SQL 101

RTSUG

20 February 2024

Charu Shankar

SAS Education

SQL 101

Charu Shankar, SAS® Institute

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking since 2007.

Charu curates and delivers unique content on SAS, SQL, Viya, etc. to support users in the adoption of SAS software.

When not coding, Charu teaches yoga and loves to explore Canadian trails with her husky Miko.

Agenda

Nuts & Bolts - PROC SQL Overview

Specifying Columns

Specifying Rows

Summarizing Data

Joining Tables

Handy Links

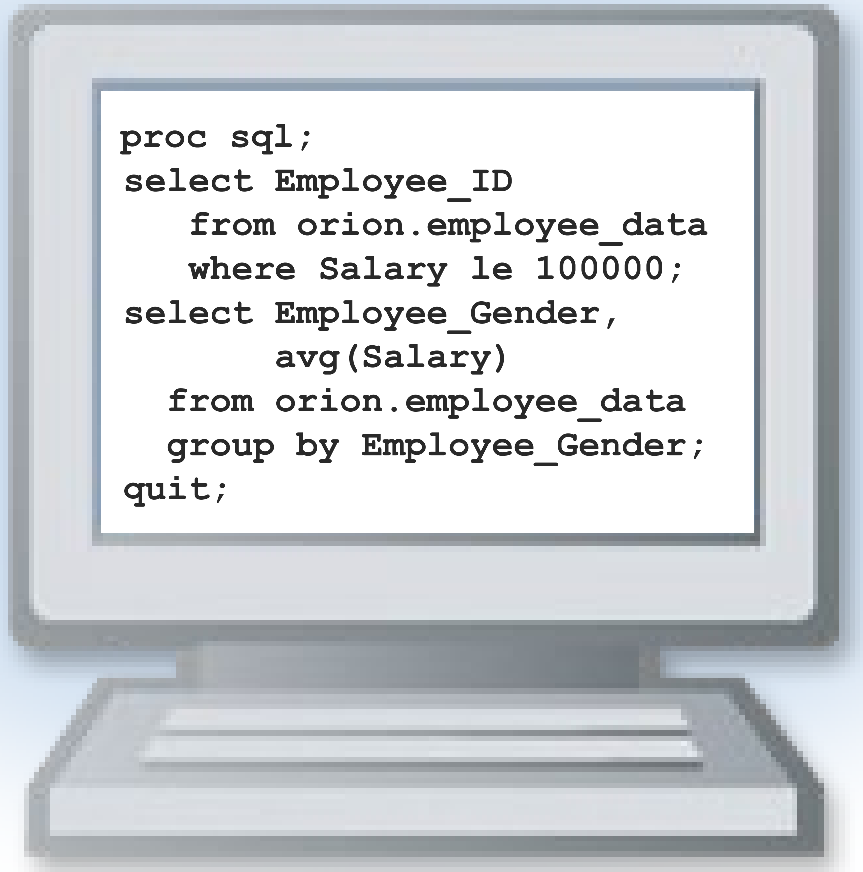
1 Introducing PROC SQL

Nuts & Bolts - PROC SQL Overview

Structured Query Language

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

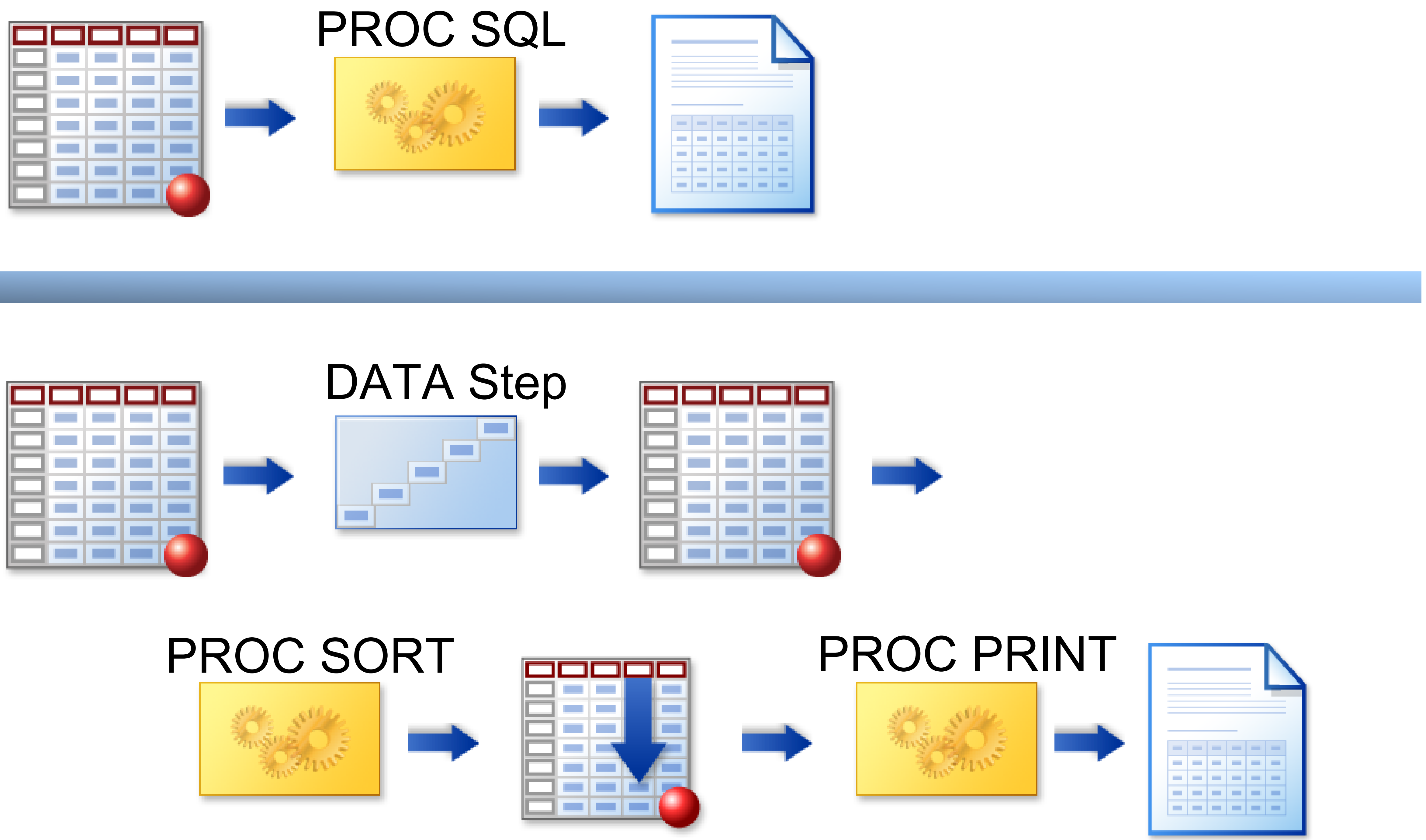
SQL is currently used in many software products to retrieve and update data.



```
proc sql;
select Employee_ID
   from orion.employee_data
   where Salary le 100000;
select Employee_Gender,
       avg(Salary)
  from orion.employee_data
  group by Employee_Gender;
quit;
```

SQL Procedure versus Traditional SAS

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



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.

