SAS® SQL 101
Pharmasug 2023

Course Notes

SAS® SQL 101 for Pharmasug Course Notes was developed by Charu Shankar.

SAS® SQL Masterclass for University of Waterloo Course Notes

Course Description

This course teaches you how to process SAS data using Structured Query Language (SQL).

To learn more…

For information about other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the web at http://support.sas.com/training/ as well as in the Training Course Catalog.

For a list of other SAS books that relate to the topics covered in this course notes, USA customers can contact the SAS Publishing Department at 1-800-727-3228 or send e-mail to sasbook@sas.com. Customers outside the USA, please contact your local SAS office.

Also, see the SAS Bookstore on the web at http://support.sas.com/publishing/ for a complete list of books and a convenient order form.

Prerequisites

Before attending this class, you should be able to

- submit SAS programs on your operating system
- create and access SAS data sets
- use arithmetic, comparison, and logical operators
- invoke SAS procedures.

You can gain this experience from the SAS® Programming 1: Essentials course. No knowledge of SQL is necessary.

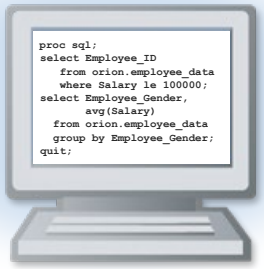
SQL 101

1.1 Introducing PROC SQL

Structured Query Language

Structured Query Language (SQL) is a standardized language originally designed as a relational database query tool.

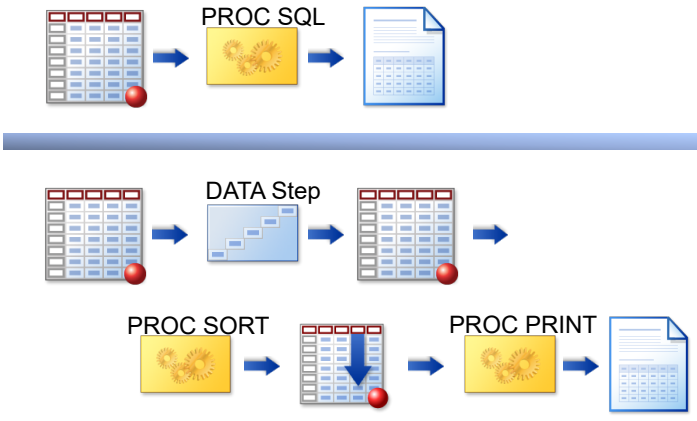
SQL is currently used in many software products to retrieve and update data.



3

SQL Procedure versus Traditional SAS

The SQL procedure can sometimes reproduce the results of multiple DATA and procedure steps with a single query.



4

Objectives

- Describe the data that is used in the course.
- Execute a SAS program to create course data files.
- Execute a SAS program to define the data location.

5

1.2 PROC SQL Overview

Objectives

- Identify key syntax of the SQL procedure.
- List key features of the SQL procedure.
- List key features of the SELECT statement.
- List SQL procedure statements.

8

SELECT Statement

A SELECT statement contains smaller building blocks called *clauses*.

```
proc sql;
select Employee_ID, Employee_Gender, Salary
    from orion.employee_information
    where Employee_Gender='F'
    order by Salary desc;
quit;
```

clauses

- Although it can contain multiple clauses, each SELECT statement begins with the SELECT keyword and ends with a semicolon.

9

s102d01

PROC SQL does not require a RUN statement. It uses the QUIT statement to explicitly terminate SQL processing. The SQL procedure, like other SAS procedures, is terminated if SAS encounters a DATA step or a PROC step.

Viewing the Output

Partial PROC SQL Output

The SAS System

Employee ID	Employee Gender	Employee Annual Salary
120260	F	$207,885
120719	F	$87,420
120661	F	$85,495
121144	F	$83,505
120798	F	$80,755

10

SELECT Statement: Required Clauses

```
SELECT object-item <, ...object-item>
    FROM from-list;
```

- The SELECT clause specifies the columns and column order.
- The FROM clause specifies the data sources.
- You can query from 1 to 256 tables.

11

SELECT Statement Syntax

```
PROC SQL;
SELECT object-item <, ...object-item>
    FROM from-list
    <WHERE sql-expression>
    <GROUP BY object-item <, … object-item >>
    <HAVING sql-expression>
    <ORDER BY order-by-item <DESC>
                <, …order-by-item>>;
QUIT;
```

- The specified order of the above clauses within the SELECT statement is required.

