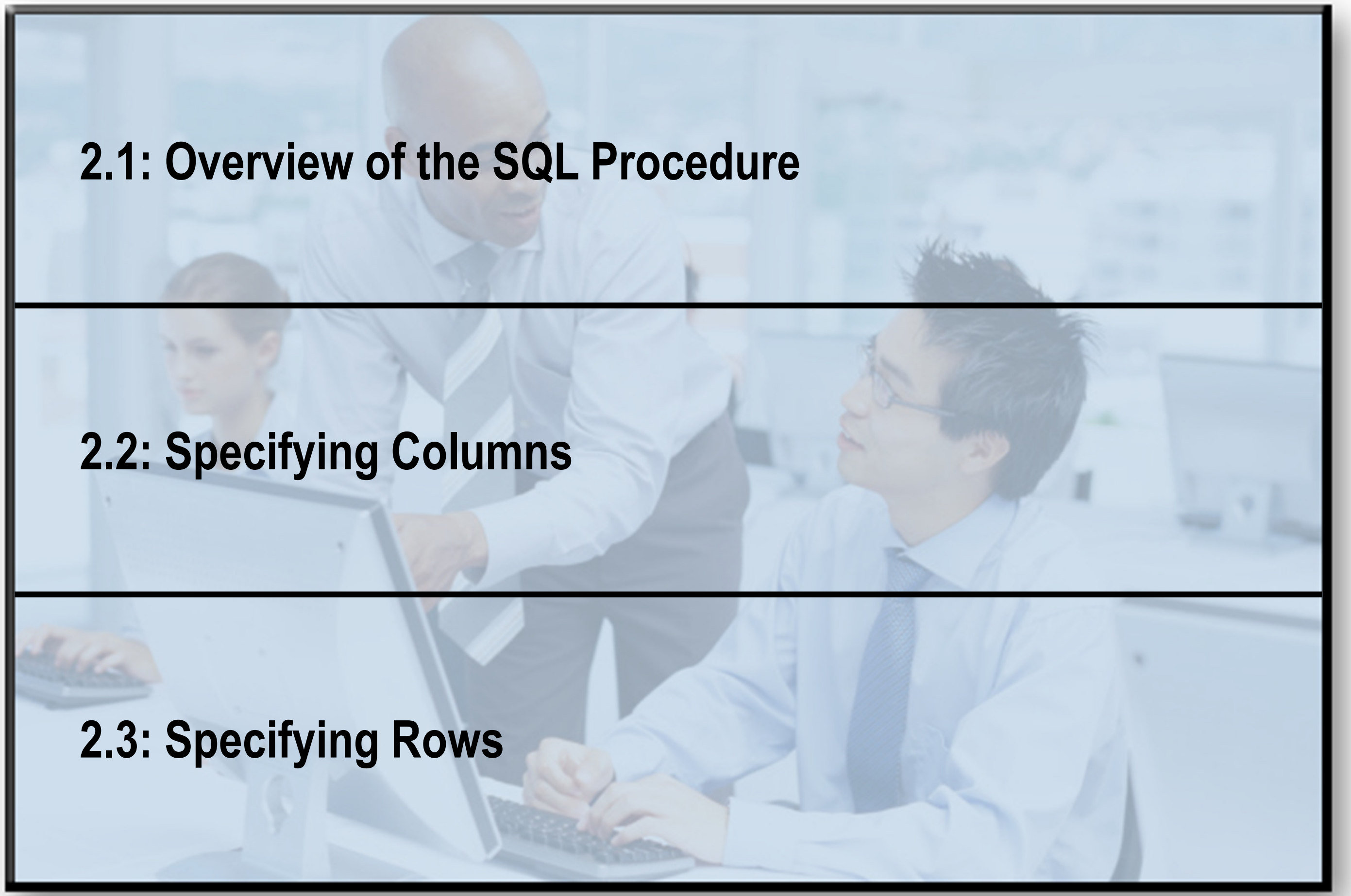
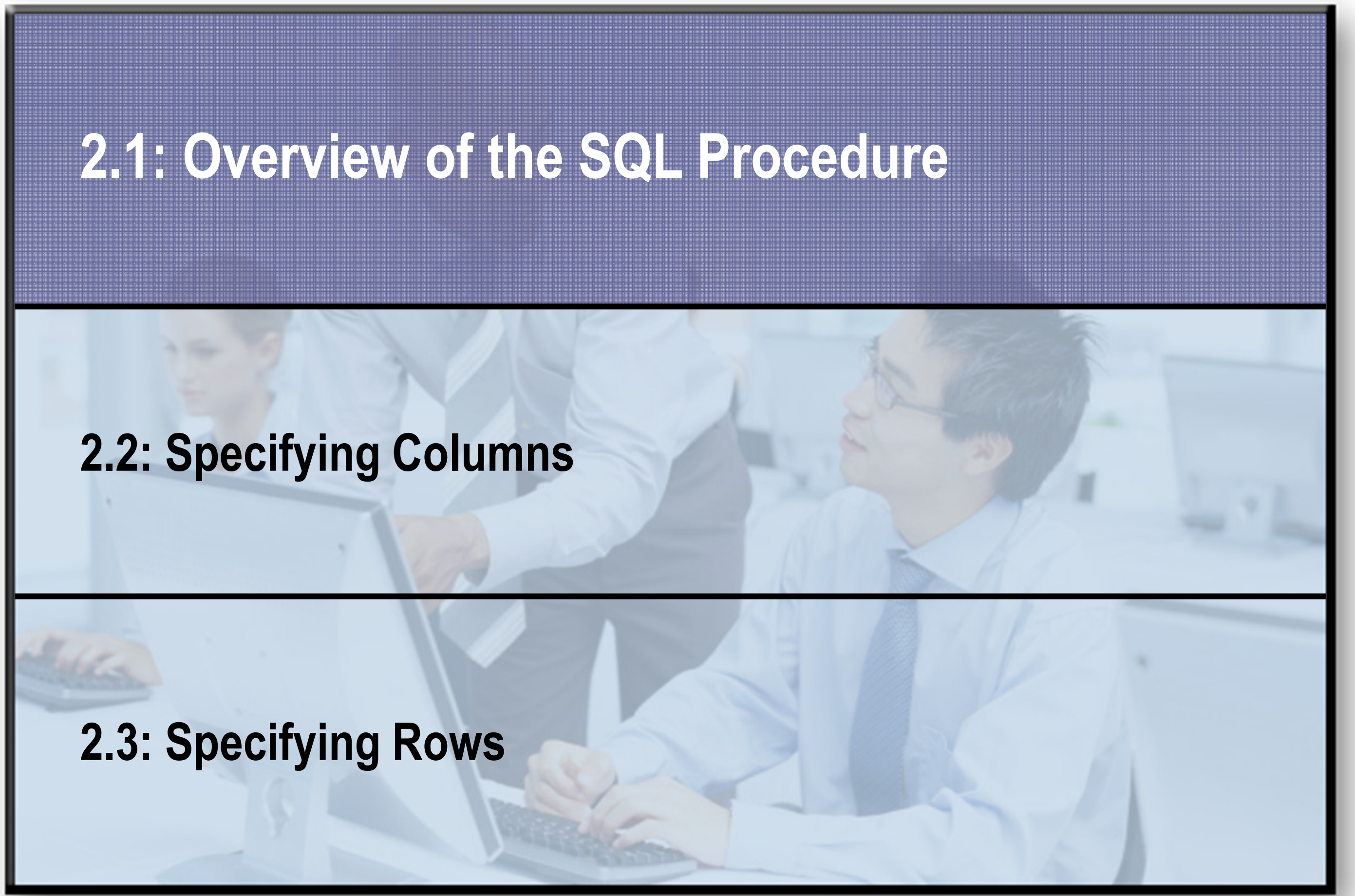
Chapter 2: Basic Queries



Chapter 2: Basic Queries



Objectives

- List key features of the SQL procedure.
- Identify key syntax of the SQL procedure.

The SQL Procedure

The following are features of PROC SQL:

- The PROC SQL statement does not need to be repeated with each query.
- Each statement is processed individually.
- No PROC PRINT step is needed to view query results.
- No PROC SORT step is needed to order query results.
- No RUN statement is needed.
- PROC SQL is terminated with a QUIT statement.

The SELECT Statement

A SELECT statement is used to query one or more SAS data sets.

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

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

s102d01

The SELECT Statement

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

clauses

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

s102d01

SELECT Statement Syntax

General form of the SELECT statement with selected clauses:

```
SELECT column-1<, ...column-n>
      FROM table-1|view-1<, ...table-n|view-n>
      <WHERE expression>
      <GROUP BY column-1<, …column-n>>
      <HAVING expression>
      <ORDER BY column-1<DESC><, …column-n>>;
```

Features of the SELECT Statement

The SELECT statement has the following features:

- selects data that meets certain conditions
- groups data
- specifies an order for the data
- formats the data
- queries 1 to 256 tables

Poll

Quiz

Setup for the Poll

- Open and submit the program **s102a01**.
- The program consists of three steps. Consider the output from the first two steps.
  1) Which step generated errors?
  2) What error message was generated?
- Run the third step and review the SAS log.

