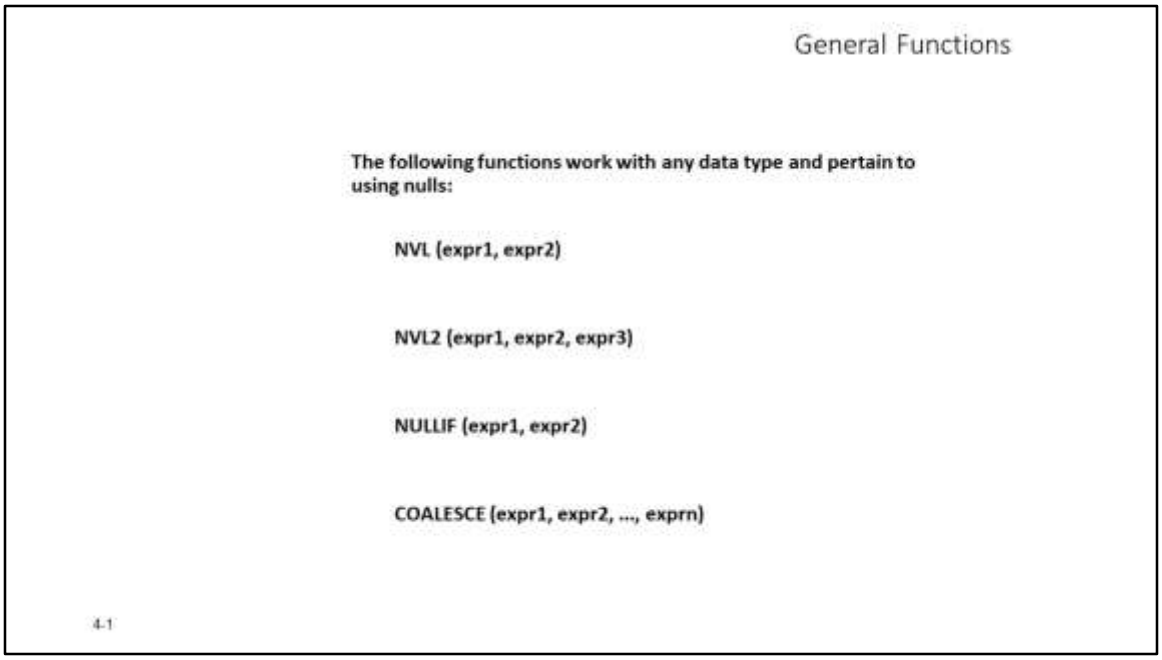
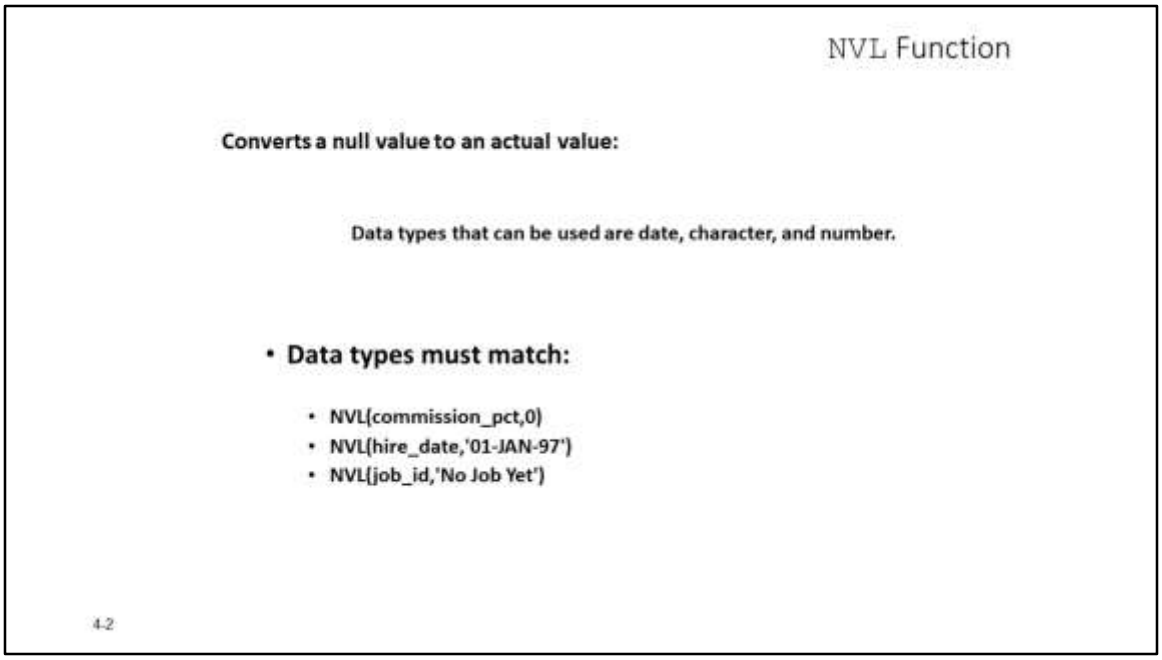


