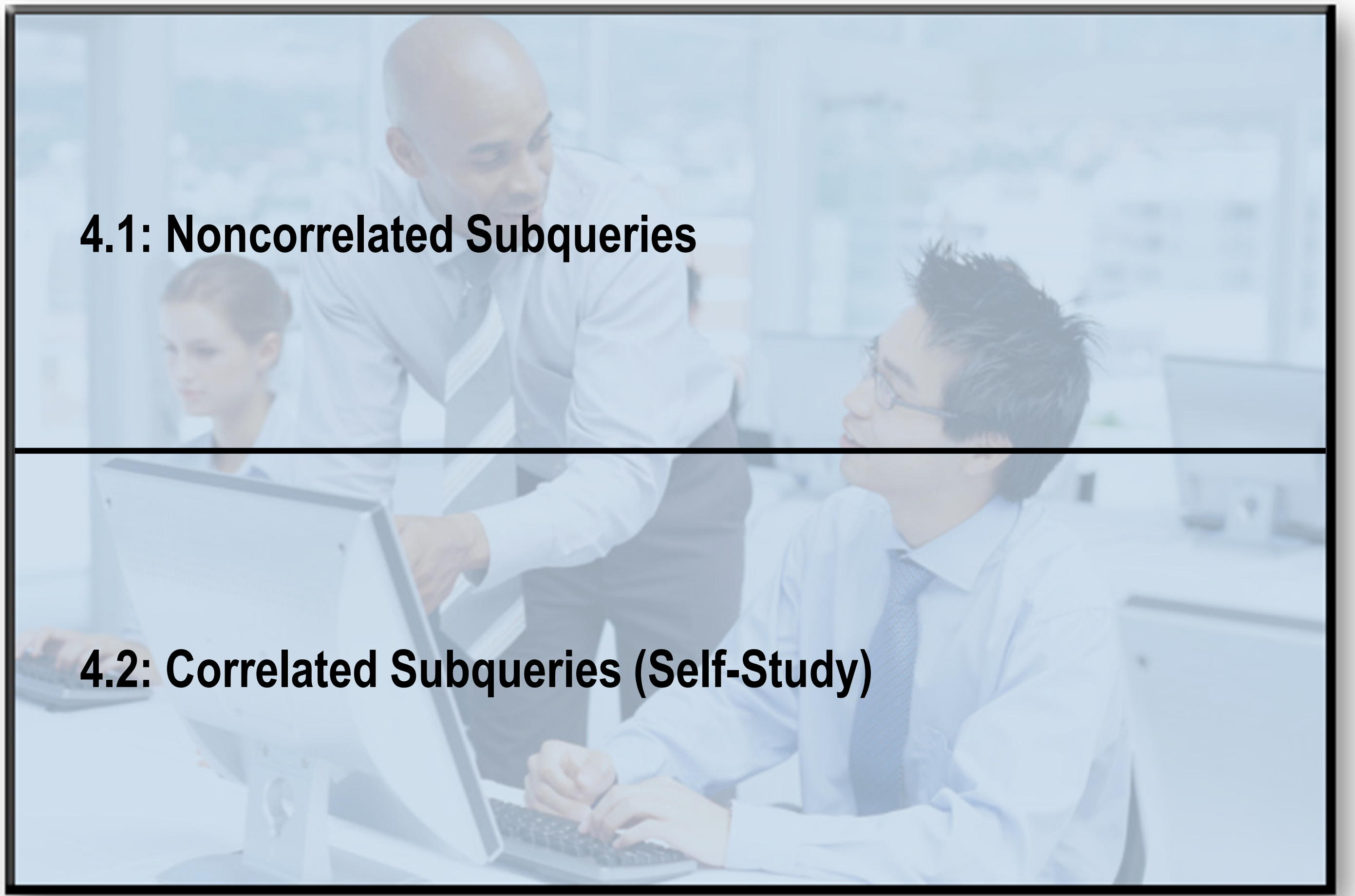
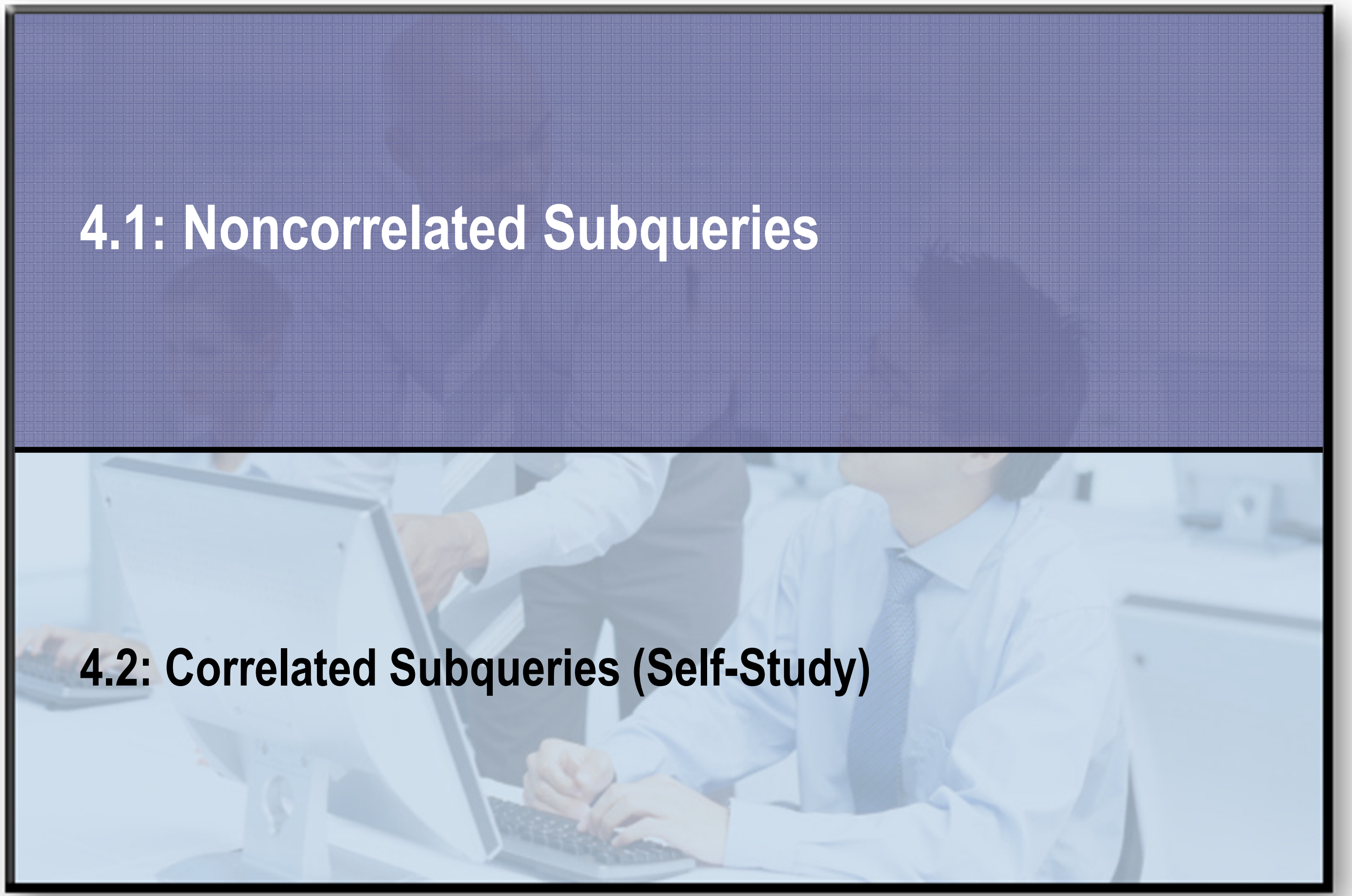
Chapter 4: Subqueries



