Programming for SAS®9 Third Edition

What will we learn?

- SAS PROC SQL programming.
- Special features of PROC SQL (e.g., the difference between PROC SQL and Oracle SQL*PLUS).
- How can PROC SQL help in advanced SAS programming?
- Do more hands-on practice on SQL.

Note: You must have a SAS license for this Part and have both SAS and Oracle on the same computer for Part 3.

History: SQL and SAS PROC SQL



IBM develops SQL.

1970: Dr. E. F. Codd of IBM proposed SQL.

1979: First commercial SQL product was released.

1989: More than 75 SQL-based systems existed. SAS 6.06 included PROC SQL.

1999: PROC SQL was enhanced for SAS 8.

2004: PROC SQL was further enhanced for SAS®9.

The SQL Procedure in SAS

The PROC SQL procedure has the following characteristics:

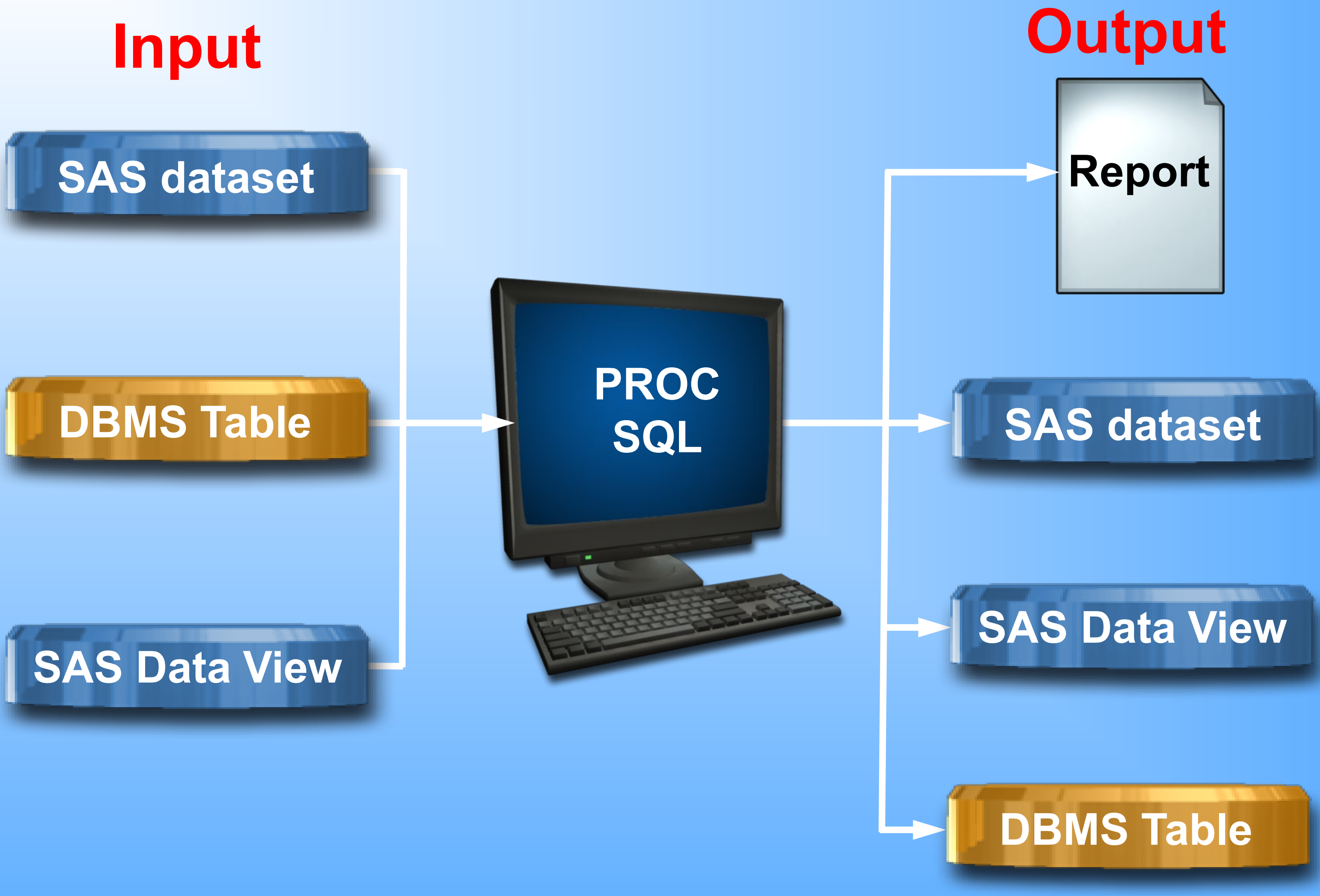
- follows American National Standards Institute (ANSI) standards
- enables the use of SQL in SAS
- a powerful tool for data query, manipulation and management
- includes enhancements for compatibility with the SAS software
- an augmentation to the DATA step, but not a DATA step replacement