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;
```

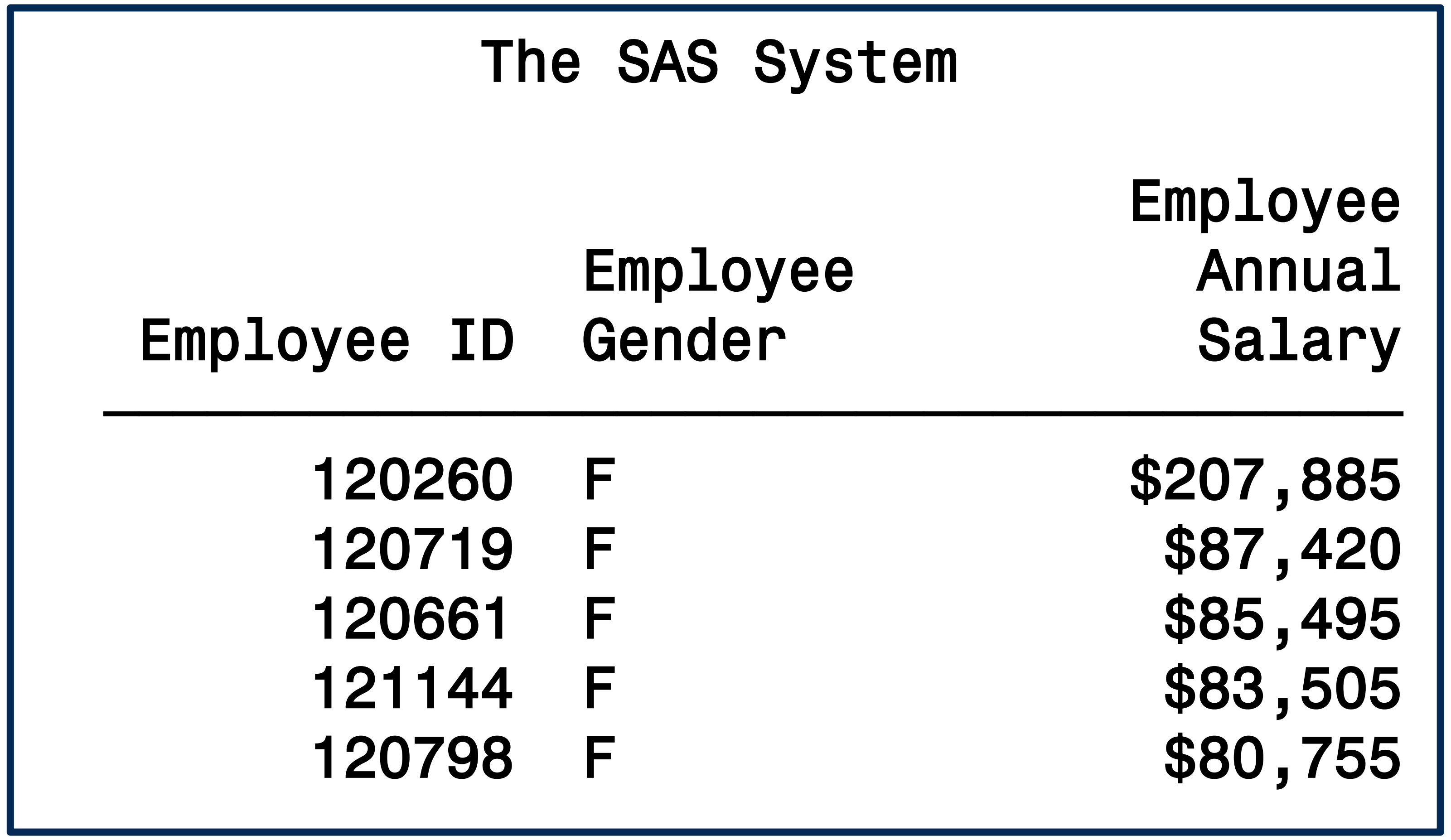
clauses

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

s102d01

Viewing the Output

Partial PROC SQL Output



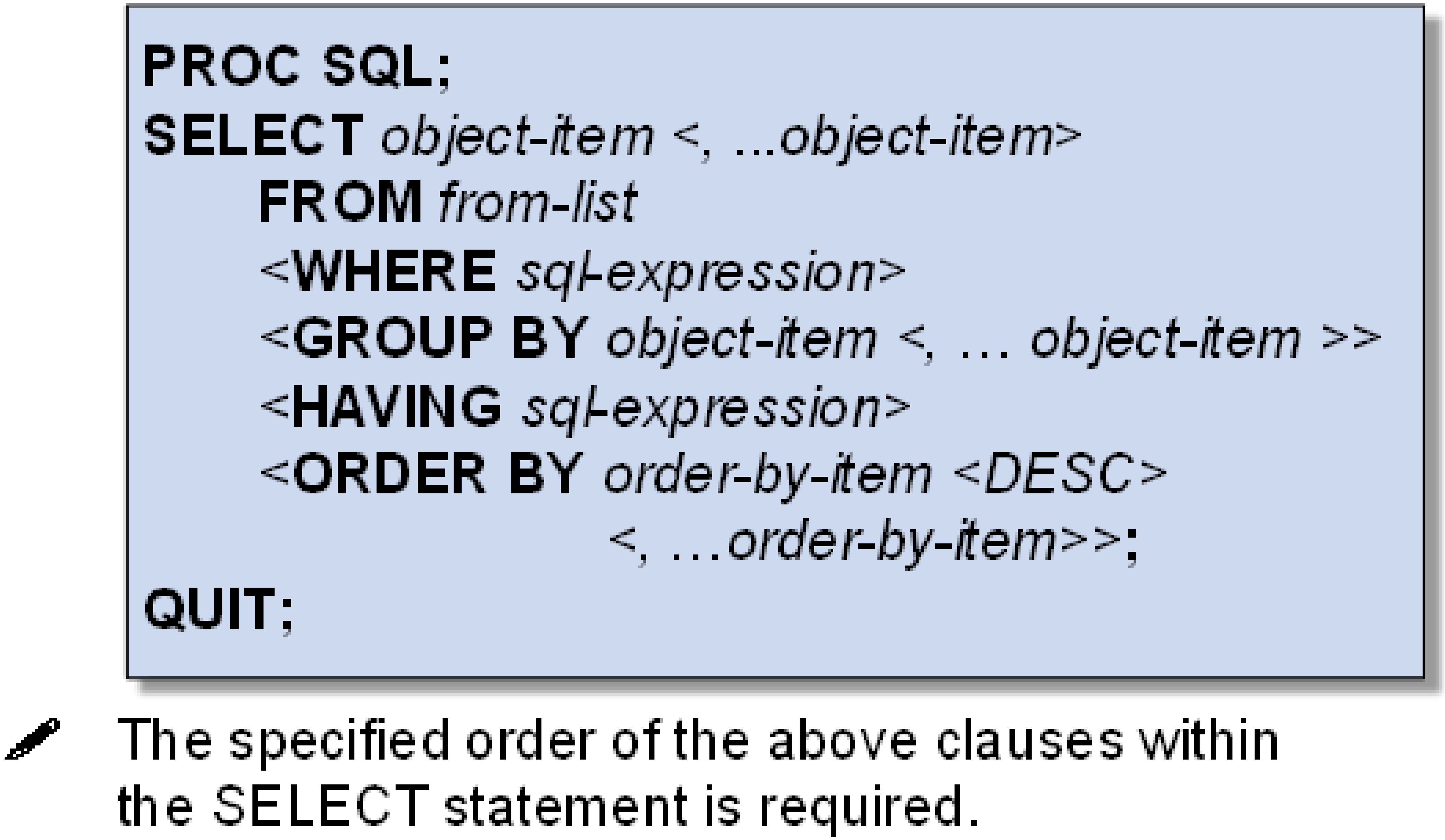
```
                The SAS System

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

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.

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.

Questions??

Specifying Columns

Objectives

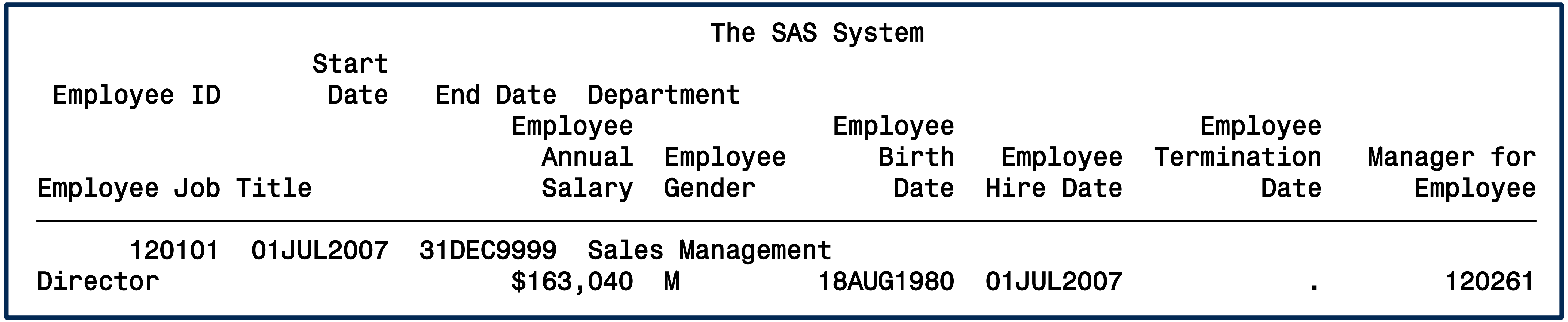
- Explore unfamiliar data.
- Display columns directly from a table.
- Display columns calculated from other columns in a query.