Chapter 4: Subqueries



Objectives

- Define PROC SQL subqueries.
- Differentiate between correlated and noncorrelated subqueries.
- Subset data based on values returned a subquery.

Queries versus Subqueries

A query corresponds to a single SELECT statement within a PROC SQL step.

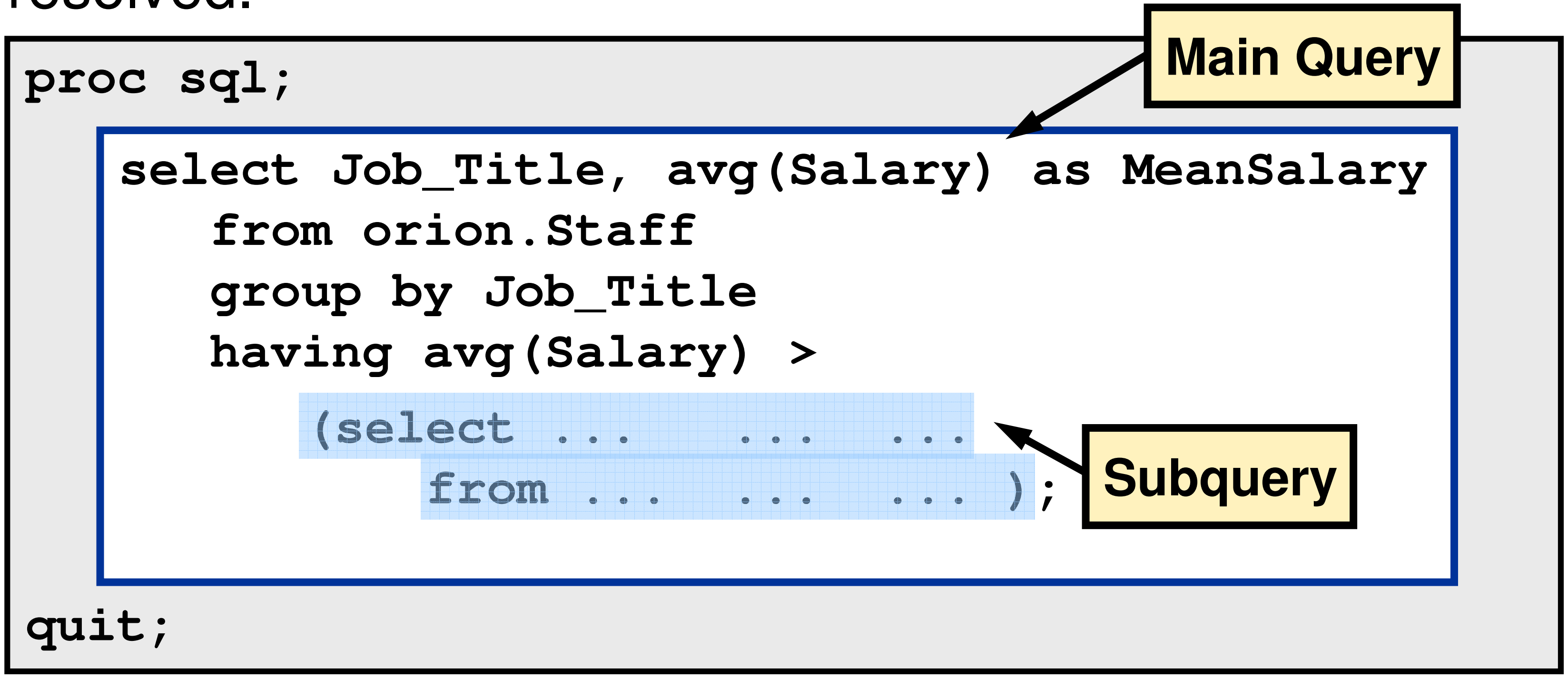
```
proc sql;
    select *
       from orion.Staff;

    select avg(Salary) as MeanSalary
       from orion.Staff;

    select Job_Title, avg(Salary) as MeanSalary
       from orion.Staff
       group by Job_Title
       having avg(Salary) > 38041.51;
quit;
```

Queries versus Subqueries

A *subquery* is a query (SELECT statement) that resides within an outer query (the main SELECT statement). The subquery must be resolved before the main query can be resolved.



```
proc sql;
   select Job_Title, avg(Salary) as MeanSalary
      from orion.Staff
      group by Job_Title
      having avg(Salary) >
         (select ...     ...     ...
            from ...     ...     ... );
quit;
```

Subqueries

Subqueries

- return values to be used in the outer query's WHERE or HAVING clause
- can return single or multiple values
- must return only a single column.

Subqueries

There are two types of subqueries:

- In a *noncorrelated subquery*, values are passed from the inner query to the outer query.

```
proc sql;

   select Job_Title, avg(Salary) as MeanSalary
      from orion.Staff
      group by Job_Title
      having avg(Salary) >
       (select avg(Salary) as MeanSalary
           from orion.Staff);

quit;
```

Stand-alone query

continued...

Subqueries

- In a *correlated subquery*, the outer query must provide information to the subquery before it can be successfully resolved.

```
proc sql;
   select Employee_ID, avg(Salary) as MeanSalary
      from orion.Employee_Addresses
      where 'AU'=
        (select Country
             from Work.Supervisors
             where Employee_Addresses.Employee_ID=
                   Supervisors.Employee_ID);
quit;
```

Business Scenario

Create a report that displays `Job_Title` for job groups with an average salary greater than the average salary of the company as a whole.