2.01 Multiple Choice Poll

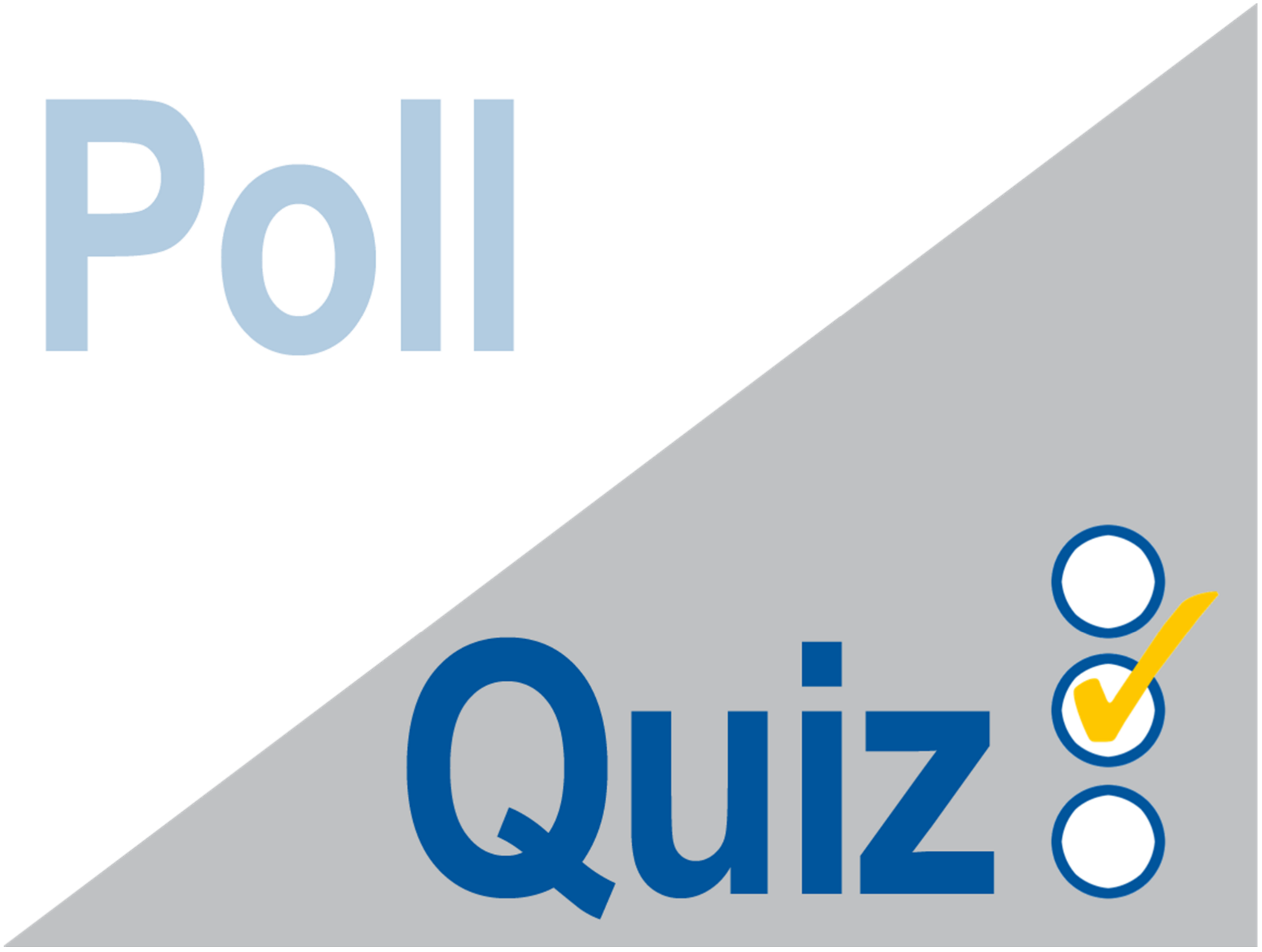
Which step generated errors?

a. Step 1
b. Step 2
c. Step 3

2.01 Multiple Choice Poll – Correct Answer

Which step generated errors?

a. Step 1
[✓ b.] Step 2
c. Step 3


Poll
Quiz

2.02 Quiz

What error message was generated in Step 2?

```
 /* Step 2 */
proc sql;
   from orion.Employee_Payroll
   select Employee_ID, Employee_Gender,
          Salary
   where Employee_Gender = 'M'
   order by EmpID;
quit;
```

s102a01

2.02 Quiz – Correct Answer

What error message was generated in Step 2?

ERROR: Statement is not valid or it is used out of proper order.

```
 /* Step 2 */
proc sql;
   from orion.Employee_Payroll
   select Employee_ID, Employee_Gender,
          Salary
   where Employee_Gender = 'M'
   order by EmpID;
quit;
```

s102a01

The VALIDATE Keyword

Use the VALIDATE keyword to check the SELECT statement syntax.

Partial SAS Log

```
proc sql;
   validate
   select Employee_ID, Employee_Gender,
          Salary
      from orion.Employee_Payroll
      where Employee_Gender = 'F'
      order by Salary desc;
NOTE: PROC SQL statement has valid syntax.
```

s102d02

The VALIDATE Keyword

A common syntax error is to include a comma after the last item in a list.

Partial SAS Log

```
proc sql;
   validate
      select Employee_ID, Employee_Gender,
             Salary,
         from orion.Employee_Payroll
         where Employee_Gender = 'F'
         order by Salary desc;
ERROR: Syntax error, expecting one of the following: !, !!, &,
       (, *, . . .
```

s102d03

Features of the VALIDATE Keyword

The VALIDATE keyword has the following features:

- tests the syntax of a query without executing the query
- checks column name validity
- prints error messages for invalid queries
- is used only for SELECT statements

The NOEXEC Option

Use the NOEXEC procedure option to check the syntax of the entire procedure without executing the statements.

Partial SAS Log

```
proc sql noexec;
   select Employee_ID, Employee_Gender,
          Salary
      from orion.Employee_Payroll
      where Employee_Gender = ' F'
      order by Salary desc;
NOTE: Statement not executed due to NOEXEC option.
```

s102d04

Resetting Options

You can use the RESET statement to add or change PROC SQL options without re-invoking the procedure.

General form of the RESET statement:

RESET *option(s)***;**

For example:

```
reset exec;
   select Employee_ID, Employee_Gender,
          Salary
      from orion.Employee_Payroll
      where Employee_Gender = 'F'
      order by Salary desc;
quit;
```

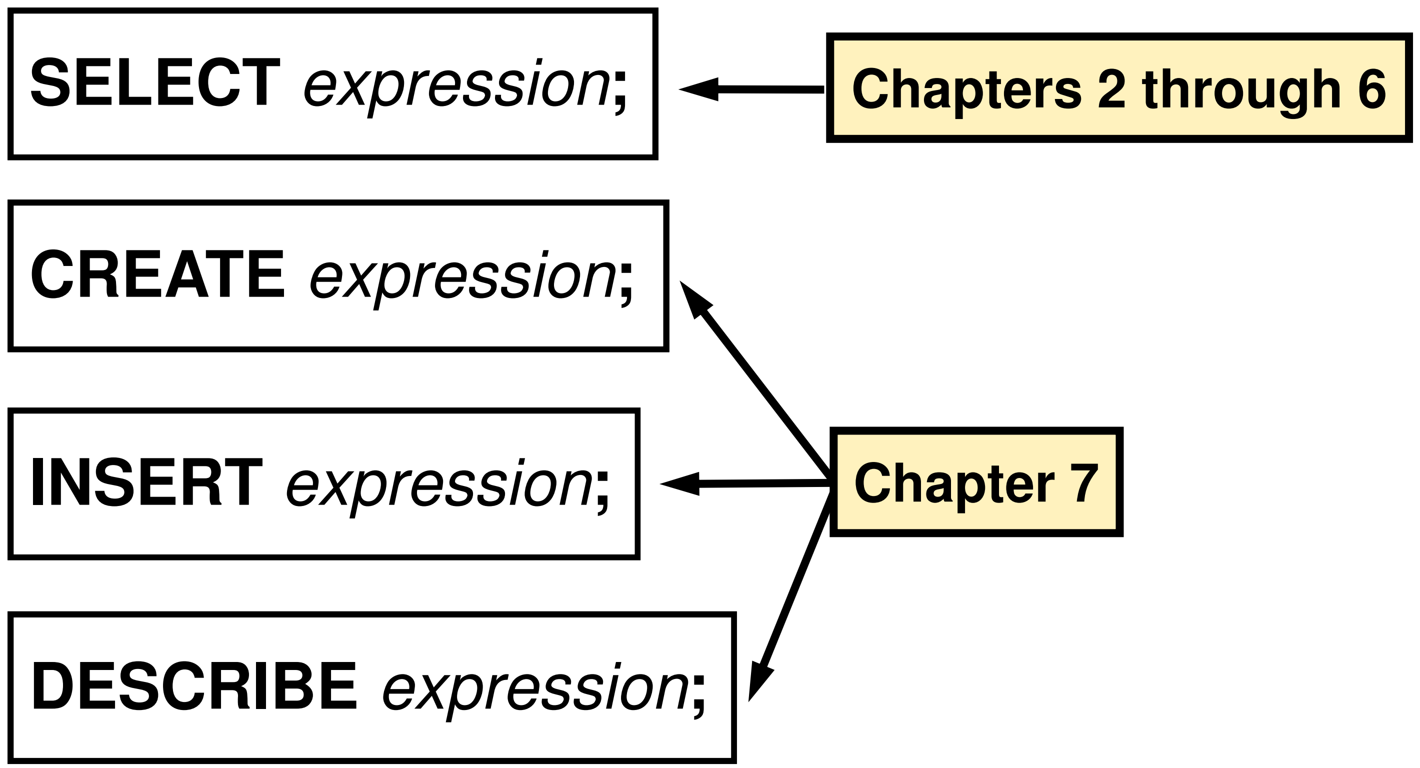
After the EXEC option is reset, the query can be executed.

s102d04

Additional PROC SQL Statements

PROC SQL supports many statements in addition to the SELECT statement.