General Functions

These functions work with any data type and pertain to the use of null values in the expression list.

Note: For more information about the hundreds of functions available, see the "Functions" section in *Oracle Database SQL Language Reference* for 10*g* or 11*g* database.

Function	Description
NVL	Converts a null value to an actual value
NVL2	If expr1 is not null, NVL2 returns expr2. If expr1 is null, NVL2 returns expr3. The argument expr1 can have any data type.
NULLIF	Compares two expressions and returns null if they are equal; returns the first expression if they are not equal
COALESCE	Returns the first non-null expression in the expression list



NVL Function

To convert a null value to an actual value, use the NVL function.

Syntax

```
NVL (expr1, expr2)
```

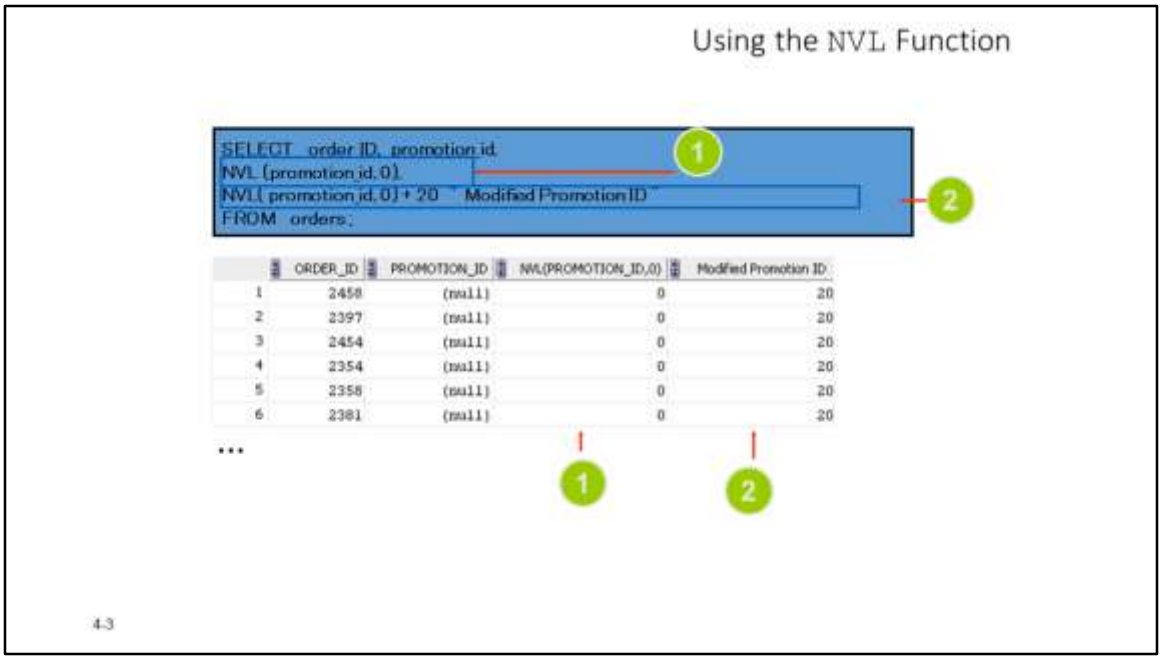
In the syntax:

- *expr1* is the source value or expression that may contain a null
- *expr2* is the target value for converting the null

You can use the NVL function to convert any data type, but the return value is always the same as the data type of *expr1*.