Using a Noncorrelated Subquery

This demonstration illustrates how to write a noncorrelated subquery.

s104d01

Noncorrelated Subqueries

```
proc sql;
   select Job_Title,
          avg(Salary) as MeanSalary
      from orion.Staff
      group by Job_Title
      having avg(Salary) >
         (select avg(Salary)
            from orion.Staff)
;
quit;
```

Evaluate the subquery first.

s104d01

Noncorrelated Subqueries

```
proc sql;
   select Job_Title,
          avg(Salary) as MeanSalary
      from orion.Staff
      group by Job_Title
      having avg(Salary) > (38041.51)
;
quit;
```

Then pass
the results
to the outer query.

s104d01

Noncorrelated Subqueries

Partial PROC SQL Output

```
Employee Job Title                 MeanSalary
_____________________________________________

Account Manager                         46090
Administration Manager                  47415
Applications Developer I                42760
Applications Developer II               47315
Applications Developer IV            55751.67
Auditing Manager                        53400
Auditor I                               42190
Auditor II                              46545
Auditor III                             51950
BI Administrator IV                     58530
BI Architect II                         47155
BI Specialist II                        44425
```

Poll
Quiz

4.01 Poll

Can a subquery contain a subquery?

❍ Yes

❍ No

4.01 Poll – Correct Answer

Can a subquery contain a subquery?

[✓ ❍ Yes]

❍ No

Business Scenario

Each month, the CEO sends a birthday card to each employee having a birthday in that month.

Create a report listing the names and addresses of employees with February birthdays.

Noncorrelated Subqueries

The **`orion.Employee_Addresses`** table contains names and addresses. Birth dates are found in the **`orion.Employee_Payroll`** table.

```
proc sql;
   select Employee_ID,
          Employee_Name, City,
          Country
      from orion.Employee_Addresses
      where Employee_ID in
        (select Employee_ID
            from orion.Employee_Payroll
            where month(Birth_Date)=2)
      order by 1;
quit;
```

s104d02

Noncorrelated Subqueries: How Do They Work?

```
proc sql;
   select Employee_ID,
          Employee_Name, City,
          Country
      from orion.Employee_Addresses
      where Employee_ID in
        (select Employee_ID
            from orion.Employee_Payroll
            where month(Birth_Date)=2)
      order by 1;
quit;
```

Partial orion.Employee_Payroll

Employee _ID	Birth _Date
…	…
120106	23DEC1948
120107	21JAN1953
120108	23FEB1988
120109	15DEC1990
120110	20NOV1953
120111	23JUL1953
120112	17FEB1973
120113	10MAY1948
…	…

Step 1: Evaluate the inner query and build a virtual table that satisfies the WHERE criteria.

s104d02

...

Noncorrelated Subqueries: How Do They Work?

```
proc sql;
   select Employee_ID,
          Employee_Name, City,
          Country
      from orion.Employee_Addresses
      where Employee_ID in
        (120108,120112,120114,120157,
         120159, 120170,…)
      order by 1;
quit;
```

Values returned by the inner query

Partial orion.Employee_Payroll

Employee_ID	Birth_Date
…	…
120106	23DEC1948
120107	21JAN1953
120108	23FEB1988
120109	15DEC1990
120110	20NOV1953
120111	23JUL1953
120112	17FEB1973
120113	10MAY1948
…	…

s104d02

...

Noncorrelated Subqueries: How Do They Work?

```
proc sql;
   select Employee_ID,
          Employee_Name, City,
          Country
      from orion.Employee_Addresses
      where Employee_ID in
        (120108,120112,120114,120157,
         120159, 120170,…)
      order by 1;
quit;
```

Partial orion.Employee_Payroll

Employee _ID	Birth _Date
…	…
120106	23DEC1948
120107	21JAN1953
120108	23FEB1988
120109	15DEC1990
120110	20NOV1953
120111	23JUL1953
120112	17FEB1973
120113	10MAY1948
…	…

Step 2: Pass the values to the outer query for use in the WHERE clause.

s104d02

Noncorrelated Subqueries: Output

```
                              The SAS System


Employee_ID   Employee_Name                City                  Country
------------------------------------------------------------------------
     120108   Gromek, Gladys                Melbourne             AU
     120112   Glattback, Ellis              Melbourne             AU
     120114   Buddery, Jeannette            Sydney                AU
     120157   Karavdic, Leonid              Sydney                AU
     120159   Phoumirath, Lynelle           Sydney                AU
     120170   Kingston, Alban               Sydney                AU
```

Do these look familiar? They are the employee IDs returned by the inner query.

Poll

Quiz

Setup for the Poll

- Open the program **s104a01**.
- Change the IN operator to an equal sign (=) in the code as shown on the previous slide.
- Run the changed program and review the SAS log for messages.
- What happens when you change the comparison operator to an equal sign?

```
proc sql;
   select Employee_Name, City, Country
      from orion.Employee_Addresses
      where Employee_ID in
        (select Employee_ID
            from orion.Employee_Payroll
            where month(Birth_Date)=2)
      order by 1;
quit;
```

4.02 Multiple Choice Poll

What happens when you change the comparison operator to an equal sign?

a. Nothing special; the program runs fine.
b. You get multiple rows returned in your output.
c. You get an error message.
d. a and b.

4.02 Multiple Choice Poll – Correct Answer

What happens when you change the comparison operator to an equal sign?

a. Nothing special; the program runs fine.
b. You get multiple rows returned in your output.
[✓ c.] You get an error message.
d. a and b.

Subqueries That Return Multiple Values

When a subquery returns multiple values and the EQUAL operator is used, an ERROR message is generated. The EQUAL operator does not accept any expression that resolves to more than a single value.

Example:

```
where Employee_ID=120108, 120112, 120114...
```

```
ERROR: Subquery evaluated to more than one row.
```

✎ If the subquery returns multiple values, you must use the IN operator or a comparison operator with the ANY or ALL keywords.

The ANY Keyword (Self-Study)

If you specify the ANY keyword before a subquery, the comparison is true if it is true for any of the values that the subquery returns.

Keyword ANY	Signifies…
= ANY(20,30,40)	=20 or =30 or =40
> ANY(20,30,40)	> 20
< ANY(20,30,40)	< 40

✐ The values 20,30,40 represent values returned from a subquery.