PROC SQL <*option* <*option*>...>;



SELECT *expression*;

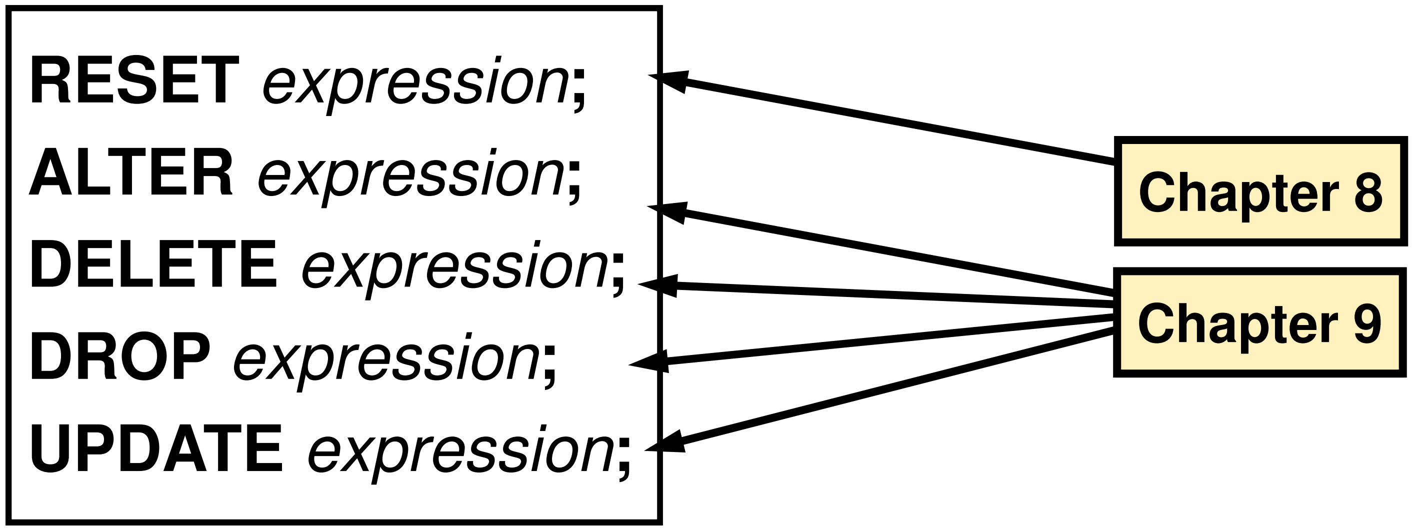
CREATE *expression*;

INSERT *expression*;

DESCRIBE *expression*;

continued...

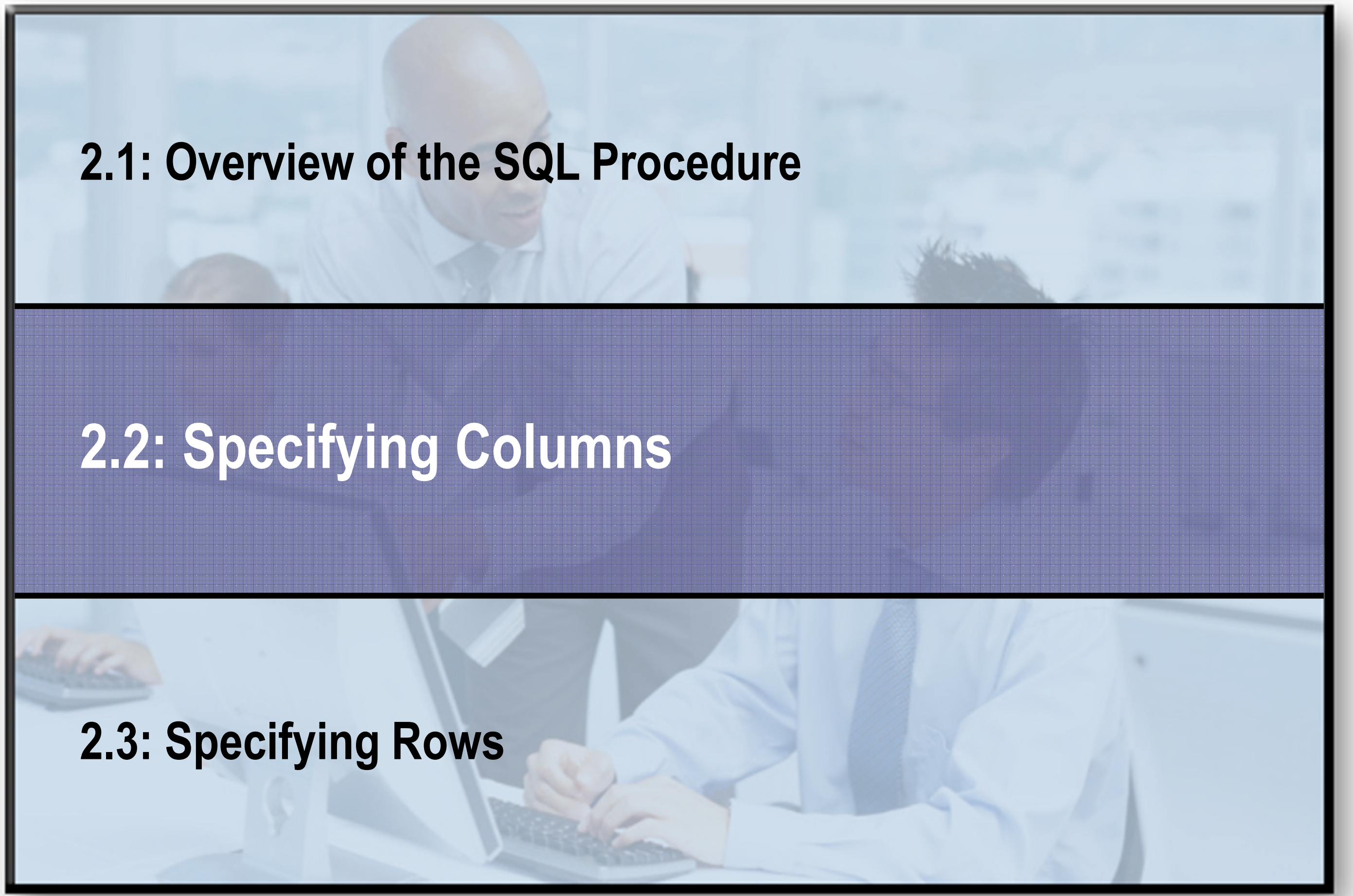
Additional PROC SQL Statements



RESET *expression*;
ALTER *expression*;
DELETE *expression*;
DROP *expression*;
UPDATE *expression*;

Question & Answer

Chapter 2: Basic Queries



Objectives

- Display columns directly from a table.
- Display columns calculated from other columns in a query.
- Calculate columns conditionally using the CASE expression.

Business Scenario

Produce a report that contains the employee identifier, gender, and salary for all Orion Star employees. The data is contained in the `orion.Employee_Payroll` table, a table with which you are not familiar.

Retrieving Data from a Table

You can use an asterisk in a SELECT statement to print all of a table's columns in the order that they were stored.

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

Partial PROC SQL Output

```
                                 The SAS System

             Employee_             Birth_    Employee_    Employee_  Marital_
Employee_ID  Gender      Salary      Date    Hire_Date    Term_Date  Status    Dependents
-------------------------------------------------------------------------------------------
     120101    M         163040      6074        15887            .  S                  0
     120102    M         108255      3510        10744            .  O                  2
     120103    M          87975     -3996         5114            .  M                  1
     120104    F          46230     -2061         7671            .  M                  1
     120105    F          27110      5468        14365            .  S                  0
     120106    M          26960     -5487         5114            .  M                  2
```

s102d05

The FEEDBACK Option

When using an asterisk for the select list, you can specify the FEEDBACK option to write the expanded SELECT statement to the SAS log.

General form of the PROC SQL FEEDBACK option:

```
PROC SQL FEEDBACK;
      SELECT *
            FROM table-1|view-1<, ...table-n|view-n>
            <WHERE expression>
            <GROUP BY column-1<, …column-n>>
            <HAVING expression>
            <ORDER BY column-1<DESC><, …column-n>>;
QUIT;
```

Poll
Quiz

Setup for the Poll

Submit the program **s102a02** and review the SAS log to answer the following question:

- How are the column names represented in the expanded log?

2.03 Multiple Choice Poll

How are the column names represented in the expanded log?

a. The column names are preceded by the table name (**`EMPLOYEE_PAYROLL`**).

b. The column names are preceded by the library reference (**`ORION`**).

c. The column names are preceded by **`Work`**.

2.03 Multiple Choice Poll – Correct Answer

How are the column names represented in the expanded log?