The Functionalities of the SQL Procedure

With PROC SQL, you can do the following:

- query SAS datasets
- generate reports from SAS datasets
- combine SAS datasets in many ways
- create and delete SAS datasets, views, and indexes
- update existing SAS datasets
- sometimes reproduce the results of multiple DATA and procedure steps with a single query

What can PROC SQL do?



Some Characteristics of PROC SQL Syntax

- It always starts with the statement: "PROC SQL;"
- It is a SAS step but does not need a RUN statement
- No PROC PRINT step is needed to view query results
- It runs continuously unless you submit another PROC step or a DATA step or you force it to end with the "QUIT" statement

SAS Datasets

A SAS dataset can be any of the following:

- a SAS data file that stores data descriptions and data values together in native SAS format
- a DBMS table accessed via a SAS/ACCESS engine
- a SAS data view, using one of the following technologies:
  - PROC SQL view – a stored SQL query that retrieves data stored in other tables
  - DATA step view – a stored DATA step that retrieves data stored in other files (by adding the VIEW= option in the DATA statement, e.g., DATA mylib.myview / VIEW= mylib.myview;)
  - SAS/ACCESS view – a stored ACCESS descriptor containing information required to retrieve data stored in a DBMS (but it is no longer recommended for accessing relational databases)

Terminology Comparison

Data Processing	SQL	SAS
File	Table	Dataset
Record	Row	Observation
Field	Column	Variable

Study Plan: SAS PROC SQL and Oracle SQL

- PROC SQL possesses the functionalities we have covered in the basic and advanced SQL classes so far.
- The same information will not be repeated but may be compared side-by-side if there is a difference in the syntaxes (SAS PROC SQL *vs.* Oracle SQL*PLUS).
- Unique enhancements of PROC SQL will be discussed in detail.