The ANY Keyword (Self-Study)

Example: Do any Level IV sales representatives have a salary that is lower than any of the lower-level sales representatives?

```
proc sql;
select Employee_ID, Salary
   from orion.Staff
   where Job_Title='Sales Rep. IV'
         and salary < any
     (select Salary
         from orion.Staff
         where Job_Title in
               ('Sales Rep. I','Sales Rep. II',
                'Sales Rep. III'));
quit;
```

Think < select max(salary).

s104d03

The ANY Keyword (Self-Study)

Partial PROC SQL Output

```
   Level IV Sales Reps Who Earn Less Than
        Any Lower Level Sales Rep

                             Employee
                               Annual
        Employee ID            Salary
      ________________________________

             120125           $32,040
             120128           $30,890
             120135           $32,490
             120159           $30,765
             120166           $30,660
             121019           $31,320
             121020           $31,750
```

The ALL Keyword (Self-Study)

The ALL keyword is true only if the comparison is true for all returned values.

Keyword ALL	Signifies
> ALL(20,30,40)	> 40
< ALL(20,30,40)	< 20

✎ The values 20,30,40 represent values returned from a subquery.

The ALL Keyword (Self-Study)

Example: What are the job titles and salaries of the employees who earn more than all Level IV employees?

```
proc sql;
   select Job_Title, Salary
      from orion.Staff
      where Salary > all
        (select Salary
           from orion.Staff
           where Job_Title contains 'IV');

quit;
```

Think > select max(salary).

s104d04

Selecting Data (Self-Study)

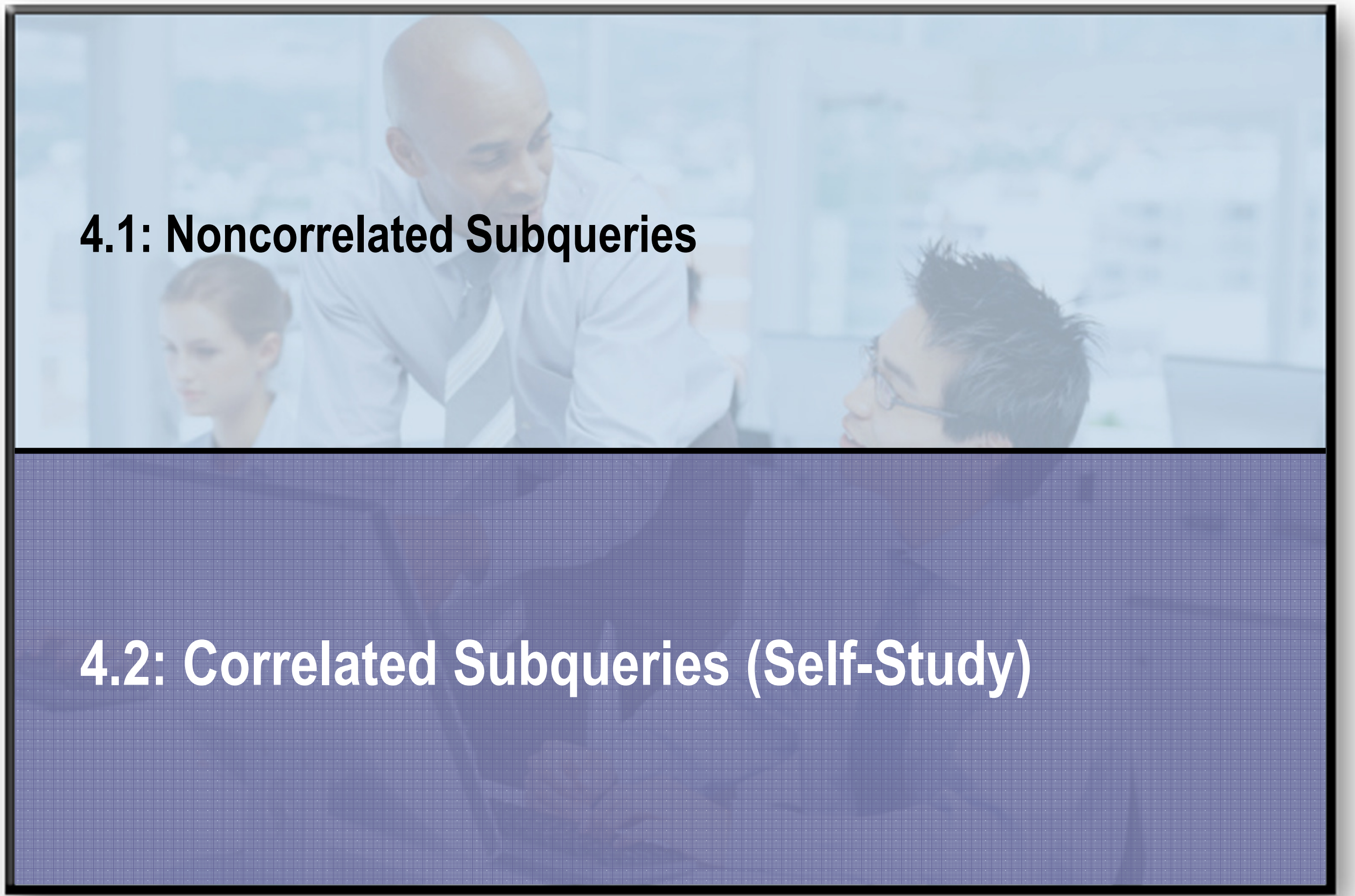
Partial PROC SQL Output

```
     Job Titles and Salaries of Employees
               That Earn More Than
             All level IV Employees

                                  Employee
                                     Annual
Employee Job Title                   Salary
___________________________________________

Director                           $163,040
Sales Manager                      $108,255
Sales Manager                       $87,975
Chief Executive Officer            $433,800
Chief Marketing Officer            $207,885
Chief Sales Officer                $243,190
Chief Financial Officer            $268,455
Senior Strategist                   $76,105
```

Chapter 4: Subqueries



4.1: Noncorrelated Subqueries

4.2: Correlated Subqueries (Self-Study)

Objectives

- Define correlated subqueries.
- Describe how data is subset using correlated subqueries.

Correlated Subqueries

Correlated subqueries

- cannot be evaluated independently
- require values to be passed to the inner query from the outer query
- are evaluated for each row in the outer query.

Business Scenario

Create a report listing the employee identifier and the first name followed by the last name for all managers in Australia.

Considerations:

- You have a temporary table, **`Supervisors`**, containing **`Employee_ID`** and **`Country`** for all managers.
- The table **`orion.Employee_Addresses`** contains **`Employee_Name`** for all employees, but the names are stored as Last, First.
- You used SCAN() to separate first and last names before. Now you need a new technique to concatenate the pieces into First, Last order.

The CATX Function

The CATX function concatenates the values in *argument-1* through *argument-n* by stripping leading and trailing spaces, and inserting the value of *argument-1* between each segment.