[✓ a.] The column names are preceded by the table name (**`EMPLOYEE_PAYROLL`**).

b. The column names are preceded by the library reference (**`ORION`**).

c. The column names are preceded by **`Work`**.

The FEEDBACK Option

The column names are preceded by the table name.

Partial SAS Log

```
proc sql feedback;
   select *
      from orion.Employee_Payroll;
NOTE: Statement transforms to
      select EMPLOYEE_PAYROLL.Employee_ID,
EMPLOYEE_PAYROLL.Employee_Gender,EMPLOYEE_PAYROLL.Salary,
EMPLOYEE_PAYROLL.Birth_Date,EMPLOYEE_PAYROLL.Employee_Hire_Date,
EMPLOYEE_PAYROLL.Employee_Term_Date,
EMPLOYEE_PAYROLL.Marital_Status,EMPLOYEE_PAYROLL.Dependents
         from ORION.EMPLOYEE_PAYROLL;
quit;
```

s102d06

Retrieving Data from a Table

You can also familiarize yourself with the columns in a table using the DESCRIBE statement.

```
proc sql;
   describe table orion.Employee_Payroll;
quit;
```

Partial SAS Log

```
Employee_ID num format=12.,
Employee_Gender char(1),
Salary num,
Birth_Date num,
Employee_Term_Date num,
Marital_Status char(1)
Dependents num
```

s102d07

Retrieving Data from a Table

After familiarizing yourself with the columns in a table, you can specify the columns to be printed in the order that you want them displayed by using a column list in the SELECT statement.

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

s102d08

Employee IDs, Genders, and Salaries

Partial PROC SQL Output

```
                The SAS System

                  Employee_
  Employee_ID     Gender                Salary
  ____________________________________________

       120101     M                     163040
       120102     M                     108255
       120103     M                      87975
       120104     F                      46230
       120105     F                      27110
       120106     M                      26960
       120107     F                      30475
       120108     F                      27660
       120109     F                      26495
```

Business Scenario

You need to modify your previous report to drop the **`Employee_Gender`** column, and add a new column named **`Bonus`**. The new column should contain an amount equal to 10% of the employee's salary.

Calculated Columns

Calculate the new column’s value using the data in an existing column, and name the new columns using the AS keyword.

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

s102d09

Employee Bonuses

Partial PROC SQL Output

```
                  The SAS System

Employee_ID           Salary          Bonus
_____________________________________________

     120101           163040          16304
     120102           108255        10825.5
     120103            87975         8797.5
     120104            46230           4623
     120105            27110           2711
     120106            26960           2696
     120107            30475         3047.5
     120108            27660           2766
     120109            26495         2649.5
```

Business Scenario

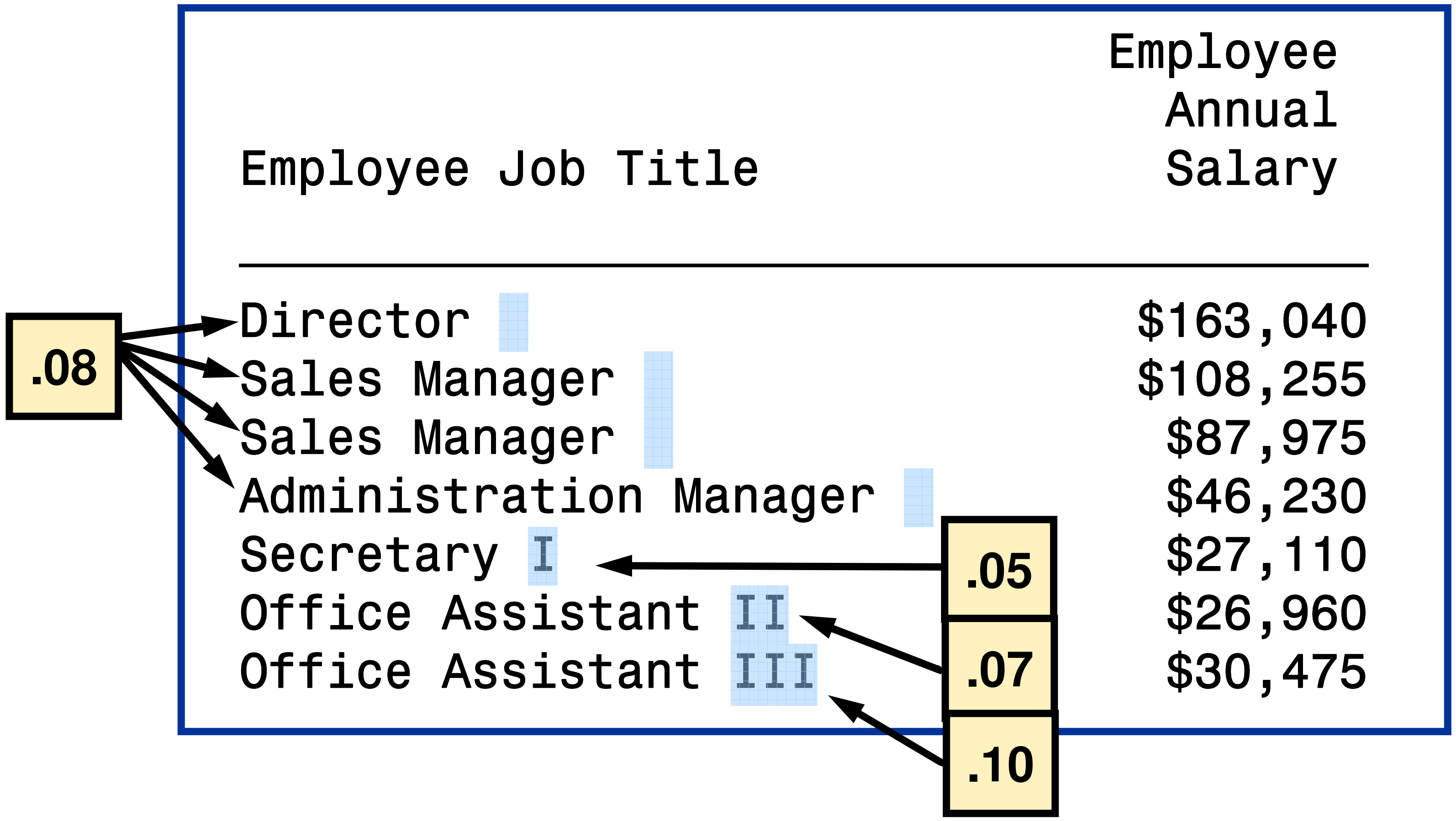
You need to modify the previous bonus report to conditionally calculate bonuses based on the employee's job title.

- Level I employees receive a 5% bonus.
- Level II employees receive a 7% bonus.
- Level III employees receive a 10% bonus.
- Level IV employees receive a 12% bonus.
- All others receive an 8% bonus.

The **`Staff`** table contains all of the information that you need to create this report.

Computing Columns Conditionally

Read data from the **`orion.Staff`** table, and base your bonus calculations on the job title and salary.



```
                                           Employee
                                             Annual
Employee Job Title                           Salary
_______________________________________________________
Director                                   $163,040
Sales Manager                              $108,255
Sales Manager                               $87,975
Administration Manager                      $46,230
Secretary I                                 $27,110
Office Assistant II                         $26,960
Office Assistant III                        $30,475
```

Poll

Quiz

2.04 Multiple Choice Poll

Which of these SAS character functions will be the most useful for identifying the level value for conditional processing?

a. CHAR( )
b. FIND( )
c. SCAN( )
d. SUBSTR( )

2.04 Multiple Choice Poll – Correct Answer

Which of these SAS character functions will be the most useful for identifying the level value for conditional processing?

a. CHAR( )

b. FIND( )

[✓ c.] SCAN( )

d. SUBSTR( )

The SCAN Function

The SCAN function returns the *n*th word or segment from a character string after breaking it up by the delimiters.

General form of the SCAN function:

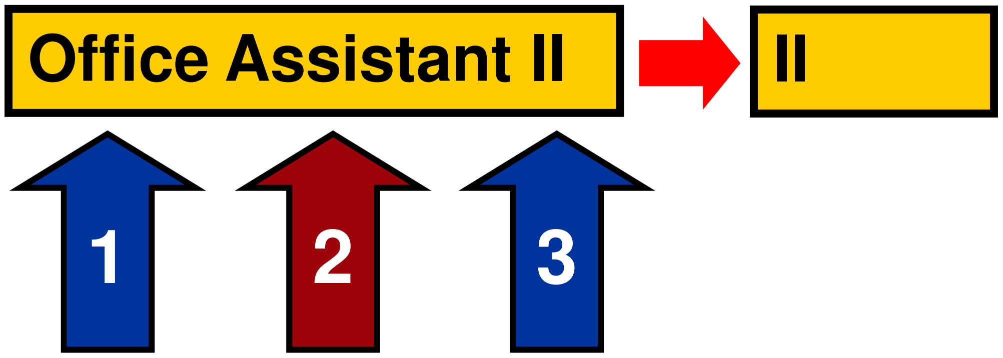
SCAN(*string,n<,charlist><,modifier(s)>*)

string	a character constant, variable, or expression
n	an integer specifying the number of the word or segment that you want SCAN to select
charlist	characters used as delimiters to separate words
modifier	a character that modifies the action of the SCAN function

Extracting the Level from `Job_Title`

Example: Return the third word from **`Job_Title`** and use a blank space as the delimiter.

```
scan(Job_Title,3,' ')
```



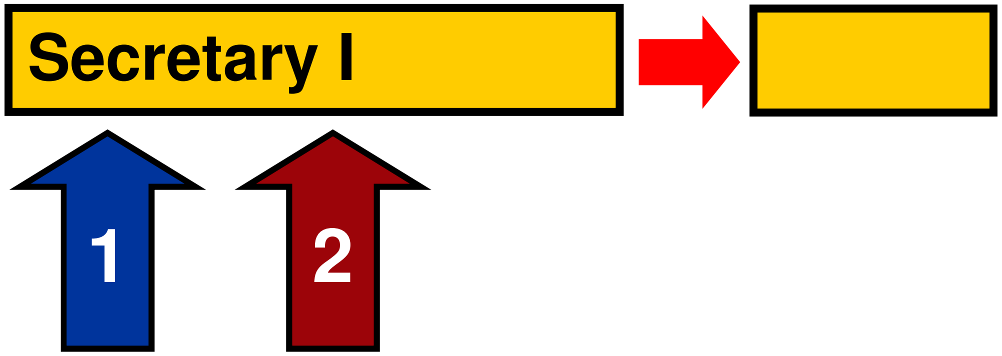
...