NVL Conversions for Various Data Types

Data Type	Conversion Example
NUMBER	NVL(*number_column*,9)
DATE	NVL(*date_column*, '01-JAN-95')
CHAR or VARCHAR2	NVL(*character_column*, 'Unavailable')



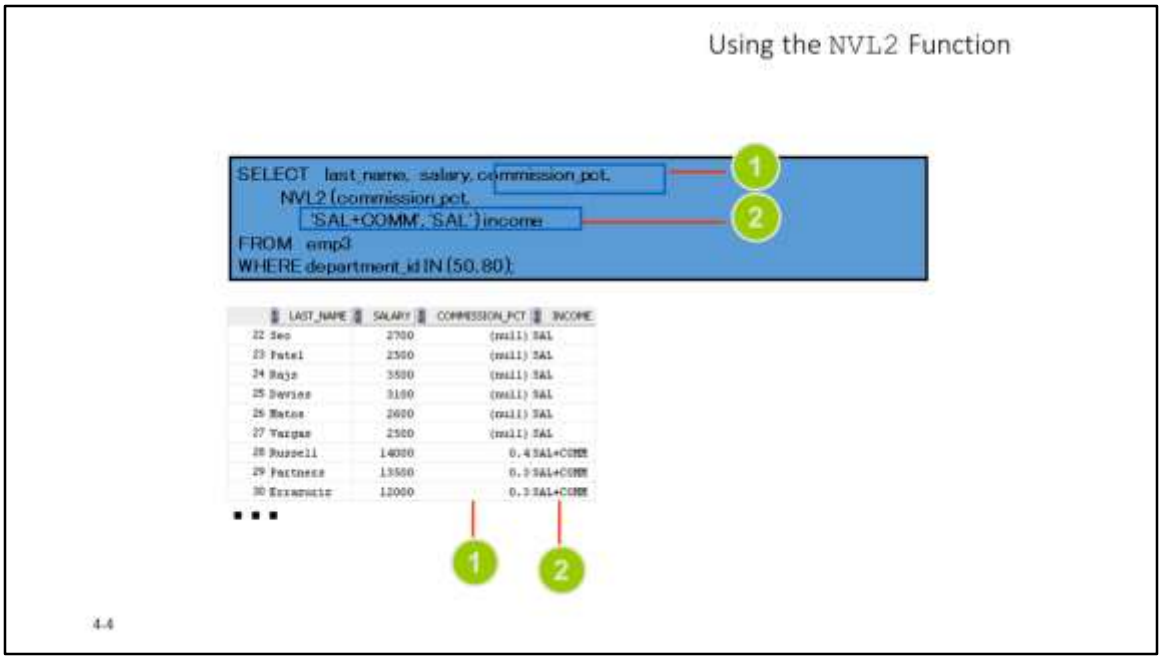
Using the NVL Function

To calculate the modified promotion IDof all the orders, you need to add 20 to the current promotion ID of every order.

```
SELECT order_id, promotion_id,
 promotion_id + 20
FROM  orders;
```

If any column value in an expression is null, the result is null. To calculate values for all the orders, you must convert the null value to a number before applying the arithmetic operator. In the example in the slide, the NVL function is used to convert null values to zero.

	LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
1	Whalen	4400	(null)	(null)
...				
16	Vargas	2500	(null)	(null)
17	Zlotkey	10500	0.2	151200
18	Abel	11000	0.3	171600
19	Taylor	8600	0.2	123840
20	Grant	7000	0.15	96600



Using the NVL2 Function

The NVL2 function examines the first expression. If the first expression is not null, the NVL2 function returns the second expression. If the first expression is null, the third expression is returned.