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

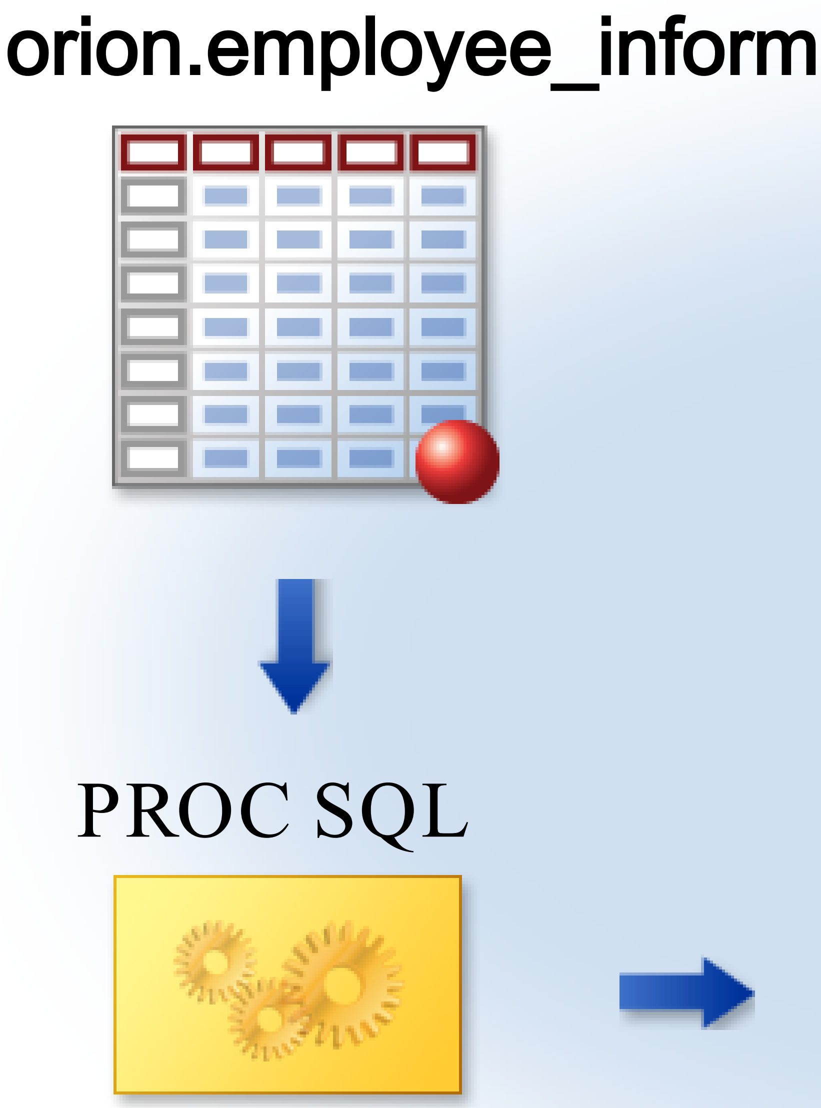


```
                                            The SAS System
                  Start
 Employee ID       Date   End Date  Department
                               Employee             Employee                Employee
                                 Annual  Employee      Birth   Employee  Termination   Manager for
Employee Job Title               Salary  Gender         Date  Hire Date         Date      Employee
--------------------------------------------------------------------------------------------------

      120101  01JUL2007  31DEC9999  Sales Management
Director                       $163,040  M         18AUG1980  01JUL2007            .        120261
```

s102d04

Business Scenario

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



orion.employee_information

PROC SQL

```
                 Employee_
Employee_ID     Gender         Salary
_____________________________________

     120101     M              163040
     120102     M              108255
     120103     M               87975
     120104     F               46230
     120105     F               27110
```

Querying Specific Columns in a Table

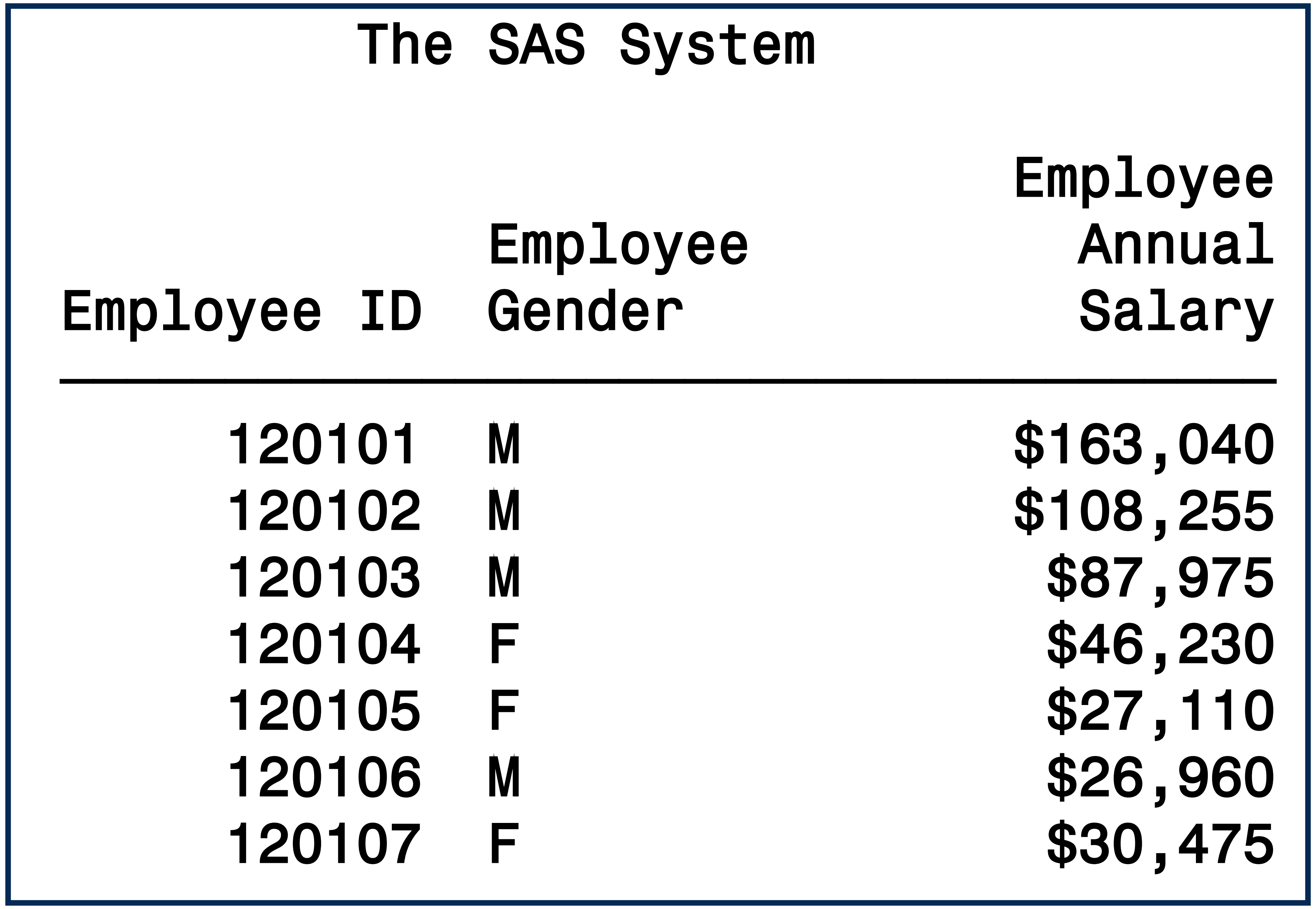
List the columns that you want and the order to display them in the SELECT clause.