Extracting the Level from `Job_Title`

Some **`Job_Title`** values have fewer than three words.

If the value of *n* is greater than the number of words in the character string, the SCAN function returns a missing value.

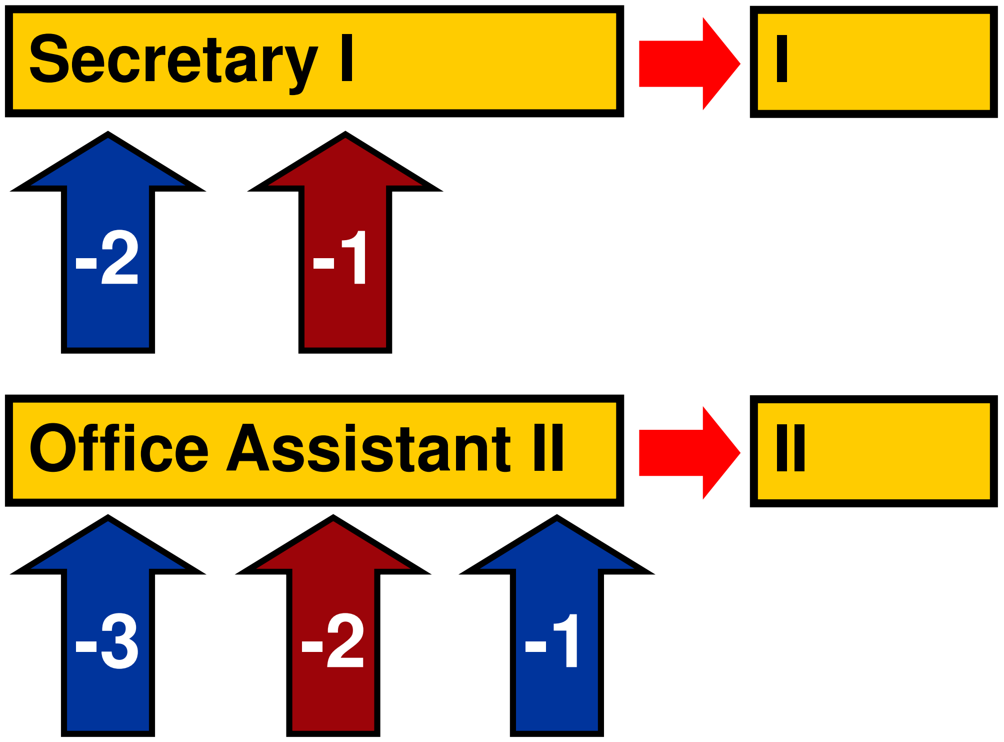
```
scan(Job_Title,3,' ')
```



Extracting the Level from `Job_Title`

If the value of *n* is negative, the SCAN function selects the word in the character string starting from the end of the string.

```
scan(Job_Title,-1,' ')
```



The CASE Expression

You can use a CASE expression in a SELECT statement to create new columns.

General form of the CASE expression in the SELECT statement:

```
SELECT column-1<, ...column-n>
       CASE <case-operand>
              WHEN when-condition THEN result-expression
             <WHEN when-condition THEN result-expression>
             <ELSE result-expression>
       END <as column>
FROM table;
```

Calculating the Bonus

The CASE expression creates employee bonuses based on job titles. Two methods are available.

Method 1:

```
proc sql;
   select Job_Title, Salary,
          case scan(Job_Title,-1,' ')
             when 'I' then Salary*.05
             when 'II' then Salary*.07
             when 'III' then Salary*.10
             when 'IV' then Salary*.12
             else Salary*.08
          end as Bonus
      from orion.Staff;
quit;
```

s102d09a

Calculating the Bonus

Method 2:

```
proc sql;
   select Job_Title, Salary,
          case
             when scan(Job_Title,-1,' ')='I'
                  then Salary*.05
             when scan(Job_Title,-1,' ')='II'
                  then Salary*.07
             when scan(Job_Title,-1,' ')='III'
                 then Salary*.10
             when scan(Job_Title,-1,' ')='IV'
                 then Salary*.12
             else Salary*.08
          end as Bonus
       from orion.Staff;
quit;
```

s102d09a

Calculating the Bonus

Partial PROC SQL Output

```
                         The SAS System

                                    Employee
                                      Annual
Employee Job Title                    Salary       Bonus
--------------------------------------------------------

Director                            $163,040     13043.2
Sales Manager                       $108,255      8660.4
Sales Manager                        $87,975        7038
Administration Manager               $46,230      3698.4
Secretary I                          $27,110      1355.5
Office Assistant II                  $26,960      1887.2
Office Assistant III                 $30,475      3047.5
Warehouse Assistant II               $27,660      1936.2
Warehouse Assistant I                $26,495     1324.75
```

Business Scenario

Management needs a report that includes the employee identifier, gender, and age for an upcoming audit.

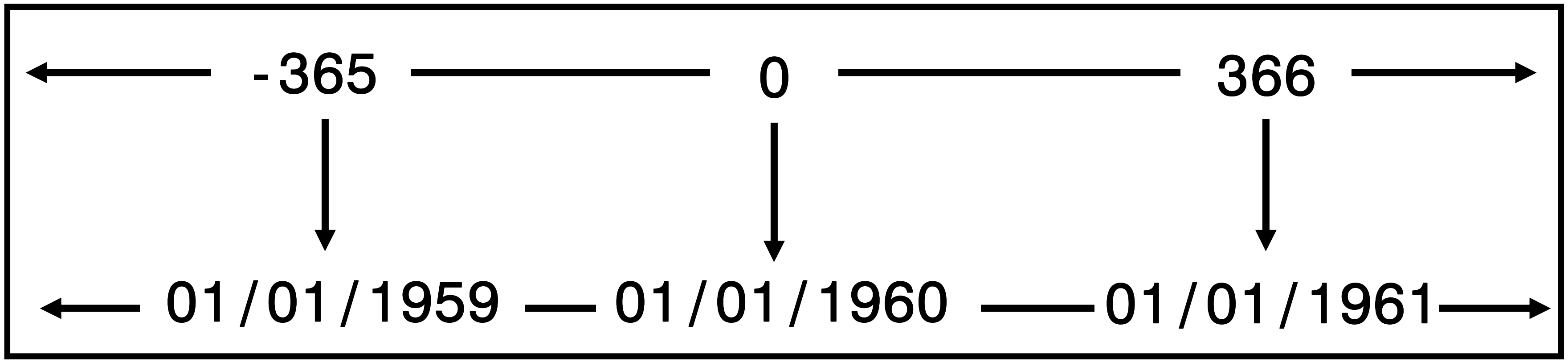
Here is a sketch of the desired report:

Employee_ID	Employee_Gender	Age
120101	M	31
120102	M	38

SAS Date Values (Review)

A SAS date is stored as the number of whole days between January 1, 1960, and the date specified.

Stored Values



Display Values (formatted MMDDYY10.)

Selected SAS Numeric Functions

The following SAS numeric functions are frequently used when you work with SAS dates.

Function	Used To Return	Example
TODAY()	today's date in SAS date form	`today() as date`
MONTH(*arg*)	the month portion of a SAS date variable as an integer between 1-12	`month(Birth_Date) as Birth_Month`
INT(*arg*)	the integer portion of a numeric value	`int(fullage) as age`

Calculated Columns Using SAS Dates

Calculating the age of each employee.

```
proc sql;
   select Employee_ID, Employee_Gender,
          int((today()-Birth_Date)/365.25)
          as Age
      from orion.Employee_Payroll;
quit;
```

s102d10

Using SAS Dates in Calculations

Calculate **Age** based on today's date being 14NOV2007 and a **Birth_Date** value of 18AUG1976.

17484

6074

```
proc sql;
   select Employee_ID, Employee_Gender,
          int((today()-Birth_Date)/365.25)
          as Age
      from orion.Employee_Payroll;
quit;
```

s102d10

...