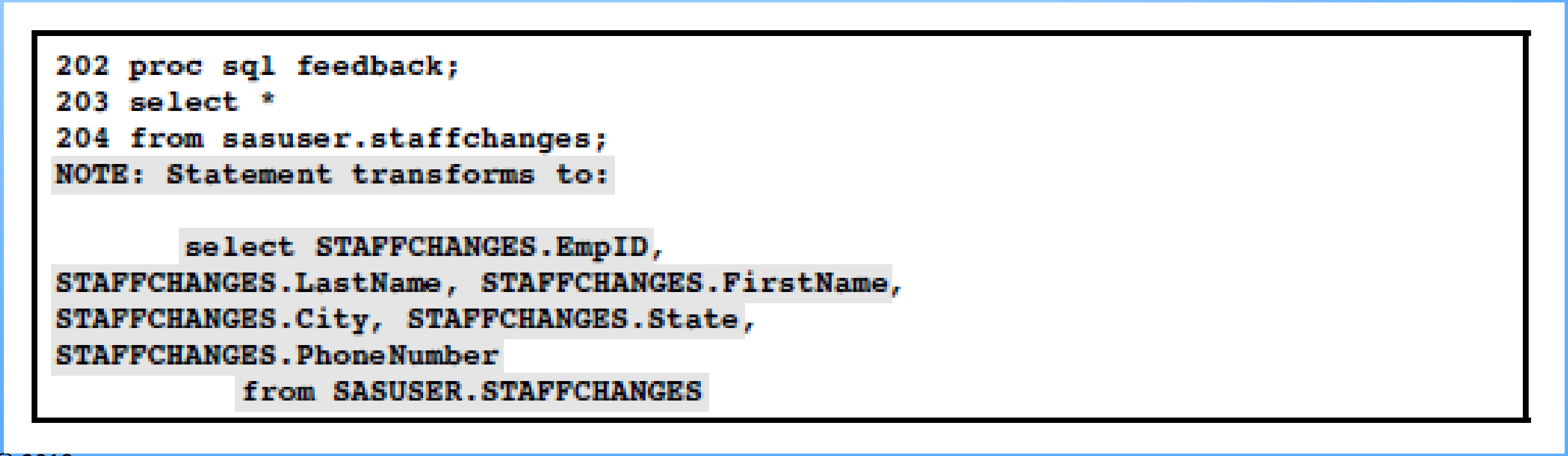
Important Features of the PROC SQL Step

- A PROC SQL statement may contain SAS options.
- Some options immediately follow the PROC SQL keyword: FEEDBACK , OUTOBS=, INOBS=, NOEXE, NUMBER|NONUMBER, STIMER|NOSTIMER.
- Using the VALIDATE keyword.
- Specifying titles and footnotes.
- Using column modifiers: LABEL= and FORMAT=.
- Using the keyword CALCULATED in the WHERE or SELECT clause.
- Adding a character constant to output.

The FEEDBACK Option

When using SELECT *, you can use the *FEEDBACK* option in the PROC SQL statement to write the expanded list of columns to the SAS log; it is a useful debugging tool.

```
proc sql feedback;
select *
from sasuser.staffchanges;
```



```
202 proc sql feedback;
203 select *
204 from sasuser.staffchanges;
NOTE: Statement transforms to:

        select STAFFCHANGES.EmpID,
STAFFCHANGES.LastName, STAFFCHANGES.FirstName,
STAFFCHANGES.City, STAFFCHANGES.State,
STAFFCHANGES.PhoneNumber
            from SASUSER.STAFFCHANGES
```

The OUTOBS= Option

It is for limiting the number of rows that PROC SQL displays in the output. The general form is

PROC SQL OUTOBS= *n*; where *n* specifies the number of rows.

```
proc sql outobs=10;
         select flightnumber, date
                  from sasuser.flightschedule;
```

Note: The OUTOBS= option does not restrict the number of the rows that are read

The INOBS= Option

It is for limiting the number of rows that PROC SQL reads. The general form is **PROC SQL INOBS= *n*;** where *n* specifies the number of rows.

```
proc sql inobs=5;
        select ffid, membertype, name,
        address, city, state, zipcode, pointsused
                from sasuser.frequentflyers
                order by pointsused;
```

Note: The INOBS= option is similar to the SAS system option OBS= and is useful for debugging queries on large tables.

The NOEXEC Option

To verify the syntaxes of all SELECT statements in the PROC SQL step and the existence of the columns and tables that are referenced in the query without executing the query:

```
proc sql noexec;
select empid, jobcode, salary
          from sasuser.payrollmaster
          where jobcode contains 'NA'
          order by salary;
```

If the queries are valid and all the referenced columns and tables exist, the SAS log displays the following message:

NOTE: Statement not executed due to NOEXEC option.

Or, if there are any errors in a query, SAS displays the standard error messages in the log.

The VALIDATE Keyword

It performs a similar task to the NOEXEC option but has a minor difference in syntax and only affects the SELECT statement that immediately follows it:

```
proc sql;
validate
select empid, jobcode, salary
        from sasuser.payrollmaster
        where jobcode contains 'NA'
        order by salary;
```

If the query is valid, the SAS log displays the following message:

NOTE: PROC SQL statement has valid syntax.

If there are errors in the query, SAS displays the standard error messages in the log.