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

s102d06

Viewing the Output

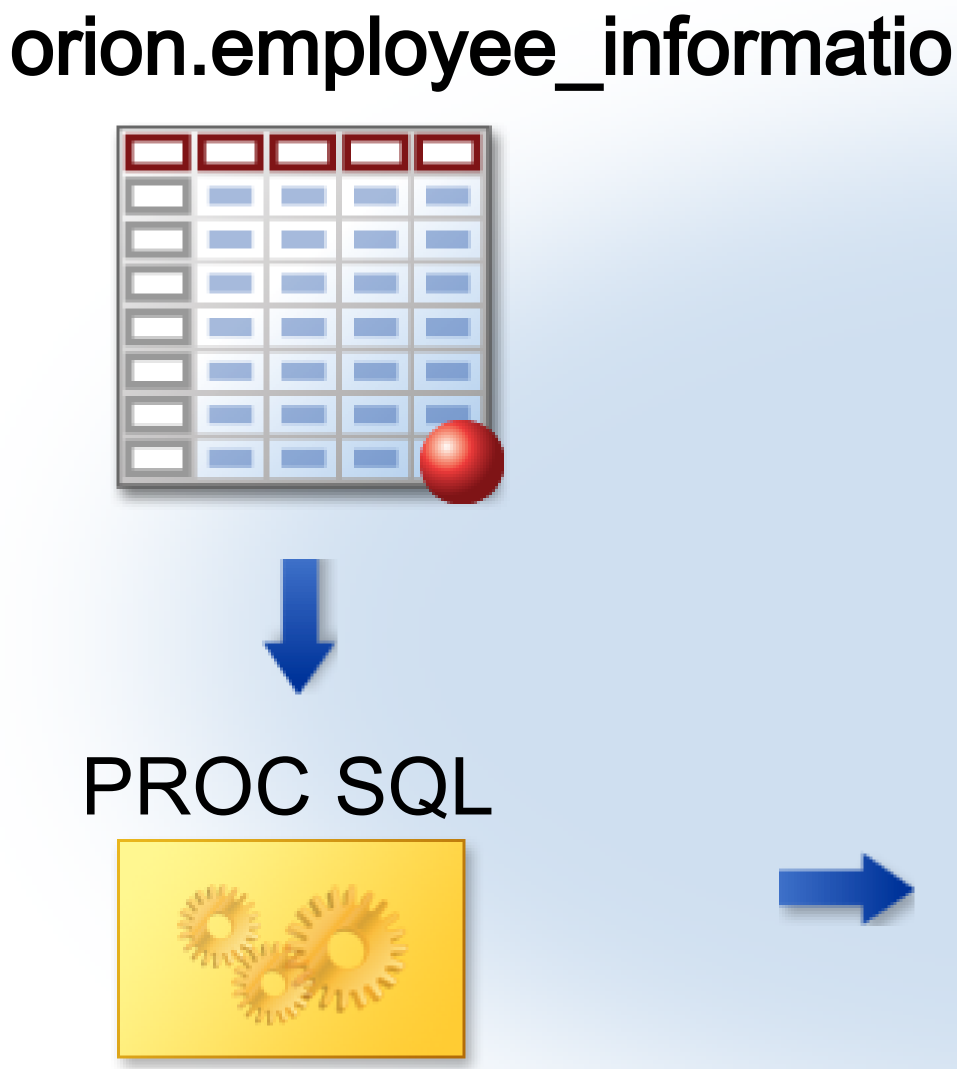
Partial PROC SQL Output

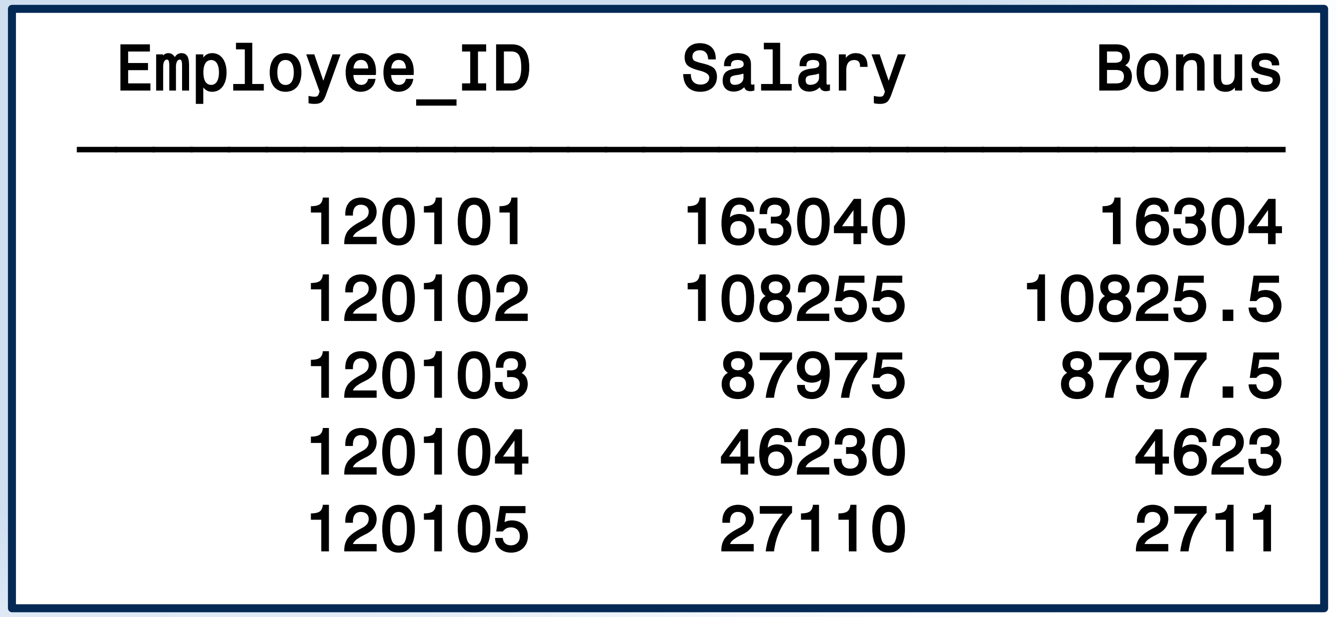


```
                    The SAS System

                                               Employee
                         Employee                Annual
 Employee ID             Gender                  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
```

Business Scenario

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





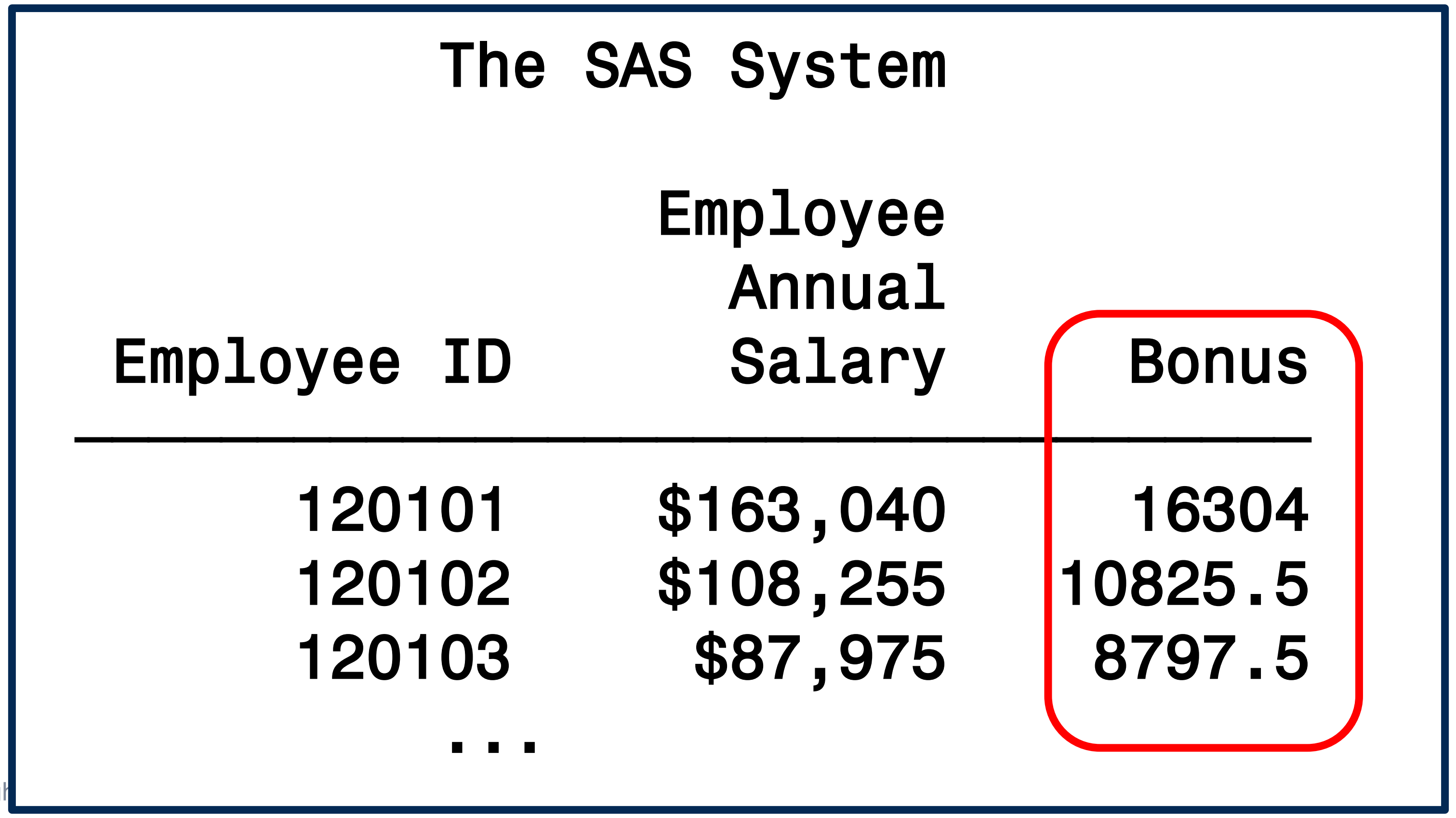
```
  Employee_ID      Salary       Bonus
  ___________________________________

       120101      163040       16304
       120102      108255     10825.5
       120103       87975      8797.5
       120104       46230        4623
       120105       27110        2711
```