12

1.3 Specifying Columns

Objectives

- Explore unfamiliar data.
- Display columns directly from a table.
- Display columns calculated from other columns in a query.

15

Querying All Columns in a Table

To print all of a table's columns in the order in which they were stored, specify an asterisk in a SELECT clause.

```
proc sql;
select *
   from orion.employee_information;
quit;
```

Partial PROC SQL Output

The SAS System

Employee ID	Start Date	End Date	Department	Employee Annual Salary	Employee Gender	Employee Birth Date	Employee Hire Date	Employee Termination Date	Manager for Employee
120101	01JUL2007	31DEC9999	Sales Management Director	$163,040	M	18AUG1980	01JUL2007	.	120261

s102d04

16

Business Scenario

Produce a report that contains selected information for all Orion Star employees.

orion.employee_information

PROC SQL

Employee_ID	Employee_ Gender	Salary
120101	M	163040
120102	M	108255
120103	M	87975
120104	F	46230
120105	F	27110

17

Querying Specific Columns in a Table

List the columns that you want and the order to display them in the SELECT clause.

```
proc sql;
select Employee_ID, Employee_Gender,
       Salary
   from orion.employee_information;
quit;
```

18 s102d06

Remember to use commas to separate items in a list, such as a list of column names in the SELECT, GROUP BY, or ORDER BY clauses.

Viewing the Output

Partial PROC SQL Output

The SAS System

Employee ID	Employee Gender	Employee Annual Salary
120101	M	$163,040
120102	M	$108,255
120103	M	$87,975
120104	F	$46,230
120105	F	$27,110
120106	M	$26,960
120107	F	$30,475

19

Business Scenario

Modify the previous report by creating a new column, **Bonus**, which contains an amount equal to 10% of the employee's salary.

orion.employee_information

PROC SQL

Employee_ID	Salary	Bonus
120101	163040	16304
120102	108255	10825.5
120103	87975	8797.5
120104	46230	4623
120105	27110	2711

20

Calculated Columns

Name the new column using the AS keyword.

```
proc sql;
select Employee_ID, Salary,
       Salary*.10 as Bonus
   from orion.employee_information;
quit;
```

Partial PROC SQL Output

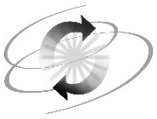
```
                The SAS System

                     Employee
                       Annual
 Employee ID           Salary        Bonus
 ------------------------------------------
      120101          $163,040      16304
      120102          $108,255      10825.5
      120103           $87,975       8797.5
         ...
```

21

s102d07

The new column name, **Bonus**, is called an *alias*. Assigning an alias to a calculated column is optional, but if an alias ***is*** assigned, the AS keyword is required. Omission of the alias causes the column heading in the report to be blank.



Exercises

If you restarted your SAS session since the last exercise, open and submit the **libname.sas** program found in the data folder.

1. Calculating a Column – Solution Program is S102s02.sas

Write a query that generates the report below. The report should do the following:

- display **Employee_ID**, **Employee_Gender**, **Marital_Status**, **Salary**, and a new column (**Tax**) that is one-third of the employee's salary
- use the **orion.employee_payroll** table

Partial PROC SQL Output

Employee_ID	Employee_ Gender	Marital_ Status	Salary	Tax
120101	M	S	163040	54346.67
120102	M	O	108255	36085
120103	M	M	87975	29325
120104	F	M	46230	15410
120105	F	S	27110	9036.667
120106	M	M	26960	8986.667

End of Exercises

1.4 Specifying Rows

Objectives

- Select a subset of rows in a query.

25

Business Scenario

Management requested a list of employees whose salaries exceed $112,000.

orion.employee_information

Employee ID	Employee Job Title	Employee Annual Salary
120101	Director	$163,040
120259	Chief Executive Officer	$433,800
120260	Chief Marketing Officer	$207,885
120261	Chief Sales Officer	$243,190
120262	Chief Financial Officer	$268,455

26

Subsetting with the WHERE Clause

Use a WHERE clause to specify a condition that the data must satisfy before being selected.

```
proc sql;
select Employee_ID, Job_Title, Salary
   from orion.employee_information
   where Salary > 112000;
quit;
```

WHERE *sql-expression*

27

s102d14

Viewing the Output

PROC SQL Output

```
                    The SAS System
                                                    Employee
                                                      Annual
Employee ID  Employee Job Title                       Salary

-------------------------------------------------------------

     120101  Director                                $163,040
     120259  Chief Executive Officer                 $433,800
     120260  Chief Marketing Officer                 $207,885
     120261  Chief Sales Officer                     $243,190
     120262  Chief Financial Officer                 $268,455
     120659  Director                                $161,290
     121141  Vice President                          $194,885
     121142  Director                                $156,065
```