The NUMBER|NONUMBER Option

It specifies whether the output from a query should include a column named ROW, which displays row numbers. NONUMBER is the default.

```
proc sql inobs=10 number;
select flightnumber, destination
  from sasuser.internationalflights;
```

Row	FlightNumber	Destination
1	182	YYZ
2	219	LHR
3	387	CPH
4	622	FRA
5	821	LHR
6	132	YYZ
7	271	CDG
8	182	YYZ
9	219	LHR
10	387	CPH

The STIMER|NOSTIMER Option

It specifies whether PROC SQL writes timing information for each statement to the SAS log, instead of writing a cumulative value for the entire procedure. NOSTIMER is the default:

```
proc sql stimer;
select name, address, city, state, zipcode
   from sasuser.frequentflyers;
select name, address, city, state, zipcode
   from sasuser.frequentflyers
   where pointsearned gt 7000 and
   pointsused lt 3000;
quit;
```

```
205  proc sql stimer;
NOTE: SQL Statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

206  select name, address, city, state, zipcode
207  from sasuser.frequentflyers;
NOTE: SQL Statement used (Total process time):
      real time           0.29 seconds
      cpu time            0.21 seconds

208  select name, address, city, state, zipcode
209  from sasuser.frequentflyers
210  where pointsearned gt 7000 and pointsused lt 3000;
NOTE: SQL Statement used (Total process time):
      real time           0.09 seconds
      cpu time            0.01 seconds

211  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.06 seconds
      cpu time            0.00 seconds
```

Resetting Options

After you specify an option, it remains in effect until you change it, or you re-invoke PROC SQL. You can use the **RESET** statement to add, drop, or change PROC SQL options without re-invoking the SQL procedure.

```
proc sql outobs=5;
select flightnumber, destination
  from sasuser.internationalflights;
reset number;
select flightnumber, destination
  from sasuser.internationalflights
  where boarded gt 200;
reset nonumber;
select flightnumber, destination
  from sasuser.internationalflights
  where boarded gt 200;
quit;
```

FlightNumber	Destination
182	YYZ
219	LHR
387	CPH
622	FRA
821	LHR

Row	FlightNumber	Destination
1	622	FRA
2	821	LHR
3	821	LHR
4	219	LHR
5	219	LHR

FlightNumber	Destination
622	FRA
821	LHR
821	LHR
219	LHR
219	LHR

Extra Attention: SAS PROC SQL *vs.* Oracle SQL*PLUS

- Quotation marks for character contents
  - PROC SQL normally allows to use single or double quotation marks.
  - SQL*PLUS only allows single quotation marks.
- Column labels and formats
  - PROC SQL allows to use labels for better column head displaying and to format your column output.
  - SQL*PLUS does not have these features.
- Column alias
  - PROC SQL displays a column alias <u>exactly as you typed in</u>.
  - SQL*PLUS always displays in upper case.
- GROUP BY clause
  - In PROC SQL, a GROUP BY clause is changed to an ORDER BY clause if there is no summary/aggregate function in the query.
  - In SQL*PLU, a GROUP BY clause must pair with a summary/aggregate function.

Column Modifiers: LABEL= and FORMAT=

You can control the formatting of columns in output by using LABEL= and FORMAT=, after any column name in the SELECT clause. When you define a new column in the SELECT clause, you can also assign a label rather than an alias

```
proc sql outobs=15;
select empid label='Employee ID',
   jobcode label="Job Code", salary,
   salary * .10 label='Bonus'
   format=dollar12.2
   from sasuser.payrollmaster
   where salary>75000
   order by salary desc;
```

Employee ID	Job Code	Salary	**Bonus**
1118	PT3	$155,931	$15,593.10
1777	PT3	$153,482	$15,348.20
1404	PT2	$127,926	$12,792.60
1107	PT2	$125,968	$12,596.80
1928	PT2	$125,801	$12,580.10
1106	PT2	$125,485	$12,548.50
1333	PT2	$124,048	$12,404.80
1890	PT2	$120,254	$12,025.40
1410	PT2	$118,559	$11,855.90
1442	PT2	$118,350	$11,835.00
1830	PT2	$118,259	$11,825.90
1478	PT2	$117,884	$11,788.40
1556	PT1	$99,889	$9,988.90
1439	PT1	$99,030	$9,903.00
1428	PT1	$96,274	$9,627.40