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
          ...
```

s102d07

Questions??

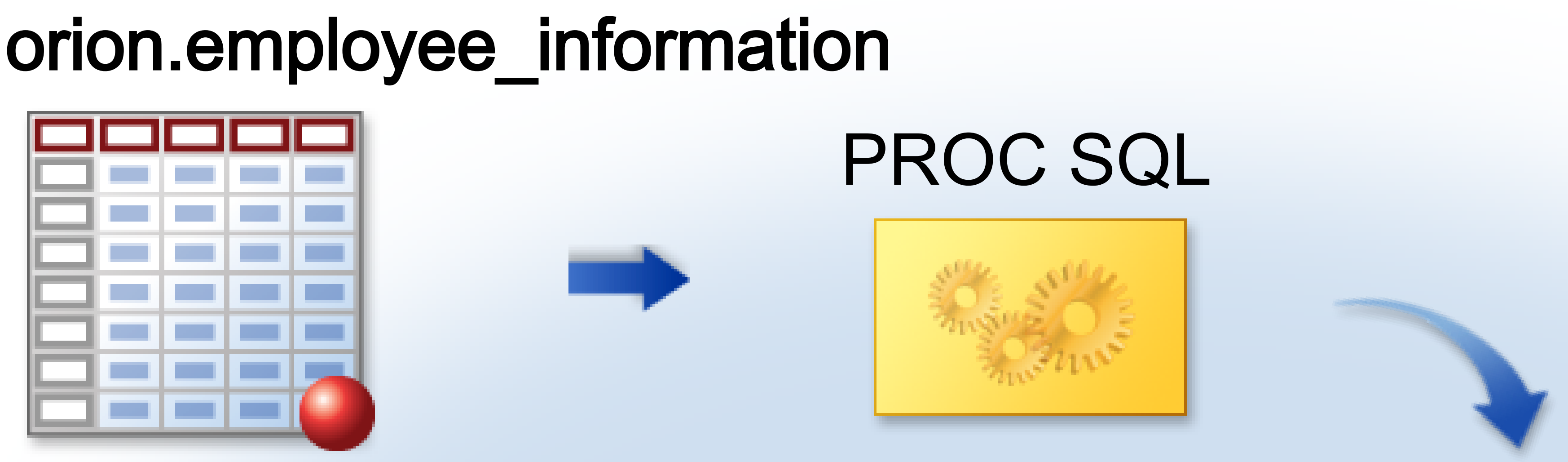
Specifying Rows

Objectives

- Select a subset of rows in a query.

Business Scenario

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



orion.employee_information

PROC SQL

```
                                                  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
```

Subsetting with the WHERE Clause

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

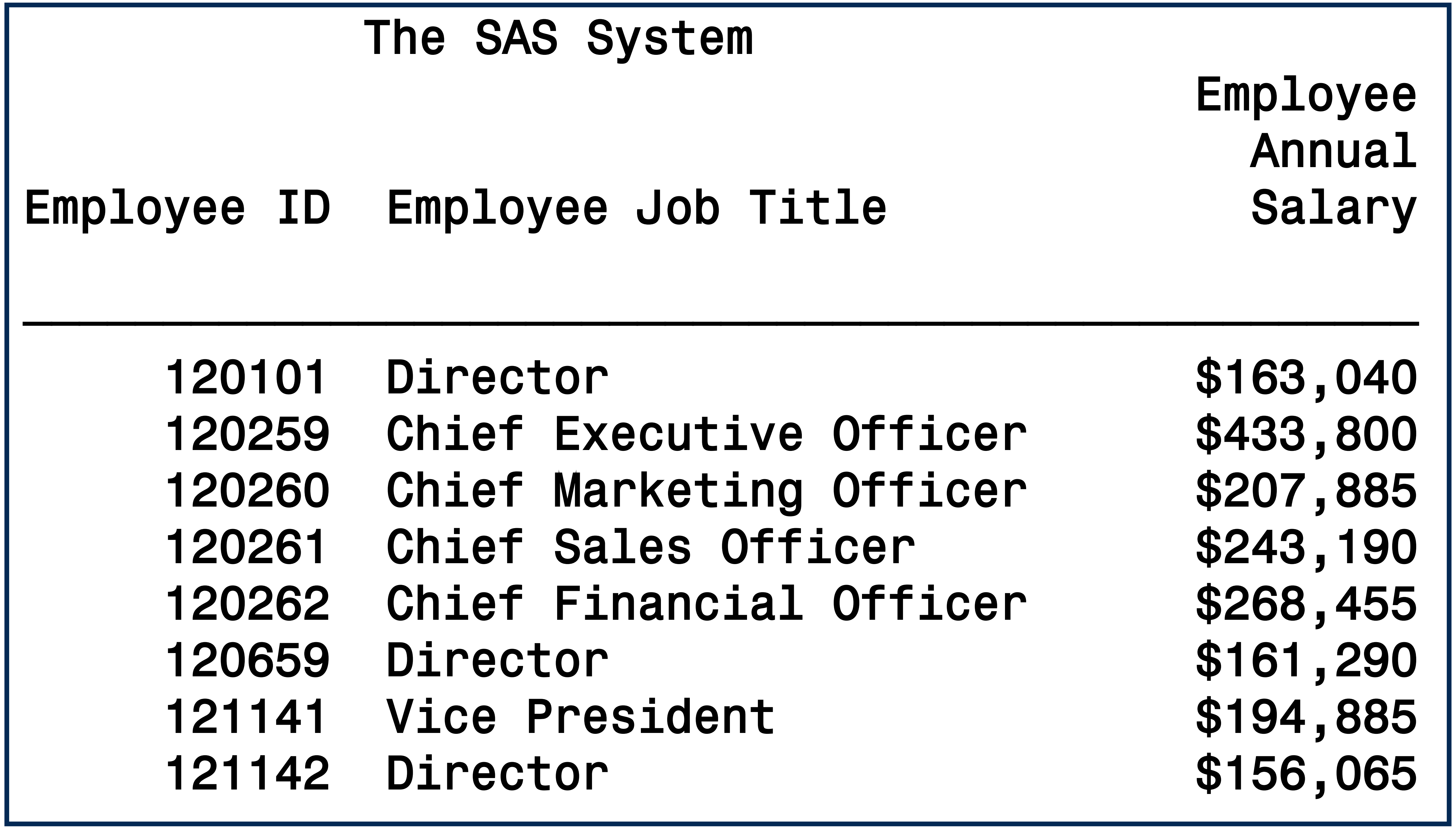
```
proc sql;
select Employee_ID, Job_Title, Salary
   from orion.employee_information
   where Salary > 112000;
quit;
```

WHERE *sql-expression*

s102d14

Viewing the Output

PROC SQL Output

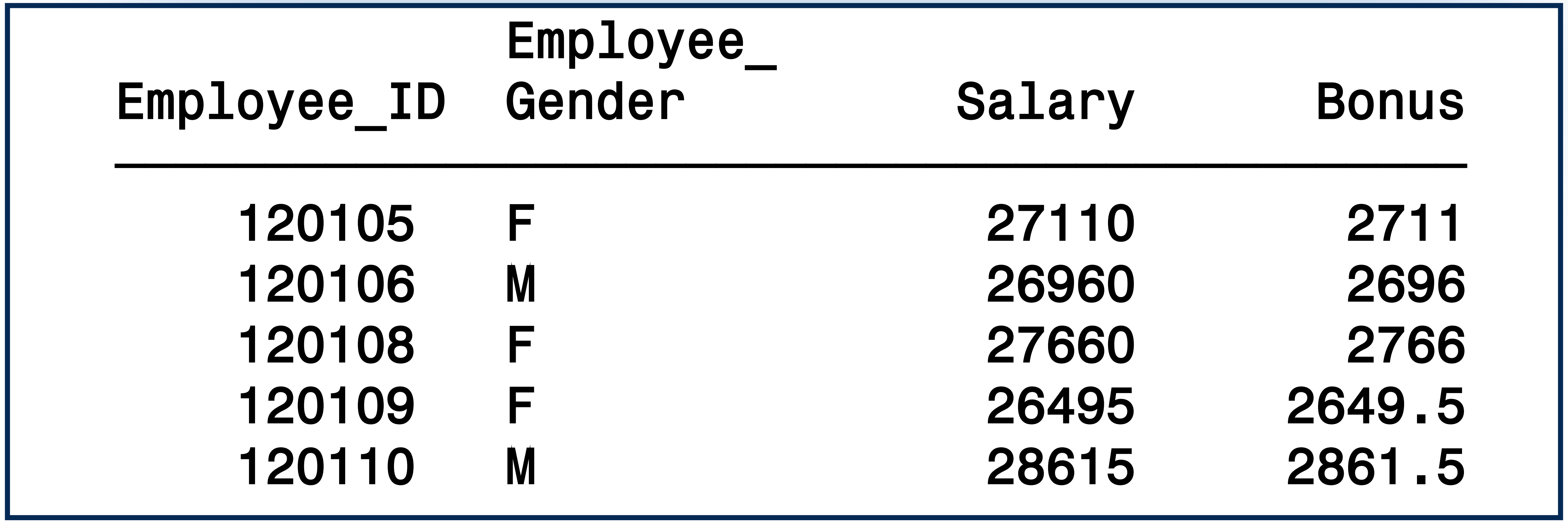


```
                 The SAS System
                                                         Employee
                                                           Annual
Employee ID  Employee Job Title                            Salary

__________________________________________________________________

     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
```

Business Scenario

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

orion.employee_information
PROC SQL