28

Business Scenario

Management requested a report that includes only those employees who receive bonuses less than $3000.

orion.employee_information

PROC SQL

Employee_ID	Employee_ Gender	Salary	Bonus
120105	F	27110	2711
120106	M	26960	2696
120108	F	27660	2766
120109	F	26495	2649.5
120110	M	28615	2861.5

29

Subsetting with Calculated Values

First attempt:

```
proc sql;
select Employee_ID, Employee_Gender,
       Salary, Salary*.10 as Bonus
   from orion.employee_information
   where Bonus<3000;
quit;
```

A ***WHERE*** clause is evaluated before the ***SELECT*** clause. Therefore, columns used in the WHERE clause must exist in the table.

Partial SAS Log

```
ERROR: The following columns were not found in the contributing
tables: Bonus.
```

s102d15

30

Subsetting with Calculated Values

An alternate method is to use the CALCULATED keyword in the WHERE clause.

```
proc sql;
select Employee_ID, Employee_Gender,
       Salary, Salary*.10 as Bonus
   from orion.employee_information
   where calculated Bonus<3000;
quit;
```

SAS enhancement

31 s102d15

Viewing the Output

Partial PROC SQL Output

The SAS System

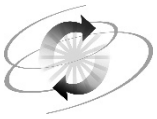
Employee ID	Employee Gender	Employee Annual Salary	Bonus
120105	F	$27,110	2711
120106	M	$26,960	2696
120108	F	$27,660	2766
120109	F	$26,495	2649.5
120110	M	$28,615	2861.5
120111	M	$26,895	2689.5
120112	F	$26,550	2655

32

The CALCULATED keyword or repeated calculated column expression is required when referring to any calculated column, character or numeric, in the SELECT or WHERE clause, but it is not necessary with the ORDER BY or HAVING clause.

Example:

```
select Employee_ID, Salary,
       (scan(Job_Title,-1,' ')) as Job_Level
    from orion.Staff
    where calculated Job_Level='IV';
```



Exercises

If you restarted your SAS session since the last exercise, open and submit the **libname.sas** program found in the data folder.

2. Subsetting Data – Solution program is s102s06.sas

Write a query that generates a report that displays Orion Star employees whose charitable contributions exceed $90.00. The report should have the following characteristics:

- display **Employee_ID**, **Recipients**, and the new column **Total** that represents the total charitable contribution for each employee over the four quarters
- use the **orion.employee_donations** table
- include only employees whose charitable contribution **Total** for all four quarters exceeds $90.00

Hint: The total charitable contribution is calculated by adding the amount of **Qtr1**, **Qtr2**, **Qtr3**, and **Qtr4**. Use the SUM function to ensure that missing values are ignored.

Partial PROC SQL Output

Employee ID	Recipients	Total
120660	Disaster Assist, Inc.	100
120677	EarthSalvors 60%, Vox Victimas 40%	100
120753	Conserve Nature, Inc. 50%, AquaMissions International 50%	100
120766	Mitleid International 80%, Save the Baby Animals 20%	100
120791	Child Survivors	120

End of Exercises

1.5 Joining Tables

Objectives

- Identify different ways to combine data horizontally from multiple tables.
- Distinguish between inner and outer SQL joins.
- Understand the Cartesian product.

46

Exploring the Data

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

The **customers** table is representative of a customer dimension table. There would be additional columns with data about our customers including address, age, and so on.

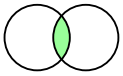
The **transactions** table is representative of a fact table. There would be columns holding all the key column data, **Product_ID**, **Employee_ID**, and so on.

47

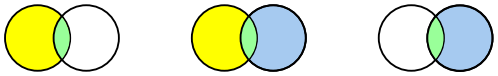
Types of Joins

PROC SQL supports two types of joins:

Inner joins return only matching rows.



Outer joins return all matching rows, plus nonmatching rows from one or both tables.



Left Full Right

48

Cartesian Product

A query that lists multiple tables in the FROM clause without a WHERE clause produces all possible combinations of rows from all tables. This result is called a *Cartesian product*.

```
proc sql;
select *
   from customers, transactions;
quit;
```

SELECT ...
FROM *table-name*, *table-name*
<, …,*table-name* >;

To understand how SQL processes a join, it is helpful to understand the concept of the Cartesian product.

s104d01

49

Building the Cartesian Product

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

Result Set

ID	Name	ID	Action	Amount
101	Smith	102	Purchase	$100
101	Smith	103	Return	$52
101				$212
104				$100
104				$52
104				$212
102				$100
102	Blank	103	Return	$52
102	Blank	105	Return	$212

The Cartesian product is rarely the desired result of a query.