Add Titles and Footnotes

You can add titles and footnotes to your PROC SQL step, but you can place the TITLE and FOOTNOTE statements in either of the following locations: 1) before the PROC SQL statement and, 2) between the PROC SQL statement and the SELECT statement (or right before the SELECT statement of interest if there is more than one SELECT statement.

```
proc sql outobs=15;
title 'Current Bonus Information';
title2 'Employees with Salaries > $75,000';
footnote1 "Created &systime &sysday, &sysdate9";
footnote2 "on the &sysscp system using Release &sysver";
select empid label='Employee ID', jobcode,
salary, salary * .10 as Bonus
        from sasuser.payrollmaster
        where salary>75000
        order by salary desc;
```

The result

Current Bonus Information
Employees with Salaries > $75,000

Employee ID	JobCode	Salary	Bonus
1118	PT3	$155,931	15593.1
1777	PT3	$153,482	15348.2
1404	PT2	$127,926	12792.6
1107	PT2	$125,968	12596.8
1928	PT2	$125,801	12580.1
1106	PT2	$125,485	12548.5
1333	PT2	$124,048	12404.8
1890	PT2	$120,254	12025.4
1410	PT2	$118,559	11855.9
1442	PT2	$118,350	11835
1830	PT2	$118,259	11825.9
1478	PT2	$117,884	11788.4
1556	PT1	$99,889	9988.9
1439	PT1	$99,030	9903
1428	PT1	$96,274	9627.4

Created 23:54 Wednesday, 01NOV2017
on the WIN system using Release 9.4

Example: In PROC SQL a GROUP BY Clause Is Changed into an ORDER BY Clause

```
proc sql;
select membertype,
  milestraveled label='Miles Traveled'
     from sasuser.frequentflyers
     group by membertype;
 quit;
```

```
WARNING: A GROUP BY clause has been transformed into
         an ORDER BY clause because neither the
         SELECT clause nor the optional HAVING clause
         of the associated table-expression referenced a
         summary function.
```