General form of the CATX function:

CATX(*delimiter*,*argument-1*,*argument-2*<, ...*argument-n*>)

delimiter	a character string that is used as a delimiter between concatenated arguments.
argument	a character variable's name, a character constant, or an expression yielding a character value.

The CATX Function

Example:

```
proc sql;
   select catx(' ',First_name,Last_name)
          format=$25. as Name
      from orion.Sales
      where First_name="John"
;
quit;
```

PROC SQL Output

```
Name
______________

John Filo
John Kirkman
John Hoppmann
```

s104d04a

Correlated Subqueries

In a correlated subquery, the outer query provides information so that the subquery resolves successfully.

```
proc sql;
   select Employee_ID,
          catx(' ',scan(Employee_Name,2),
          scan(Employee_Name,1) as Manager_Name
          length=25
      from orion.Employee_Addresses
      where 'AU'=
         (select Country
             from Work.Supervisors
             where Employee_Addresses.Employee_ID=
                   Supervisors.Employee_ID);
quit;
```

You must qualify each column with a table name.

s104d05

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Step 1: The outer query takes the first row in `orion.Employee_Addresses` and finds `Employee_ID` and `Employee_Name`.

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Step 2: In the subquery, try to match `Employee_Addresses.Employee_ID` of 120145 with the value of `Supervisors.Employee_ID` to find a qualifying row in `Work.Supervisors`.

NO MATCH

Partial Listing of
`orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
   select Employee_ID,
          catx(' ',scan(Employee_Name,2),
          scan(Employee_Name,1) as Manager_Name
          length=25
      from orion.Employee_Addresses
      where 'AU'=
        (select Country
           from Work.Supervisors
           where Employee_Addresses.Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Steps 1 and 2 (Repeat):
Read the next row from `orion.Employee_Addresses` and pass the corresponding employee ID to the subquery to look for a matching employee ID in `Work.Supervisors`.
There is a match.

MATCH

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Coun
          from Wor
          where Em                    Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'US'

Partial Listing of
orion.Employee_Addresses

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

Work.Supervisors

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='US'`) resolves to FALSE.

FALSE

Poll

Quiz

4.03 Quiz

Given the following query, subquery, and data in **`Work.Supervisors`**, what is the maximum number of rows that will be selected by the outer query?

```
proc sql;
   select Employee_ID,
          catx(' ',scan(Employee_Name,2),
          scan(Employee_Name,1) as Manager_Name
          length=25
      from orion.Employee_Addresses
      where 'AU'=
        (select Country
           from Work.Supervisors
           where Employee_Addresses.Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Work.Supervisors

Employee _ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

4.03 Quiz – Correct Answer

Given the following query, subquery, and data in **`Work.Supervisors`**, what is the maximum number of rows that will be selected by the outer query?

Only the three managers where `Country='AU'` would be selected.

```
proc sql;
   select Employee_ID,
          catx(' ',scan(Employee_Name,2),
          scan(Employee_Name,1) as Manager_Name
          length=25
      from orion.Employee_Addresses
      where 'AU'=
        (select Country
           from Work.Supervisors
           where Employee_Addresses.Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

The Outer Query Controls the Result Set

The outer query determines which rows cause the inner query to resolve successfully.

```
proc sql;
   select Employee_ID,
          catx(' ',scan(Employee_Name,2),
          scan(Employee_Name,1) as Manager_Name
          length=25
      from orion.Employee_Addresses
      where 'AU'=
         (select Country
             from Work.Supervisors
             where Employee_Addresses.Employee_ID=
                   Supervisors.Employee_ID) ;
quit;
```

Work.Supervisors

Employee _ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Continue repeating steps 1, 2, and 3 until all `orion.Employee_Addresses` rows are read.

`Employee_ID` 120656 has no match.

NO MATCH

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Continue repeating steps 1, 2, and 3 until all rows are read from `orion.Employee_Addresses`. For `Employee_ID` 120104, which is passed from the main query to the subquery, there is a match.

MATCH

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Count
          from Work
          where Emp                mployee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'AU'

Partial Listing of
`orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='AU'`) resolves to TRUE.

TRUE

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Step 4: Write `Employee_ID` and `Manager_Name` from `orion.Employee_Addresses` as the first row in a newly created report.

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

Build the first row of the report:

```
Employee_ID   Manager_Name
--------------------------------------------
     120104   Kareen Billington
```

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Continue repeating steps 1, 2, and 3 until all `orion.Employee_Addresses` rows are read.

`Employee_ID` 121035 has no match.

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

NO MATCH

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Steps 1 and 2 (repeated):
Read the next row from `orion.Employee_Addresses` and pass the corresponding employee ID to the subquery to look for a matching employee ID in `Work.Supervisors`. There is a match.

MATCH

Partial Listing of
`orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Cour
           from Wor
           where Em                 Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'US'

Partial Listing of orion.Employee_Addresses

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

Work.Supervisors

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='US'`) resolves to FALSE.

FALSE

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Steps 1 and 2 (repeated):
Read the next row from `orion.Employee_Addresses` and pass the corresponding employee ID to the subquery to look for a matching employee ID in `Work.Supervisors`. There is a match.

MATCH

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Cour
           from Wor
           where Em                  Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'US'

Partial Listing of orion.Employee_Addresses

Employee _ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

Work.Supervisors

Employee _ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='US'`) resolves to FALSE.

FALSE

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
           from Work.Supervisors
           where Employee_Addresses.Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Continue repeating steps 1, 2, and 3 until all rows are read from `orion.Employee_Addresses`. For `Employee_ID` 120103, which is passed from the main query to the subquery, there is a match.

MATCH

Partial Listing of
`orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Cou
          from Wo
          where Em                   Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'AU'

Partial Listing of orion.Employee_Addresses

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

Work.Supervisors

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='AU'`) resolves to TRUE.

TRUE

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Step 4: Write `Employee_ID` and `Manager_Name` from `orion.Employee_Addresses` as the second row in the report.

Partial Listing of `orion.Employee_Addresses`

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

Build the second row of the report:

```
Employee_ID   Manager_Name
------------------------------------------
     120104   Kareen Billington
     120103   Wilson Dawes
```

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
           from Work.Supervisors
           where Employee_Addresses.Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Continue repeating steps 1, 2, and 3 until all rows are read from `orion.Employee_Addresses`. For `Employee_ID` 120103, which is passed from the main query to the subquery, there is a match.