Using SAS Dates in Calculations

Calculate **Age** based on today's date being 14NOV2007 and a **Birth_Date** value of 18AUG1976.

```
proc sql;
   select Employee_ID, Employee_Gender,
          int((today()-Birth_Date)/365.25)
          as Age
      from orion.Employee_Payroll;
quit;
```

31.23887748

...

Using SAS Dates in Calculations

Calculate **Age** based on today's date being 14NOV2007 and a **Birth_Date** value of 18AUG1976.

```
proc sql;
   select Employee_ID, Employee_Gender,
          int((today()-Birth_Date)/365.25)
          as Age
      from orion.Employee_Payroll;
quit;
```

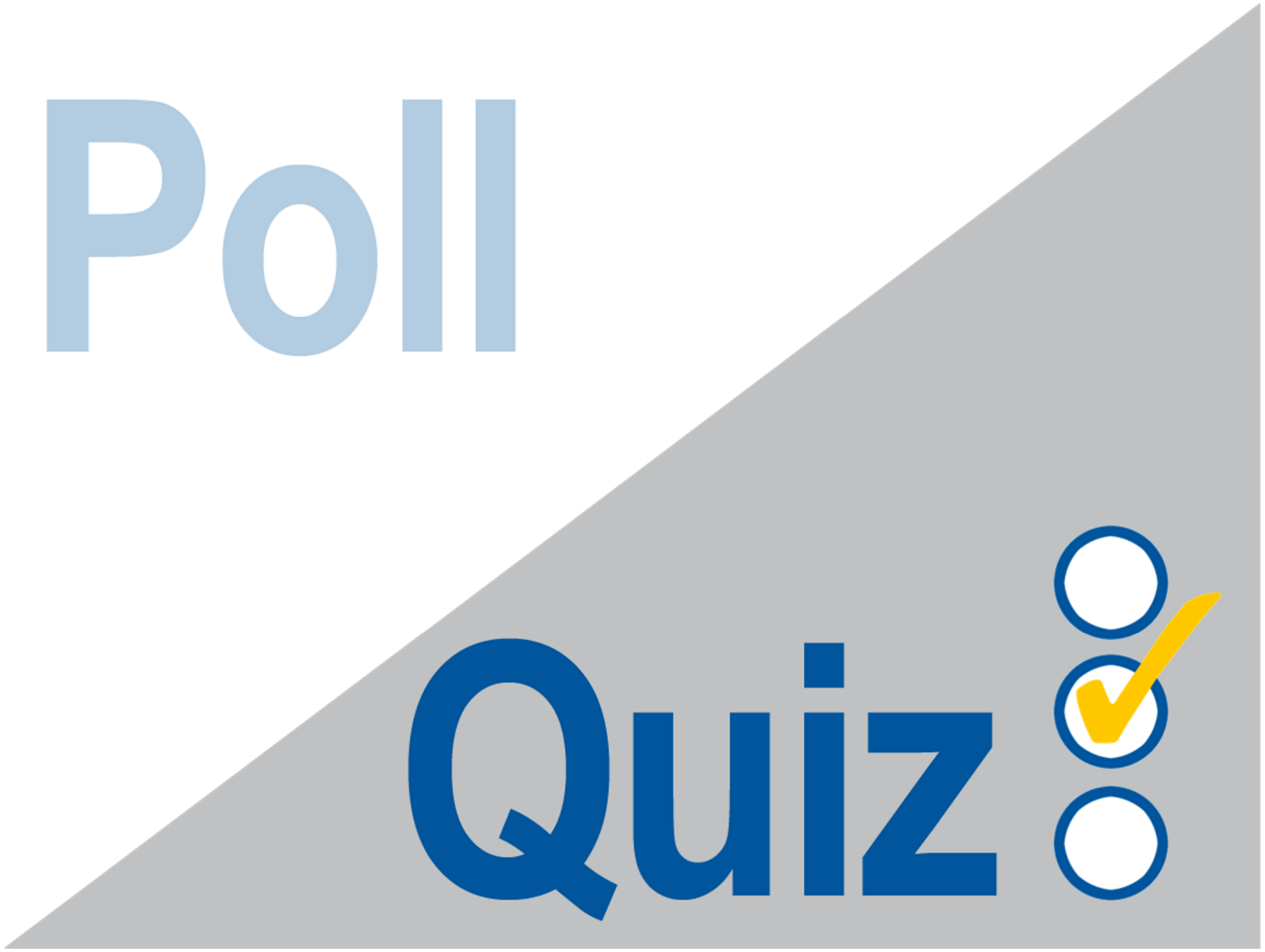
$\boxed{31} \longrightarrow$ `int((today()-Birth_Date)/365.25`

s102d10

Employee Ages

Partial PROC SQL Output

```
                 The SAS System

                  Employee_
Employee_ID   Gender                  Age

_____________________________________________

     120101   M                        31
     120102   M                        38
     120103   M                        58
     120104   F                        53
     120105   F                        32
     120106   M                        62
     120107   F                        58
     120108   F                        23
     120109   F                        20
```


Poll
Quiz

2.05 Quiz

What numeric function would you use to create the **`Birth_Month`** column for the following rows from the **`Employee_Payroll`** table?

```
                          The SAS System

                                                    Employee_
Employee_ID  Birth_Date   Birth_Month   Gender
------------------------------------------------------------------
   120101         6074             8    M
   120102         3510             8    M
   120103        -3996             1    M
   120104        -2061             5    F
   120105         5468            12    F
```

2.05 Quiz – Correct Answer

What numeric function would you use to create the **Birth_Month** column for the following rows from the **Employee_Payroll** table?

The MONTH Function

```
proc sql;
   select Employee_ID, Birth_Date,
          month(Birth_Date) as Birth_Month,
          Employee_gender
        from orion.Employee_Payroll;
quit;
```

s102d10a

Poll

Quiz

2.06 Poll

Would you like to create a table from the results of your queries?

❍ Yes

❍ No

Creating a Table

To create and populate a table with the rows from an SQL query, use the CREATE TABLE statement.

General form of the CREATE TABLE statement:

```
CREATE TABLE table-name AS
      query-expression;
```

Create and Populate a Table with an SQL Query

The SELECT list defines the structure of the **`work.birth_months`** table, and the rows are populated with the data returned by the query.

Define table columns.

```
proc sql;
   create table work.birth_months as
      select Employee_ID, Birth_Date,
             month(Birth_Date) as
             Birth_Month,
             Employee_gender
      from orion.Employee_Payroll;
   describe table work.birth_months;
   select * from work.birth_months;
quit;
```

s102d10b

Create and Populate a Table with an SQL Query

Partial SAS Log

```
NOTE: Table WORK.BIRTH_MONTHS created, with 424
rows and 4 columns.
```

```
create table WORK.BIRTH_MONTHS
  (Employee_ID num format=12.,
   Birth_Date num,
   Birth_Month num,
   Employee_Gender char(1))
```

Create and Populate a Table with an SQL Query

Partial SAS Output (5 out of 424)

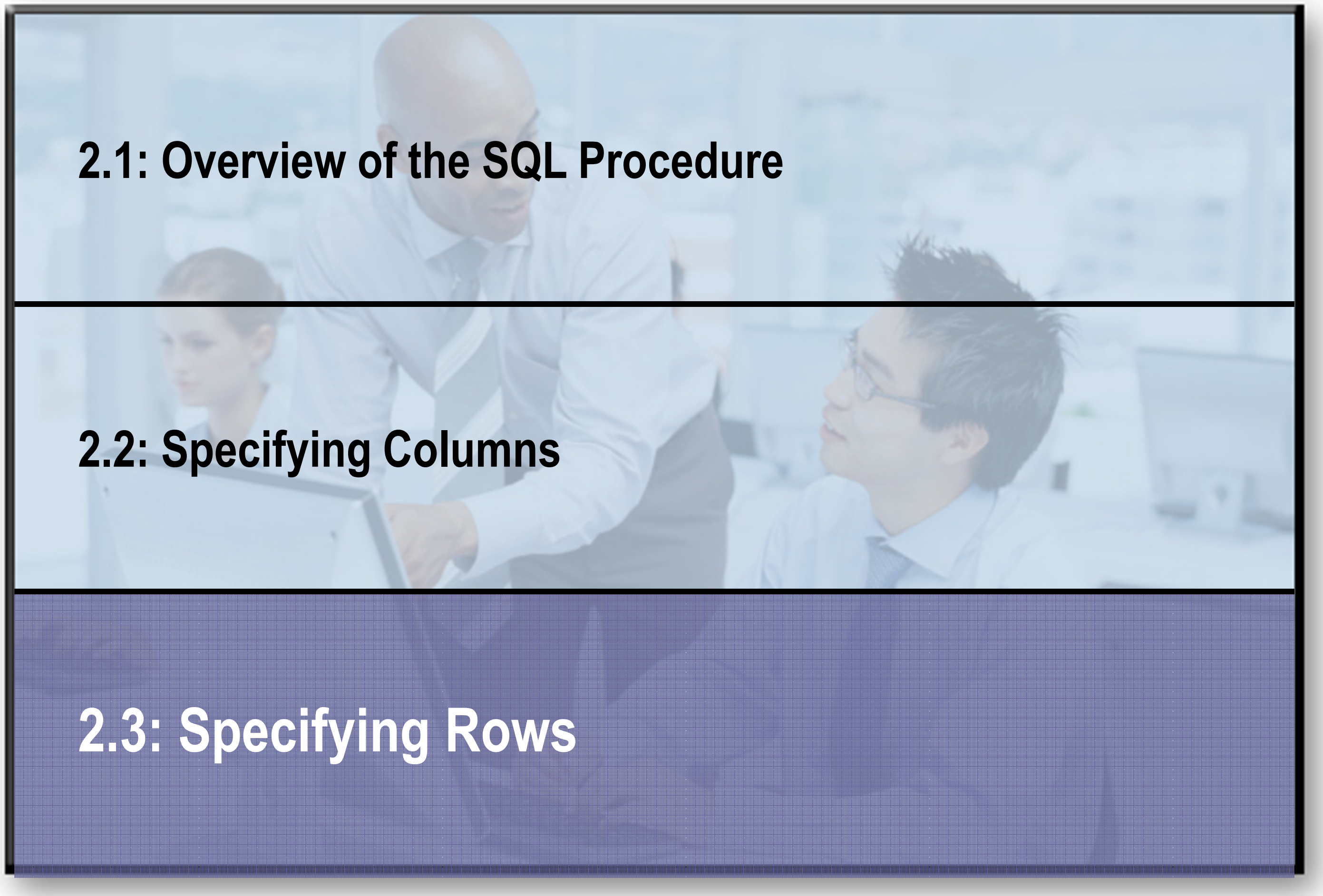
```
                 Employee_
Employee_ID  Birth_Date  Birth_Month  Gender
__________________________________________________

     120101        6074            8  M
     120102        3510            8  M
     120103       -3996            1  M
     120104       -2061            5  F
     120105        5468           12  F
```