```
               Employee_
  Employee_ID  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
```

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

Subsetting with Calculated Values

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

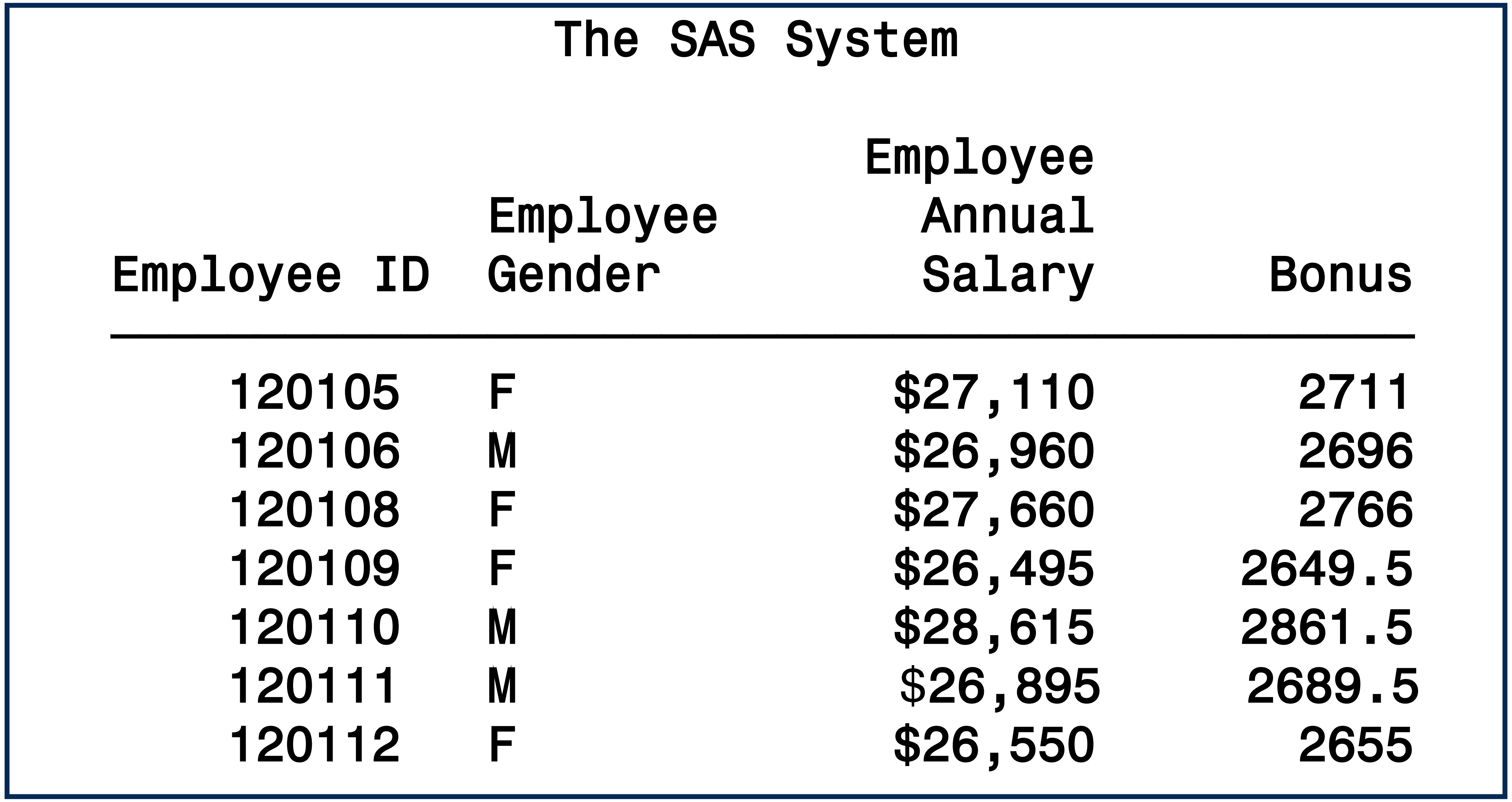
```
proc sql;
select Employee_ID, Employee_Gender,
       Salary, Salary*.10 as Bonus
   from orion.employee_information
   where calculated Bonus<3000;
quit;
```

SAS enhancement

s102d15

Viewing the Output

Partial PROC SQL Output



```
                        The SAS System

                                   Employee
                   Employee          Annual
     Employee ID   Gender            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
```

Questions??

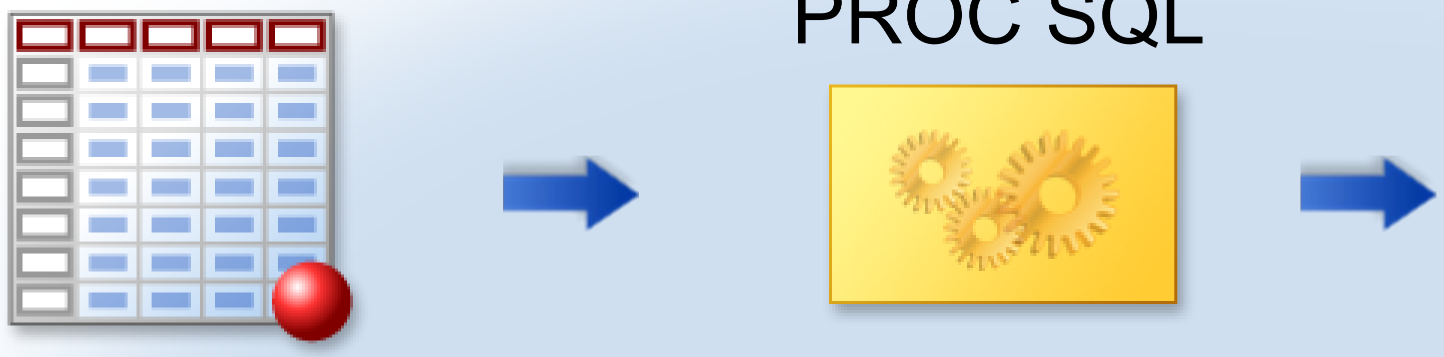
Summarizing Data

Objectives

- Group data and produce summary statistics for each group.
- Subset a query on summarized values.

Business Scenario

Management requested a report containing the total annual donations for each employee.



Partial Results

```
 Employee      Annual
Identifier    Donation
-----------------------
120736        $45.00
120759        $40.00
120681        $40.00
```

Summary Functions: Across a Row

Total each employee’s annual cash donations.

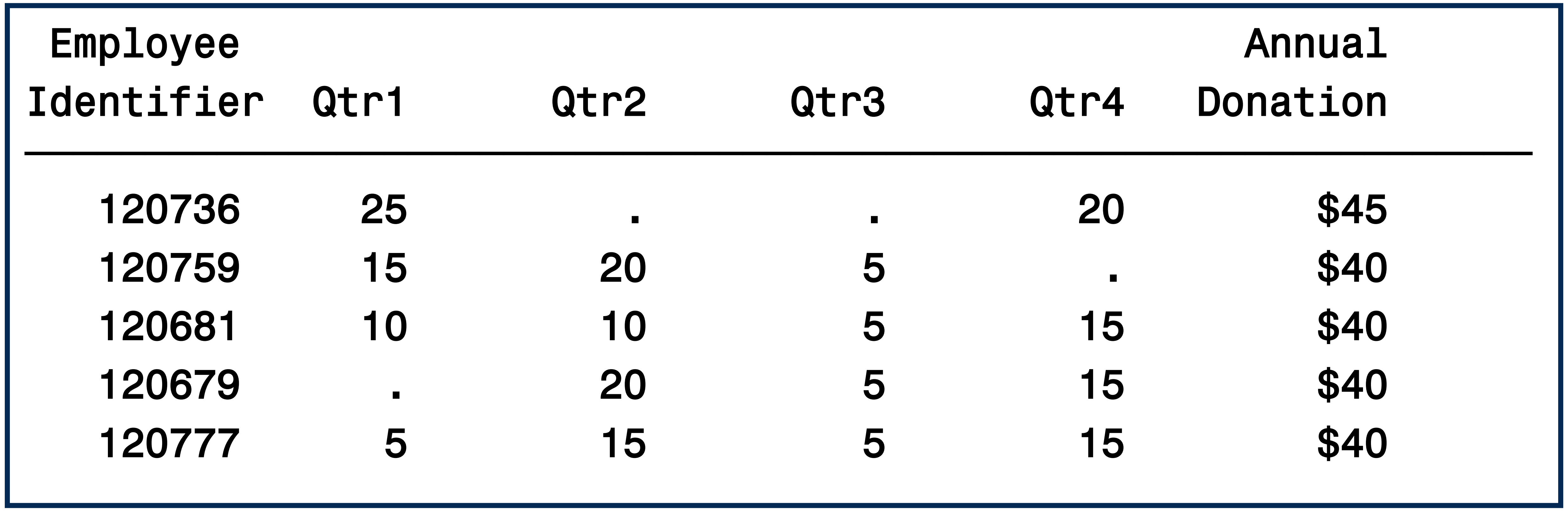
SUM(*col1*, …, *coln*)