The CALCULATED Keyword in PROC SQL

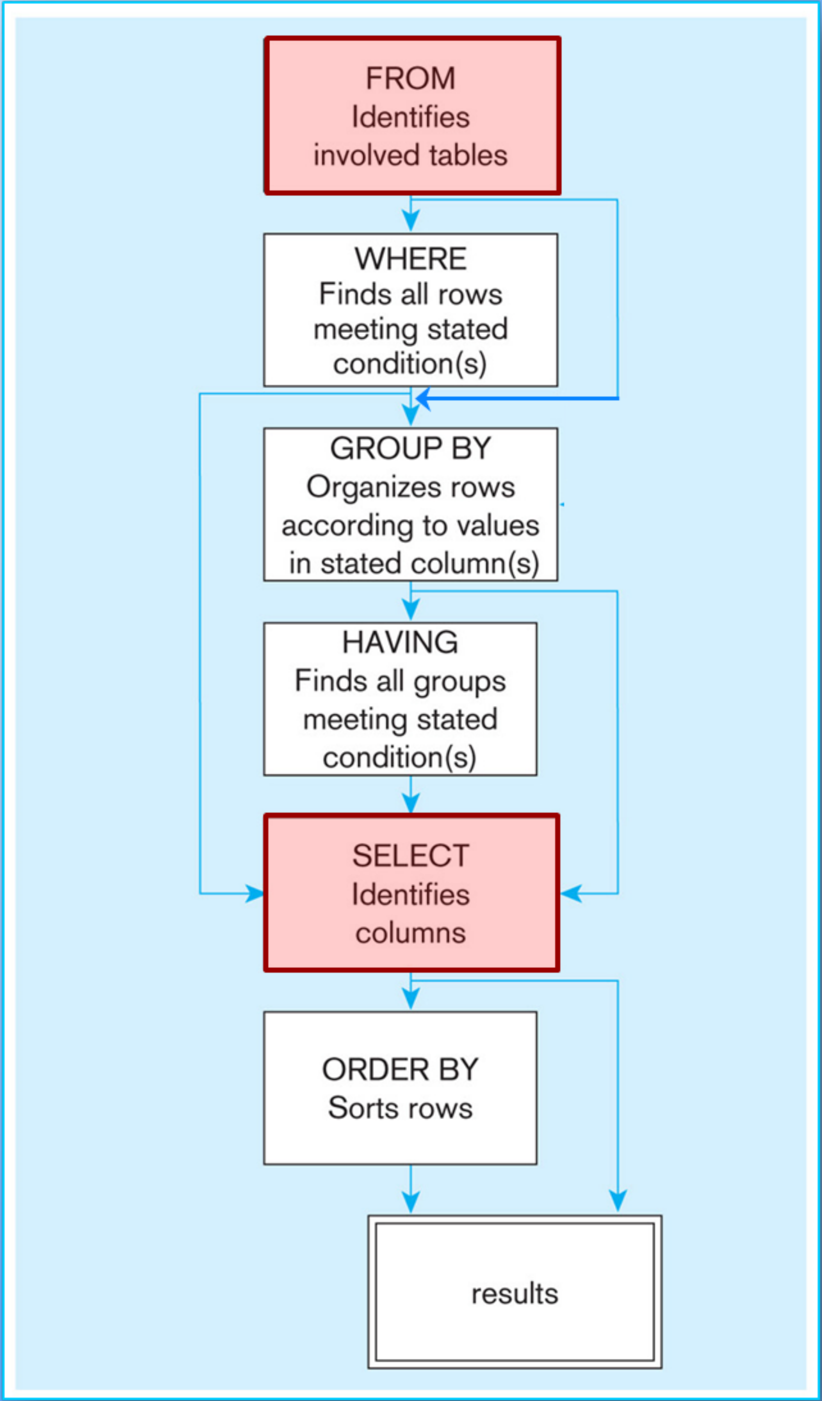
- Because of the order in which SQL queries are processed, you cannot just specify a column alias in the WHERE clause. You must use the keyword *CALCULATED* along with the alias.

```
proc sql outobs=10;
select flightnumber, date, destination,
boarded + transferred + nonrevenue as Total
   from sasuser.marchflights
   where calculated total < 100;
```

- The CALCULATED keyword can also be used in the SELECT clause

```
proc sql outobs=10;
select flightnumber, date, destination,
   boarded + transferred + nonrevenue as Total,
   calculated total/2 as Half
   from sasuser.marchflights;
```

SELECT statement processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results

Adding a Character Constant to Output

A text string in quotation marks can be included in the SELECT clause; this will define a column that contains a character constant.

```
proc sql outobs=15;
title 'Current Bonus Information';
title2 'Employees with Salaries > $75,000';
select empid label='Employee ID',
   jobcode label='Job Code', salary, "bonus is:",
   salary * .10 format=dollar12.2
   from sasuser.payrollmaster
   where salary>75000
   order by salary desc;
```

Current Bonus Information
Employees with Salaries > $75,000

Employee ID	Job Code	Salary		
1118	PT3	$155,931	bonus is:	$15,593.10
1777	PT3	$153,482	bonus is:	$15,348.20
1404	PT2	$127,926	bonus is:	$12,792.60
1107	PT2	$125,968	bonus is:	$12,596.80
1928	PT2	$125,801	bonus is:	$12,580.10
1106	PT2	$125,485	bonus is:	$12,548.50
1333	PT2	$124,048	bonus is:	$12,404.80
1890	PT2	$120,254	bonus is:	$12,025.40
1410	PT2	$118,559	bonus is:	$11,855.90
1442	PT2	$118,350	bonus is:	$11,835.00
1830	PT2	$118,259	bonus is:	$11,825.90
1478	PT2	$117,884	bonus is:	$11,788.40
1556	PT1	$99,889	bonus is:	$9,988.90
1439	PT1	$99,030	bonus is:	$9,903.00
1428	PT1	$96,274	bonus is:	$9,627.40

Note: Oracle also supports this functionality but you have to use single quotation marks. Also, there is a difference in column name display.