Question & Answer

Exercise

This exercise reinforces the concepts discussed previously.

Chapter 2: Basic Queries



Objectives

- Select a subset of rows in a query.
- Eliminate duplicate rows in a query.

Specifying Rows in a Table

Example: Display the names of the Orion Star departments using the **`orion.Employee_Organization`** table.

```
proc sql;
   select Department
      from orion.Employee_Organization;
quit;
```

s102d11

All Rows in a Table

Partial PROC SQL Output

```
          The SAS System
Department
_____________________________
Sales Management
Sales Management
Sales Management
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Administration
Engineering
Engineering
```

Eliminating Duplicate Rows

Use the DISTINCT keyword to eliminate duplicate rows in query results.

Example: Determine the distinct departments in the Orion Star organization.

```
proc sql;
   select distinct Department
      from orion.Employee_Organization;
quit;
```

s102d11

Eliminating Duplicate Rows

Partial PROC SQL Output

```
                The SAS System

Department
________________________________________

Accounts
Accounts Management
Administration
Concession Management
Engineering
Executives
Group Financials
Group HR Management
IS
Logistics Management
Marketing
Purchasing
Sales
```

Poll
Quiz

2.07 Multiple Answer Poll

Which SELECT clauses select only the unique combinations of **`Employee_Gender`** and **`Job_Title`**?

a. **`select distinct Employee_Gender, distinct Job_Title...`**

b. **`select unique Employee_Gender, Job_Title...`**

c. **`select distinct Employee_Gender, Job_Title...`**

d. **`select distinct Employee_Gender Job_Title...`**

2.07 Multiple Answer Poll – Correct Answer

Which SELECT clauses select only the unique combinations of **`Employee_Gender`** and **`Job_Title`**?

a. **`select distinct Employee_Gender, distinct Job_Title...`**

[✓ b.] **`select unique Employee_Gender, Job_Title...`**

[✓ c.] **`select distinct Employee_Gender, Job_Title...`**

d. **`select distinct Employee_Gender Job_Title...`**

Business Scenario

Create a list of personnel with salaries above $112,000. Include the employee identifier, job title, and salary.

Subsetting with the WHERE Clause

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

Example: Display all employees who earn more than $112,000.

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

s102d12

Subsetting with the WHERE Clause

Partial 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
```

Subsetting with the WHERE Clause

You can use all common comparison operators in a WHERE clause.

Mnemonic	Symbol	Definition
LT †	<	Less than
GT †	>	Greater than
EQ †	=	Equal to
LE †	<=	Less than or equal to
GE †	>=	Greater than or equal to
NE †	< >	Not equal to
	¬= †	Not equal to (EBCDIC)
	^= †	Not equal to (ASCII)

Subsetting with the WHERE Clause

Use only one WHERE clause in a SELECT statement. To specify multiple subsetting criteria, combine expressions with logical operators.

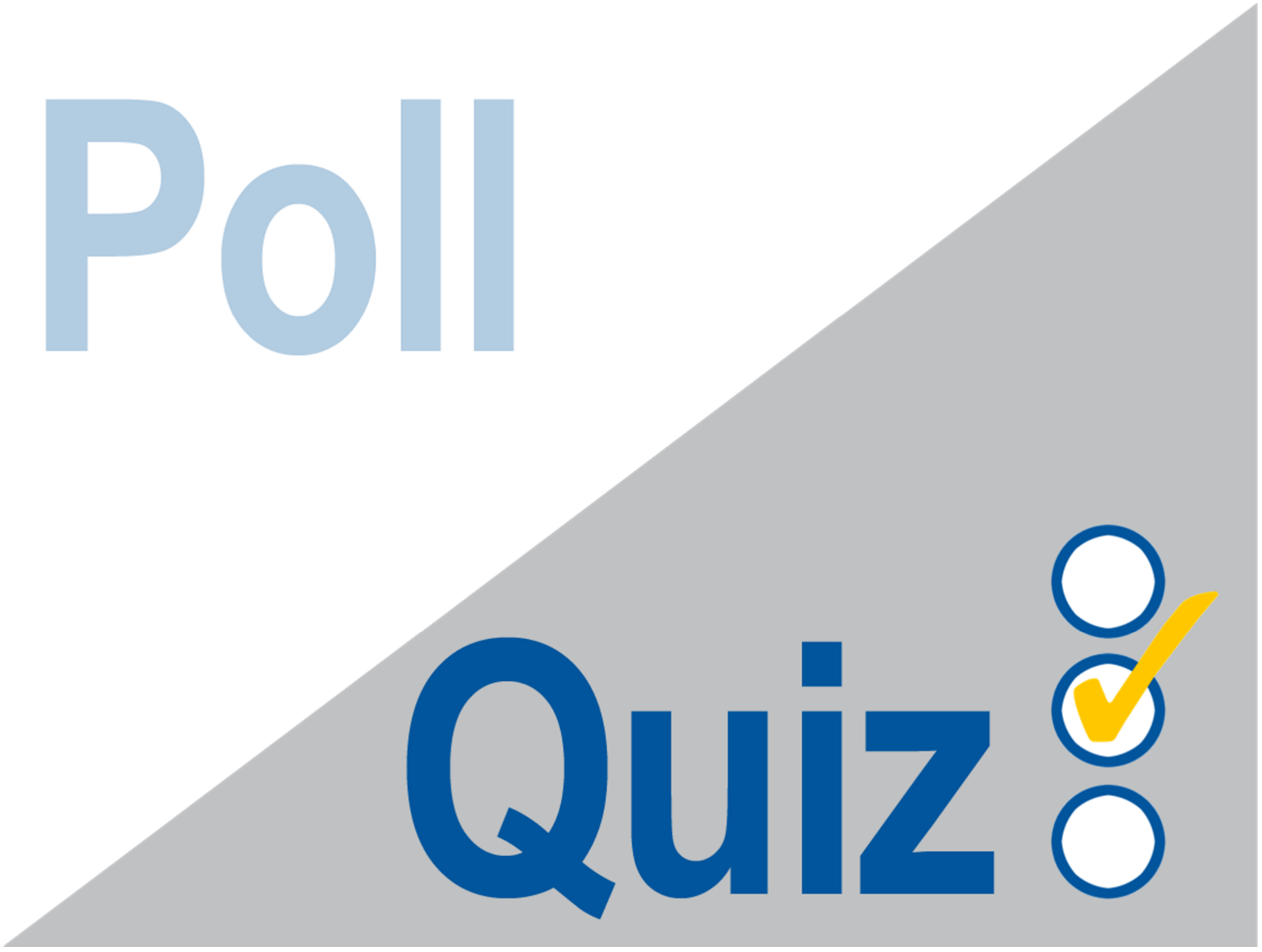
Mnemonic	Symbol	Definition
OR	| †	or, either
AND	& †	and, both
NOT	¬ †	not, negation (EBCDIC)
NOT	^ †	not, negation (ASCII)

Subsetting with the WHERE Clause

Common WHERE clause operators with examples:

Operator	Example
IN	`where JobCategory in ('PT','NA','FA')`
CONTAINS or ? †	`where word ? 'LAM'`
IS NULL or IS MISSING †	`where Product_ID is missing`
BETWEEN – AND	`where Salary between 70000 and 80000`
SOUNDS LIKE (=*) †	`where LastName =* 'SMITH'`

LIKE using % or _	`where Employee_Name like 'H%'`
	`where JobCategory like '__1'`


Poll
Quiz

2.08 Quiz

Modify program **s102a03** to provide a WHERE expression that selects only those rows where the employees’ first names begin with N.

Desired Output

```
                    The SAS System

Employee_Name                              Employee_ID
______________________________________________________