```
proc sql;
select Employee_ID
       label='Employee Identifier',
       Qtr1,Qtr2,Qtr3,Qtr4,
       sum(Qtr1,Qtr2,Qtr3,Qtr4)
      label='Annual Donation'
      format=dollar5.
   from orion.employee_donations
   where Paid_By="Cash or Check"
   order by 6 desc;
quit;
```

s103d06

Viewing the Output

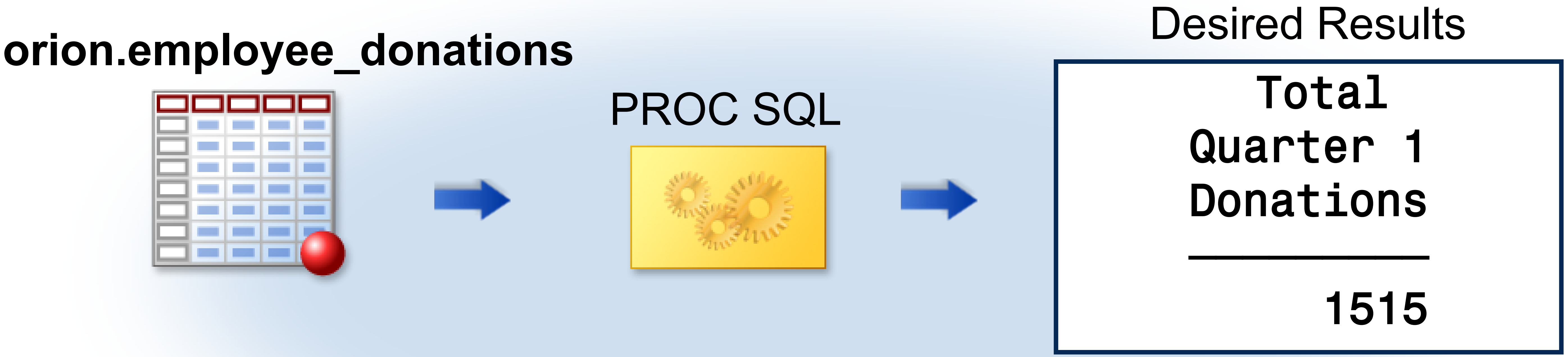
Partial PROC SQL Output



```
 Employee                                          Annual
Identifier  Qtr1       Qtr2       Qtr3       Qtr4  Donation
-------------------------------------------------------------

   120736     25          .          .         20       $45
   120759     15         20          5          .       $40
   120681     10         10          5         15       $40
   120679      .         20          5         15       $40
   120777      5         15          5         15       $40
```

Business Scenario

Management requested a report containing the total contributions for all employees in the first quarter.



orion.employee_donations

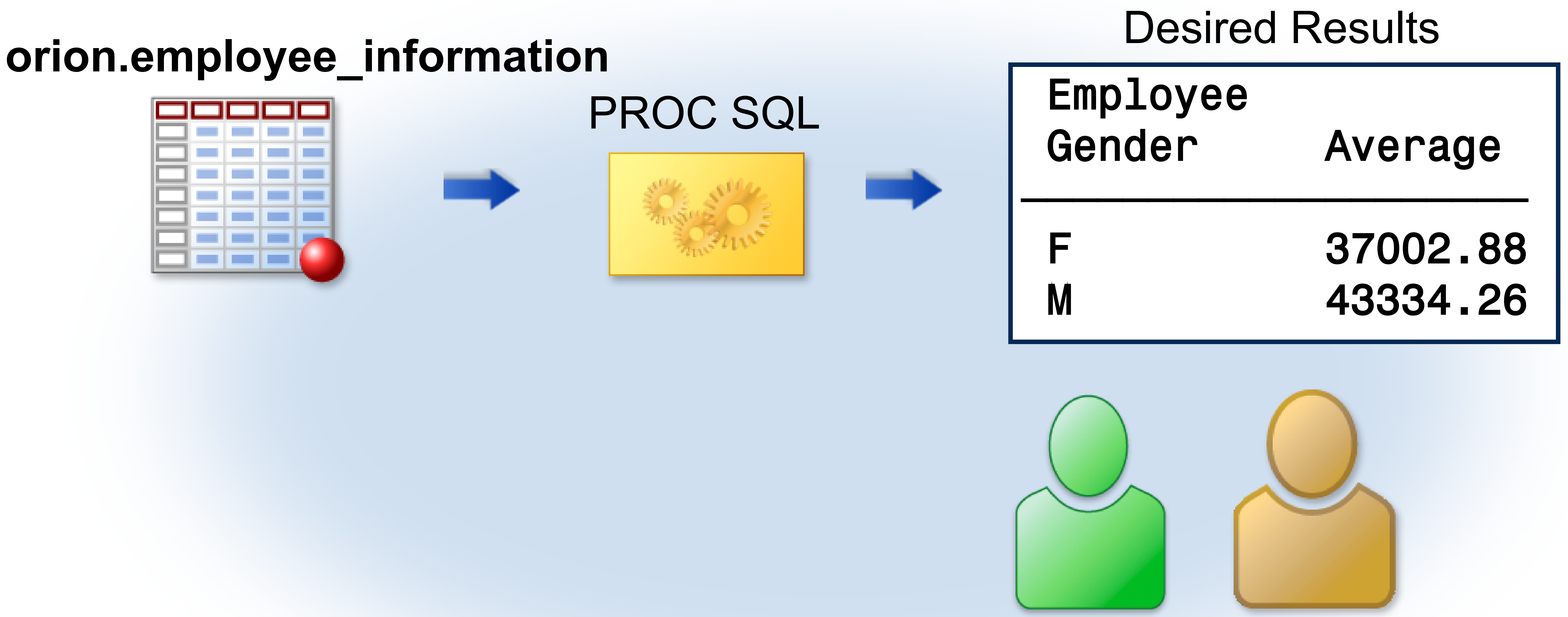
PROC SQL

Desired Results

```
      Total
  Quarter 1
  Donations
  _________

       1515
```

Business Scenario

Produce a report that determines the average salary by gender.



```
Employee
Gender        Average
__________________________
F             37002.88
M             43334.26
```

Grouping Data

You can use the GROUP BY clause to do the following:

- classify the data into groups based on the values of one or more columns
- calculate statistics for each unique value of the grouping columns

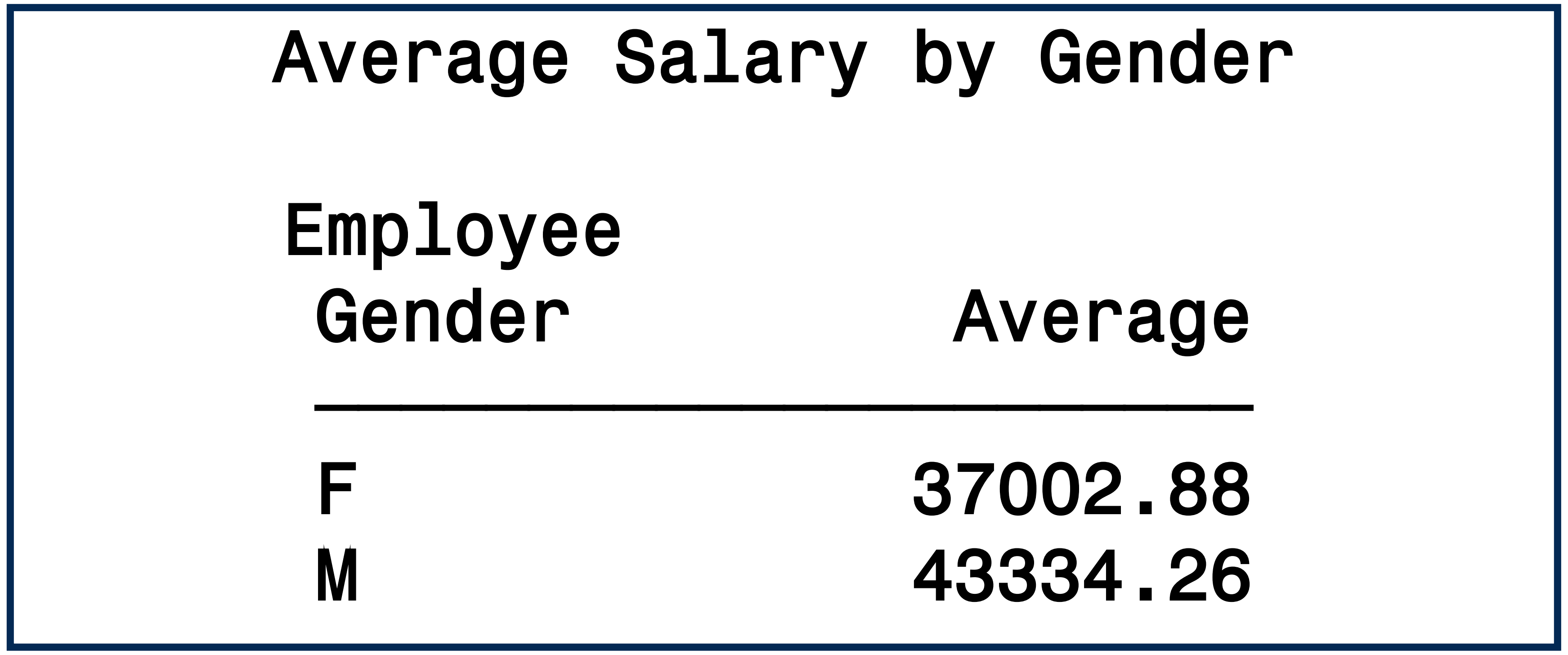
```
proc sql;
title "Average Salary by Gender";
select Employee_Gender as Gender,
       avg(Salary) as Average
   from orion.employee_information
   where Employee_Term_Date is missing
   group by Employee_Gender;
quit;
```

GROUP BY *group-by-item<,…, group-by-item>*

s103d10

Viewing the Output

PROC SQL Output