MATCH

Partial Listing of
`orion.Employee_Addresses`

Employee _ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee _ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Coun
           from Wor
           where Em                Employee_ID=
                 Supervisors.Employee_ID) ;
quit;
```

Subquery returns 'AU'

Partial Listing of `orion.Employee_Addresses`

Employee _ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

`Work.Supervisors`

Employee _ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Step 3: The subquery passes the value of `Country` from the selected row in `Work.Supervisors` back to the outer query, where the = operator compares this value to `'AU'` for selection in the main query.
In this case, the main query WHERE expression (`where 'AU'='AU'`) resolves to TRUE.

TRUE

Correlated Subqueries

```
proc sql;
  select Employee_ID,
         catx(' ',scan(Employee_Name,2),
         scan(Employee_Name,1) as Manager_Name
         length=25
     from orion.Employee_Addresses
     where 'AU'=
       (select Country
          from Work.Supervisors
          where Employee_Addresses.Employee_ID=
                Supervisors.Employee_ID) ;
quit;
```

Step 4: Write Employee_ID and Manager_Name from orion.Employee_Addresses as the third row in the report.

Partial Listing of orion.Employee_Addresses

Employee_ID	Employee_Name
120145	Aisbitt, Sandy
120798	Ardskin, Elizabeth
120656	Amos, Salley
120104	Billington, Kareen
121035	Blackley, James
121141	Bleu, Henri Le
120679	Cutucache, Chrisy
120103	Dawes, Wilson
120672	Guscott, Verne

Work.Supervisors

Employee_ID	Country
120798	US
120800	US
120104	AU
120735	US
121141	US
…	…
120262	US
120679	US
120103	AU
120668	US
121143	US
120260	US
120672	AU

...

Correlated Subqueries

Build third (and final) row of report:

```
Employee_ID   Manager_Name
______________________________________

     120104   Kareen Billington
     120103   Wilson Dawes
     120672   Verne Guscott
```

Business Scenario

Create a report showing **`Employee_ID`** and **`Job_Title`** columns of all sales personnel who did not make any sales.

The table **`orion.Sales`** contains **`Employee_ID`** and **`Job_Title`** columns for all sales personnel.

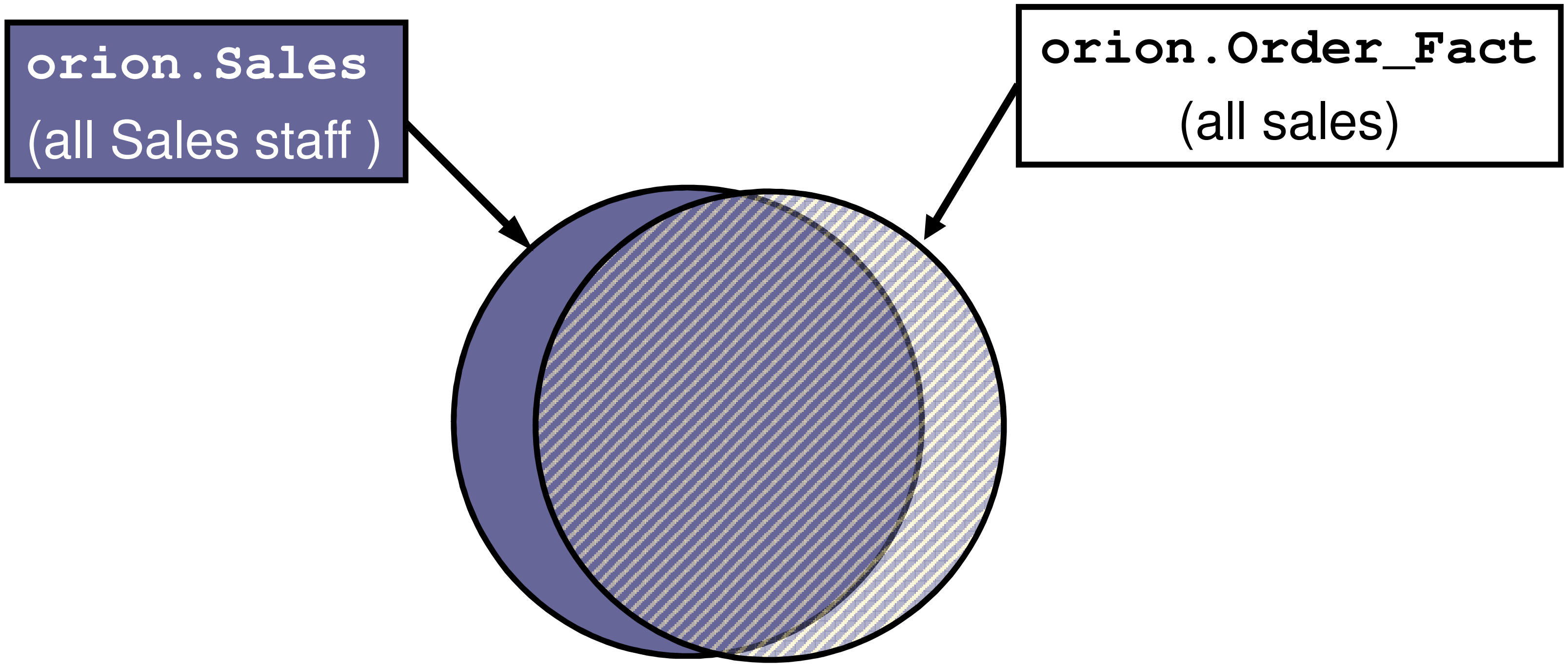
The table **`orion.Order_Fact`** holds information about all sales, and the **`Employee_ID`** column contains the employee identifier of the staff member who made the sale.

The EXISTS and NOT EXISTS Condition

The EXISTS condition tests for the existence of a set of values returned by the subquery.

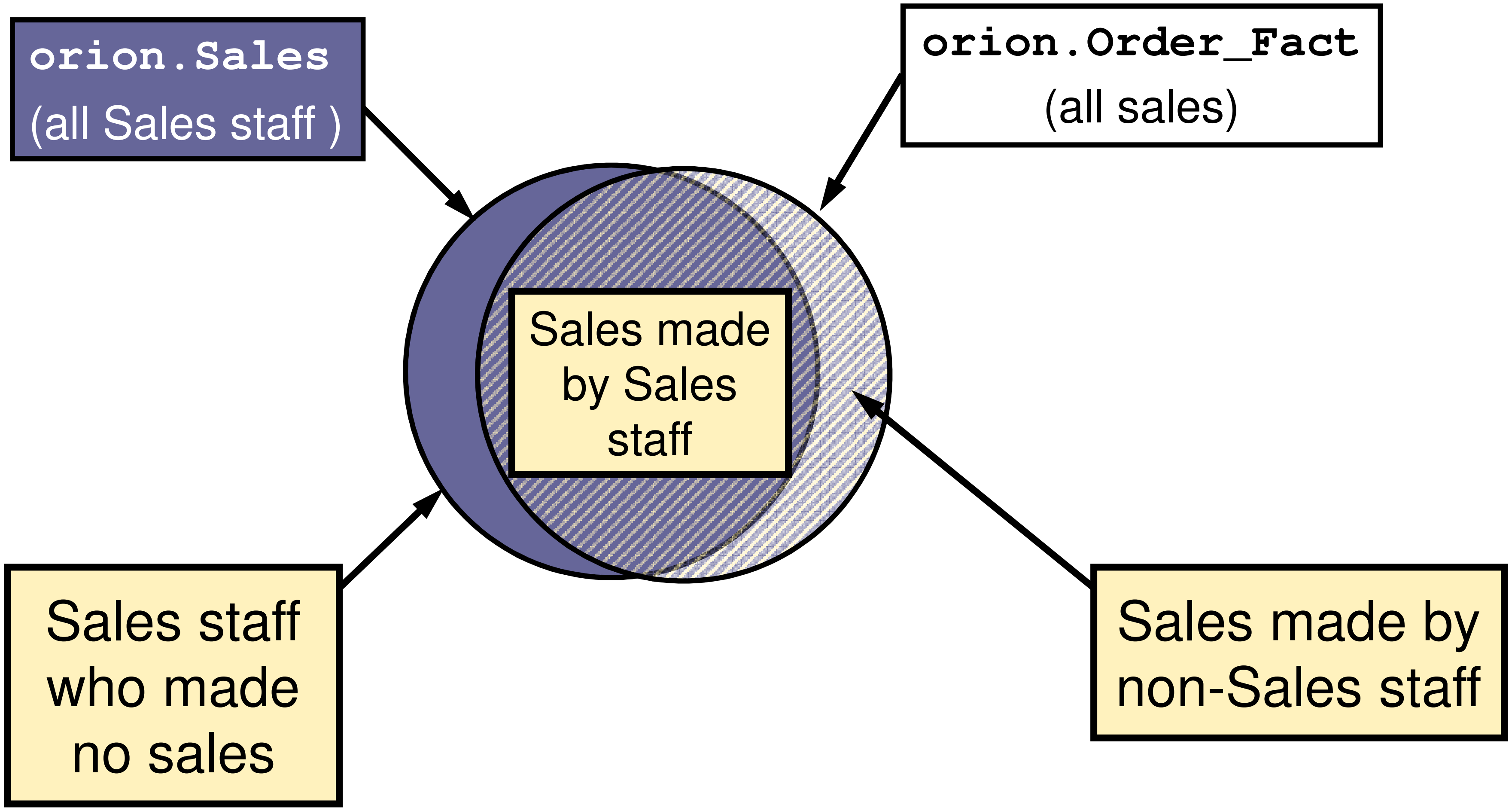
- The EXISTS condition is true if the subquery returns at least one row.
- The NOT EXISTS condition is true if the subquery returns no data.

Correlated Subqueries



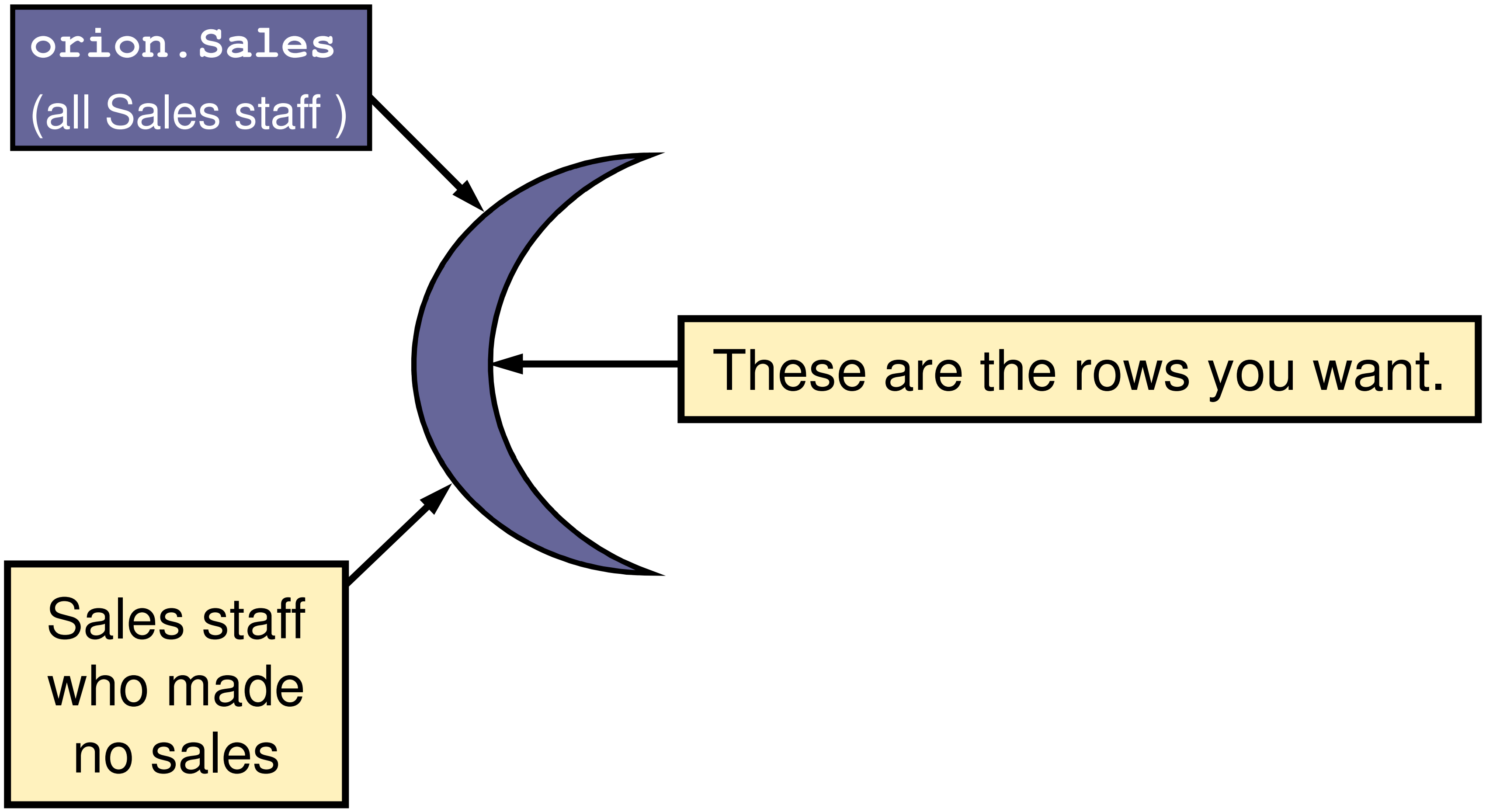
...

Correlated Subqueries


orion.Sales
(all Sales staff)
orion.Order_Fact
(all sales)
Sales made by Sales staff
Sales staff who made no sales
Sales made by non-Sales staff

...

Correlated Subqueries

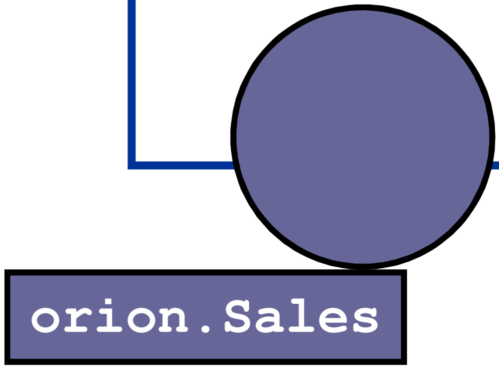


Correlated Subqueries

The table **orion.Sales** contains the employee IDs, job titles, and other demographic information about the Orion Star Sales staff.

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
       where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
```