Apr, Nishan                                     120759
James, Narelle                                  120155
Kokoszka, Nikeisha                              120765
Plybon, Nicholas                                120276
Post, Nahliah                                   120748
Smith, Nasim                                    121032
```

2.08 Quiz – Correct Answer

Modify program **s102a03** to provide a WHERE expression that selects only those rows where the employees' first names begin with N.

One possible solution:

```
select Employee_Name, Employee_ID
   from orion.Employee_Addresses
   where Employee_Name contains ', N';
quit;
```

s102a03a

Subsetting with the WHERE Clause

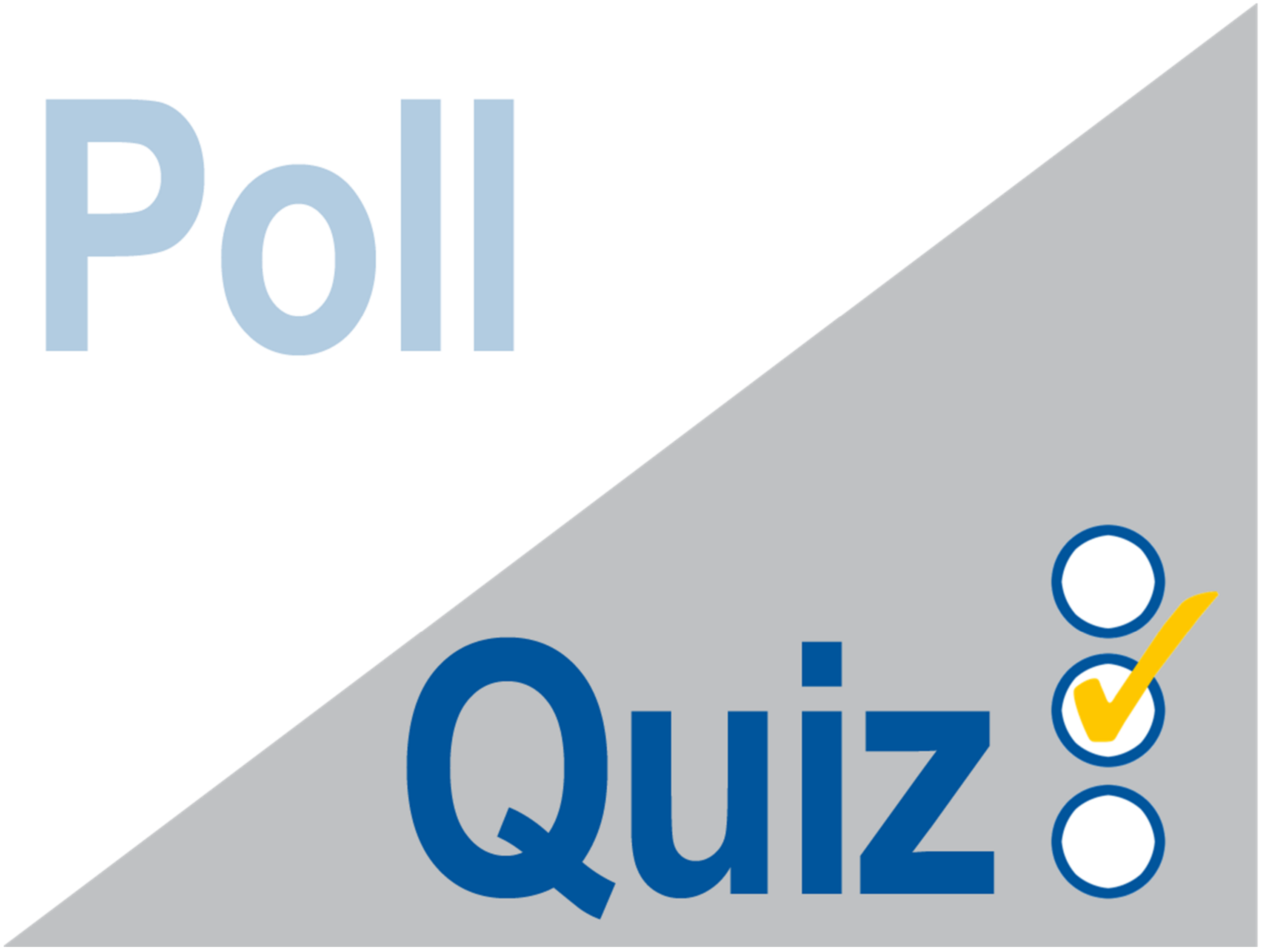
Select all SA job codes that contain an underscore (_).

```
proc sql;
   select Employee_ID, Job_Code
      from work.Employee_Organization2
      where Job_Code like 'SA_%';
quit;
```

Partial PROC SQL Output

```
Employee_ID  Job_Code
_____________________
     120115  SAI
     120116  SA_II
     120669  SAIV
     120671  SAIII
     120673  SA_II
     120678  SAII
```

s102d13a


Poll
Quiz

2.09 Quiz

Why do you see all SA job codes and not only the ones that contain an underscore (_)?

```
Employee_ID  Job_Code
---------------------
     120115  SAI
     120116  SA_II
     120669  SAIV
     120671  SAIII
     120673  SA_II
     120678  SAII
```

```
proc sql;
   select Employee_ID, Job_Code
      from work.Employee_Organization2
      where Job_Code like 'SA_%';
quit;
```

2.09 Quiz – Correct Answer

Why do you see all SA job codes and not only the ones that contain an underscore (_)?

You see all the SAS job codes because the WHERE expression uses the LIKE operator. The underscore represents any one character, not an underscore (_) in the third position.

```
proc sql;
   select Employee_ID, Job_Code
      from work.Employee_Organization2
      where Job_Code like 'SA_%';
quit;
```

ESCAPE Clause

To search for actual percent or underscore characters in your text using the LIKE operator, you must use an ESCAPE clause.

The ESCAPE clause in the LIKE condition enables you to designate a single character string literal, known as an *escape character*, to indicate how PROC SQL should interpret the LIKE wildcards (% and _) when SAS is searching within a character string.

ESCAPE Clause

```
proc sql;
   select Employee_ID, Job_Code
      from work.Employee_Organization2
      where Job_Code like 'SA/_%' ESCAPE '/';
quit;
```

Partial PROC SQL Output

```
Employee_ID  Job_Code
_____________________
     120116  SA_II
     120673  SA_II
     120681  SA_II
     120692  SA_II
     120792  SA_II
     121012  SA_II
```

s102d13b

Poll

Quiz

2.10 Multiple Choice Poll

Which of the following WHERE clauses correctly selects rows with a **`Job_Code`** value that begins with an underscore?

a. **`where Job_Code like '_%'`**

b. **`where Job_Code contains '_%'`**

c. **`where Job_Code like '%_' escape '/_'`**

d. **`where Job_Code like '/_%' escape '/'`**

2.10 Multiple Choice Poll – Correct Answer

Which of the following WHERE clauses correctly selects rows with a **`Job_Code`** value that begins with an underscore?

a. **`where Job_Code like '_%'`**

b. **`where Job_Code contains '_%'`**

c. **`where Job_Code like '%_' escape '/_'`**

[✓ d.] **`where Job_Code like '/_%' escape '/'`**

Business Scenario

Return to the original 10% bonus program.

You want to create a report that includes only those employees who receive bonuses less than $3000.

Subsetting with Calculated Values

First attempt:

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

s102d14

Subsetting with Calculated Values

Because a WHERE clause is evaluated before the SELECT clause, columns used in the WHERE clause must exist in the table or be derived from existing columns.

Because the **`Bonus`** column is not in the source table, an error was generated.

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

Subsetting with Calculated Values

One solution is to repeat the calculation in the WHERE clause.

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

s102d14

Subsetting with Calculated Values

An alternate method is to use the CALCULATED keyword to refer to an already calculated column in the SELECT clause.

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

s102d14

Subsetting with Calculated Values

Partial PROC SQL Output

```
                       The SAS System

              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
     120111   M                    26895      2689.5
     120112   F                    26550        2655
```

Subsetting with Calculated Values

You can also use the CALCULATED keyword in other parts of a query.

```
proc sql;
   select Employee_ID, Employee_Gender,
          Salary, Salary * .10 as Bonus,
          calculated Bonus/2 as Half
      from orion.Employee_Payroll
      where calculated Bonus < 3000;
quit;
```

s102d14

Subsetting with Calculated Values

Partial PROC SQL Output

```
                          The SAS System

                 Employee_
Employee_ID      Gender          Salary        Bonus        Half
----------------------------------------------------------------
     120105      F                27110         2711      1355.5
     120106      M                26960         2696        1348
     120108      F                27660         2766        1383
     120109      F                26495       2649.5     1324.75
     120110      M                28615       2861.5     1430.75
     120111      M                26895       2689.5     1344.75
     120112      F                26550         2655      1327.5
```

Question & Answer

Chapter Review

1. What SQL statement is used to display the values from columns in a table?

2. What expression is used to conditionally calculate column values?

3. Name the clause that selects a subset of rows in a query.

4. If your query returns multiple rows with identical content, what keyword can eliminate duplicate rows?

Chapter Review Answers

1. What SQL statement is used to display the values from columns in a table?

   The SELECT statement

2. What expression is used to conditionally calculate column values?

   The CASE expression

3. Name the clause that selects a subset of rows in a query.

   The WHERE clause

4. If your query returns multiple rows with identical content, what keyword can eliminate duplicate rows?

   The DISTINCT keyword