```
     Average Salary by Gender

      Employee
       Gender               Average
       ____________________________

       F                   37002.88
       M                   43334.26
```

Questions??

Joining Tables

Objectives

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

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 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.

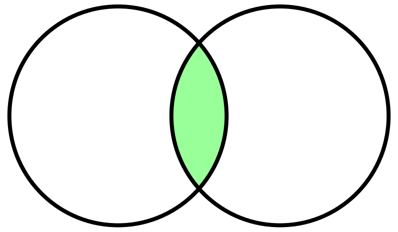
Objectives

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

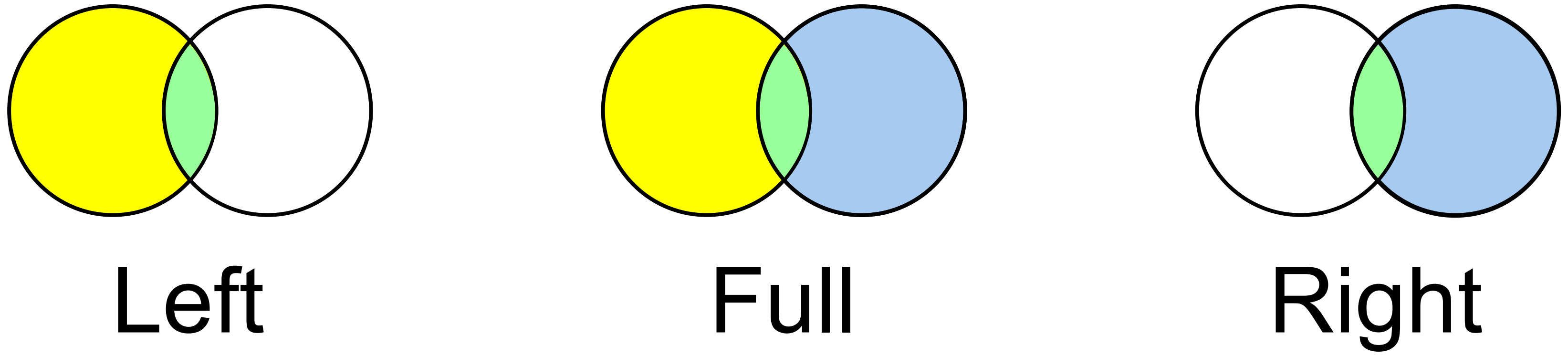
Types of Joins

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.



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*>;

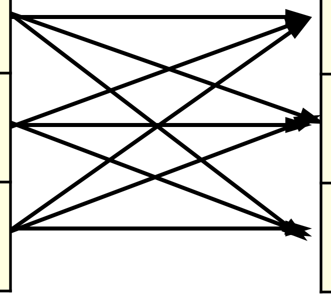
To understand how SQL processes a join, it is helpful to understand the concept of the Cartesian product.

s104d01

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                                   $52
102                                  $212
```

The Cartesian product is rarely the desired result

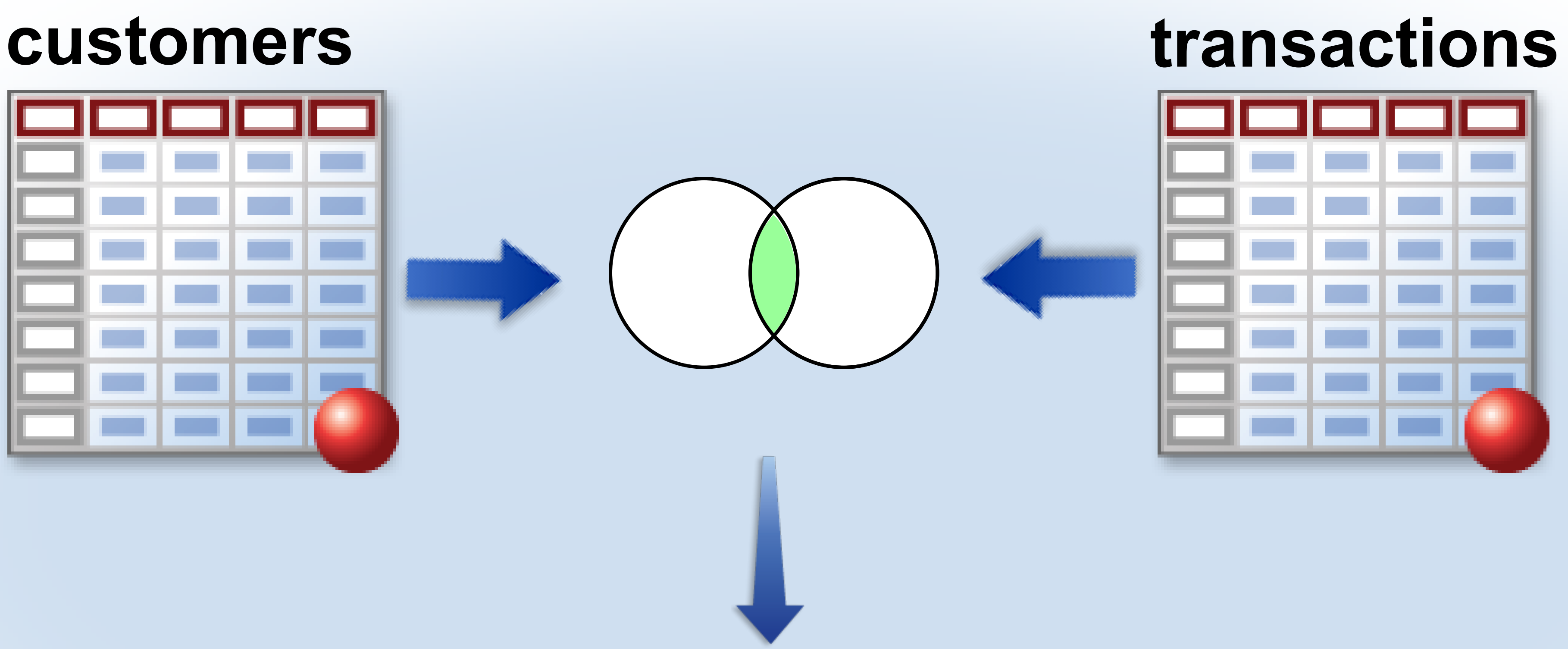
Questions??

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.

Report 1: Inner Join

Management has requested a report showing all valid order information.



ID	Name	Action	Amount
102	Blank	Purchase	$100

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 SQ

```
ID  Name    ID  Action      Amount
----------------------------------
102 Blank   102 Purchase      $100
```

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

Abbreviating the Code with a Table Alias

```
proc sql;
select c.ID, Name, Action, Amount
   from customers as c, transactions as t
   where c.ID=t.ID;
quit;
```

PROC SQL Output

```
 ID  Name   Action        Amount
--------------------------------
102  Blank  Purchase        $100
```

s104d04

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

Questions??

Handy Links

- SAS 9.4 Proc sql user's guide
- Video - Step-by-step PROC SQL
- Go home on time with 5 PROC SQL tips
- Video - Mastering the WHERE clause in PROG SQL
- Video - Power of SAS SQL –SAS Global Forum 2021
- Video - Step by step PROC SQL – SAS Global forum 2020
- Know thy data: Dictionary tables SAS Global Forum Paper

Handy Links

- SAS 9.4 PROC SQL user's guide
- Video - Step-by-step PROC SQL
- Go home on time with 5 PROC SQL tips
- Ask The Expert Webinar – Top 5 Handy PROC SQL Tips
- SAS YouTube Video - Mastering the WHERE clause in PROG SQL
- SAS YouTube Video - Power of SAS SQL –SAS Global Forum 2021
- SAS YouTube Video - Step by step PROC SQL – SAS Global forum 2020
- Know thy data: Dictionary tables SAS Global Forum Paper
- "Ask the Expert Webinar - Why choose between SAS data Step & PROC SQL When You Can Have Both

Recommended Courses From This Presentation

- SAS®SQL 1: Essentials

Thank You

✓ Did you enjoy this session, Let us know in the evaluation

Charu Shankar
SAS Institute Toronto

EMAIL	Charu.shankar@sas.com
BLOG	https://blogs.sas.com/content/author/charushankar/
TWITTER	CharuYogaCan
LINKEDIN	https://www.linkedin.com/in/charushankar/