50

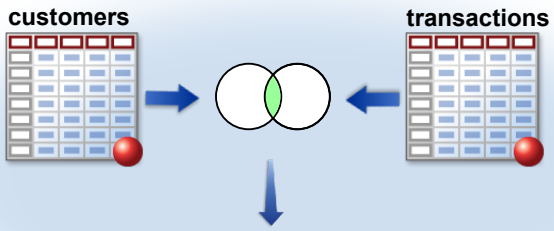
Objectives

- Join two or more tables on matching columns.
- Qualify column names to identify specific columns.
- Use a table alias to simplify the SQL code.

52

Report 1: Inner Join

Management has requested a report showing all valid order information.



ID	Name	Action	Amount
102	Blank	Purchase	$100

53

Inner Join

Specify the matching criteria in the WHERE clause.

```
proc sql;
select *
   from customers, transactions
   where customers.ID=
         transactions.ID;
quit;
```

SELECT *object-item<, …object-item>*
FROM *table-name, … table-name*
WHERE *join condition*
*<**AND** sql-expression>*
<other clauses>;

PROC SQL Output

ID	Name	ID	Action	Amount
102	Blank	102	Purchase	$100

54

s104d02

Completed Code for Report 1

To display the ID column only once in the results, qualify the ID column in the SELECT clause.

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

```
select customers.ID, Name, Action, Amount
   from customers, transactions
   where customers.ID=transactions.ID;
```

ID	Name	Action	Amount
102	Blank	Purchase	$100

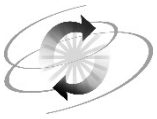
s104d03

55

Compare SQL Join and DATA Step Merge

Key Points	SQL Join	DATA Step Merge
Explicit sorting of data before join/merge	Not required	Required
Same-name columns in join/merge expressions	Not required	Required
Equality in join or merge expressions	Not required	Required

57



Exercises

If you restarted your SAS session since the last exercise, open and submit the **libname.sas** program found in the data folder.

3. Creating a Summary Report from Two Tables – solution program is s104s02.sas

The head of the Sales Department wants to know how many of each product was sold since the beginning of 2007. The report should include the product ID, the product name, and the total sold for that product and ordered to match the output shown below.

The data that you need can be found in the listed columns of the following tables:

- **ORION.product_dim** contains
  - **Product_ID**
  - **Product_Name**.
- **ORION.order_fact** contains
  - **Product_ID**
  - **Quantity**.

Partial PROC SQL Output

Total Quantities Sold by Product ID and Name

Product ID	Product Name	Total Sold
240800200035	Shine Black PRO	6
240700100001	Armour L	5
220101400088	Casual Genuine Polo-Shirt	5
240100100737	Wyoming Men's T-Shirt with V-Neck	5
240200100057	Carolina II	4
210200600067	Children's Knit Sweater	4
240700400017	Fga Home Shorts	4
240700200018	Helmet L	4
240200100118	Hi-fly Intrepid Stand 8 Black	4
220100300019	Instyle Pullover Mid w/Small Zipper	4
230100500092	Mayday Sports Pullover	4
210200900033	Osprey France Nylon Shorts	4
240200100173	Proplay Executive Bi-Metal Graphite	4

End of Exercises

1.6 Solutions

Solutions to Exercises

1. **Calculating a Column**

```
*** s102s02 ***;
proc sql;
select Employee_ID, Employee_Gender, Marital_Status,
       Salary, Salary/3 as Tax
   from orion.employee_payroll;
quit;
```

2. **Subsetting Data**

```
*** s102s06 ***;
proc sql;
select Employee_ID, Recipients,
       sum(Qtr1,Qtr2,Qtr3,Qtr4) as Total
   from orion.employee_donations
   where calculated Total>90;
quit;
```

3. **Creating a Summary Report from Two Tables**

```
*** s104s02 ***;
proc sql;
title 'Total Quantities Sold by Product ID and Name'; select
   p.Product_ID,
Product_Name, sum(Quantity) 'Total Sold'
from orion.product_dim as p, orion.order_fact as o
where p.Product_ID = o.Product_ID and Order_Date>='01Jan2007'd group
   by p.Product_ID, Product_Name
order by 3 desc, Product_Name; quit;
title;
```

End of Solutions