The population of Sales staff



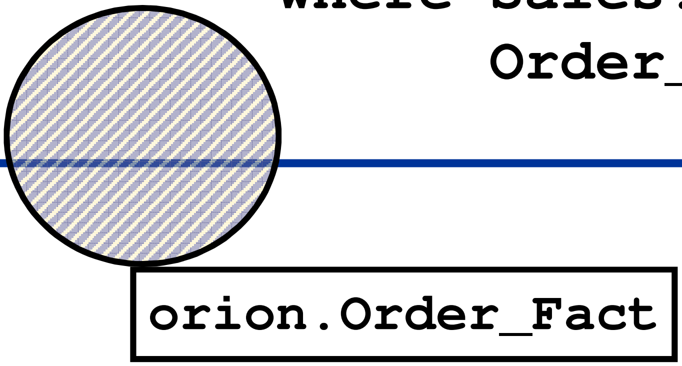
s104d06

Correlated Subqueries

The **orion.Order_Fact** table contains a row for each product sold to a customer.

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
       where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
```

Staff who placed orders



s104d06

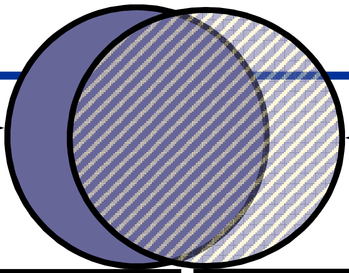
...

Correlated Subqueries

Find Sales employees who exist here...

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
```

...but do **not** exist here.



`orion.Sales` `orion.Order_Fact`

s104d06

Testing Concepts: Referencing Columns

Are the highlighted column references equivalent? Will they result in the same output?

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
       (select *
            from orion.Order_Fact
            where Sales.Employee_ID =
                 Order_Fact.Employee_ID);
quit;
```

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
       (select *
            from orion.Order_Fact
            where Employee_ID=Employee_ID);
quit;
```

Poll

Quiz

Setup for the Poll

1. Submit the program **s104a02** and review the results.
2. Change the original code to the code shown below.
3. Submit the changed program and review the results.

Your instructor will review the log results with you.

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
       (select *
           from orion.Order_Fact
           where Employee_ID=Employee_ID);
quit;
```

4.04 Poll

Is it necessary to qualify the column names in the inner WHERE clause as follows?

```
where sales.Employee_ID=Order_Fact.Employee_ID
```

- ❍ Yes
- ❍ No

4.04 Poll – Correct Answer

Is it necessary to qualify the column names in the inner WHERE clause as follows?

```
where sales.Employee_ID=Order_Fact.Employee_ID
```

[✓ Yes]

No

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
        (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

where 120121=120121

MATCH

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

FALSE

where 120121=120121

MATCH

The NOT EXISTS clause is **FALSE**.
No output rows are written.

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
        (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

where 120122=120122

MATCH

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

FALSE

```
where 120122=120122
```

MATCH

The NOT EXISTS clause is **FALSE**.
No output rows are written.

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
        (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

NO MATCH

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

TRUE

NO MATCH

The NOT EXISTS clause evaluates as **TRUE**.
The first output row is written.

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

Partial PROC SQL Output

```
  Employee_ID  Job_Title
 ---------------------------------
       120102  Sales Manager
```

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
        (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

```
where 120123=120123
```

MATCH

orion.Sales

Employee _ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee _ID	Order _Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
        (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

FALSE

where 120123=120123

MATCH

The NOT EXISTS clause is **FALSE**.
No output rows are written.

orion.Sales

Employee_ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee_ID	Order_Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

NO MATCH

orion.Sales

Employee_ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee_ID	Order_Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

Partial PROC SQL Output

```
Employee_ID  Job_Title
------------------------------------
     120102  Sales Manager
```

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

TRUE

NO MATCH

The NOT EXISTS clause evaluates as **TRUE**.
The next output row is written.

orion.Sales

Employee_ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee_ID	Order_Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

Partial PROC SQL Output

```
 Employee_ID  Job_Title
 ----------------------------------
      120102  Sales Manager
      120103  Sales Manager
```

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

FALSE

where 120124=120124

MATCH

The NOT EXISTS clause is **FALSE**.
No output rows are written.

orion.Sales

Employee_ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
...	...

orion.Order_Fact

Employee_ID	Order_Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

...

Correlated Subqueries

```
proc sql;
   select Employee_ID, Job_Title
      from orion.Sales
      where not exists
         (select *
            from orion.Order_Fact
            where Sales.Employee_ID=
                  Order_Fact.Employee_ID);
quit;
```

When EOF is reached for the table in the outer query, PROC SQL stops processing the query.

orion.Sales

Employee_ID	Job_Title
...	...
120121	Sales Rep. II
120122	Sales Rep. II
120102	Sales Manager
120123	Sales Rep. I
120103	Sales Manager
120124	Sales Rep. I
EOF	

orion.Order_Fact

Employee_ID	Order_Date	Quantity
...	...	...
120122	28MAY2004	1
120121	24JUN2004	1
120124	08OCT2005	1
120123	18AUG2004	1
...	...	...

Partial PROC SQL Output

```
Employee_ID  Job_Title
------------------------------------
     120102  Sales Manager
     120103  Sales Manager
```

...

Chapter Review

True or False:

1. SQL subqueries can return values to be used in an outer query’s FROM clause.

2. A subquery can return several rows of data, but must only return values from a single column.

3. Correlated subqueries use very few resources and are inexpensive to execute.