Syntax

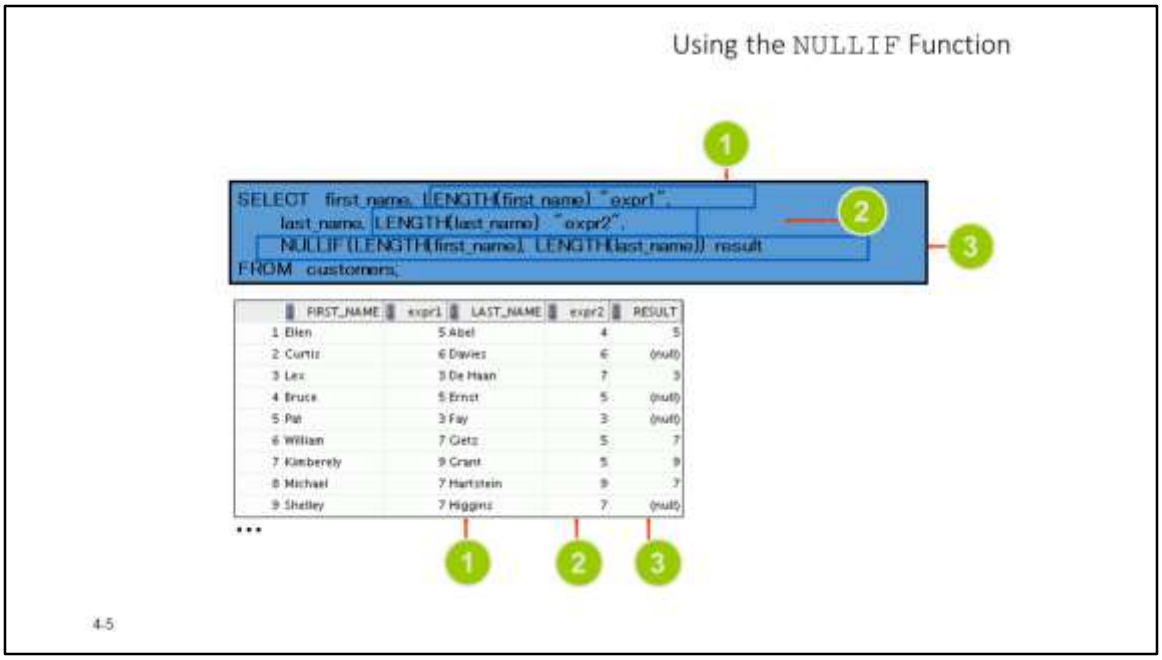
NVL2(*expr1*, *expr2, expr3*)

In the syntax:

- *expr1* is the source value or expression that may contain a null
- *expr2* is the value that is returned if *expr1* is not null
- *expr3* is the value that is returned if *expr1* is null

In the example shown in the slide, the COMMISSION_PCT column is examined. If a value is detected, the text literal value of SAL+COMM is returned. If the COMMISSION_PCT column contains a null value, the text literal value of SAL is returned.

Note: The argument *expr1* can have any data type. The arguments *expr2* and *expr3* can have any data types except LONG.



Using the NULLIF Function

The NULLIF function compares two expressions.

Syntax

```
NULLIF (expr1, expr2)
```

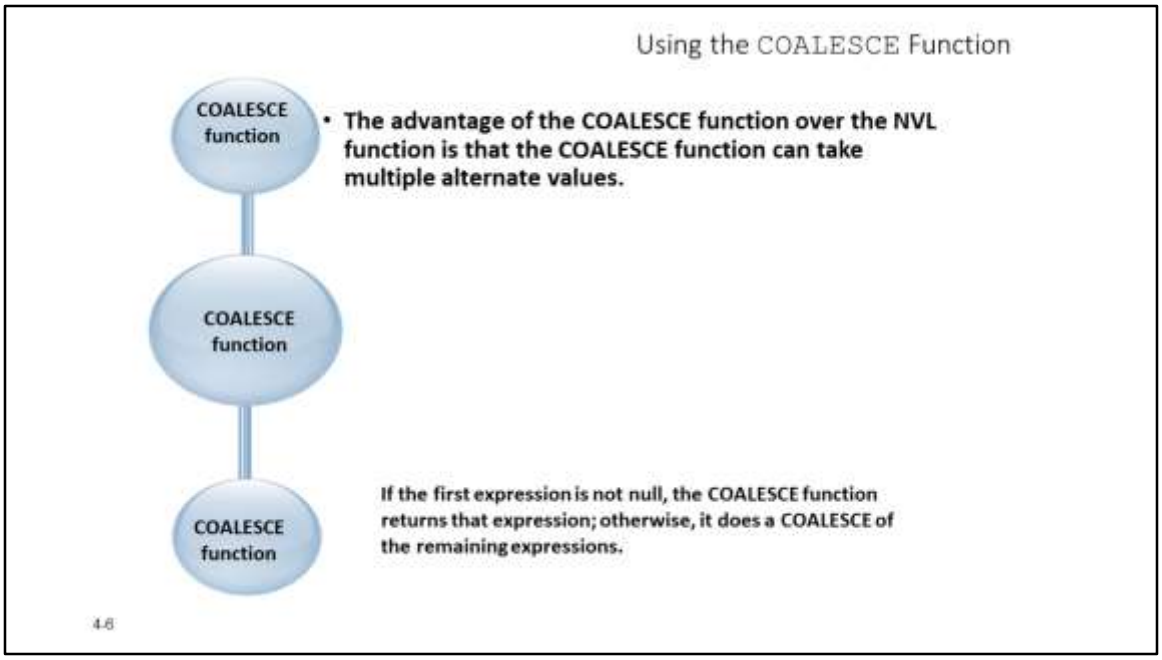
In the syntax:

- NULLIF compares *expr1* and *expr2*. If they are equal, the function returns null. If they are not, the function returns *expr1*. However, you cannot specify the literal NULL for *expr1*.

In the example shown in the slide, the length of the first name in the CUSTOMERS table is compared to the length of the last name in the CUSTOMERS table. When the lengths of the names are equal, a null value is displayed. When the lengths of the names are not equal, the length of the first name is displayed.

Note: The NULLIF function is logically equivalent to the following CASE expression. The CASE expression is discussed on a subsequent page:

```
CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
```



Using the COALESCE Function

The COALESCE function returns the first non-null expression in the list.

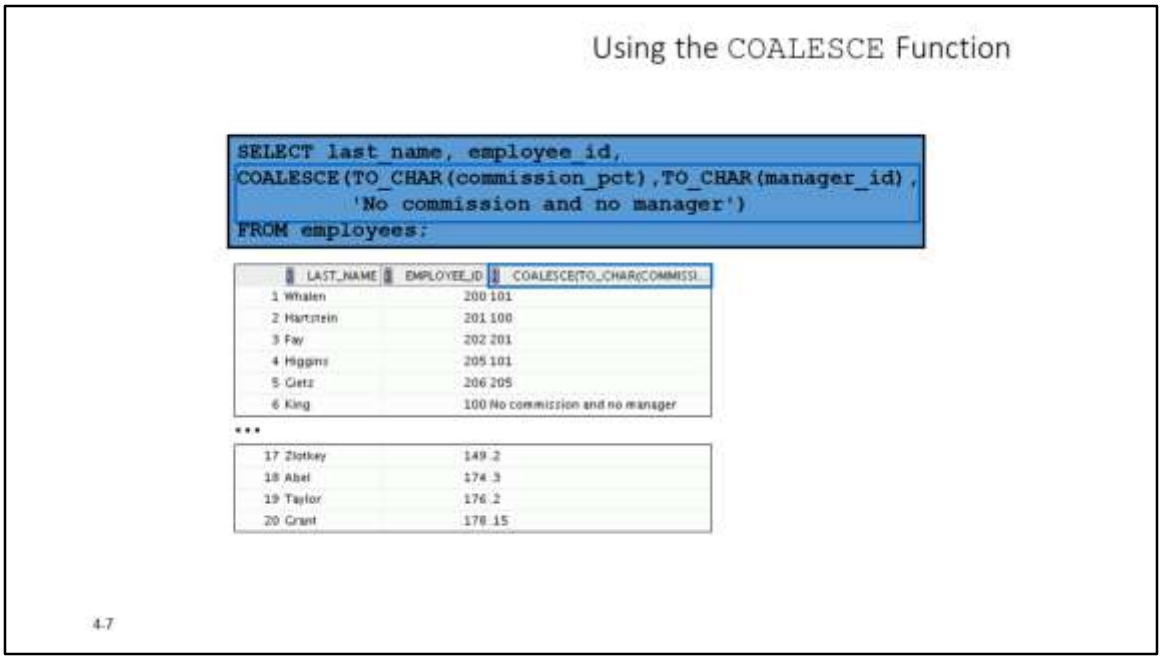
Syntax

COALESCE (*expr1*, *expr2, ... exprn*)

In the syntax:

- *expr1* returns this expression if it is not null
- *expr2* returns this expression if the first expression is null and this expression is not null
- *exprn* returns this expression if the preceding expressions are null

Note that all expressions must be of the same data type.



Using the COALESCE Function (continued)

In the example shown in the slide, if the `manager_id` value is not null, it is displayed. If the `manager_id` value is null, the `commission_pct` is displayed. If the `manager_id` and `commission_pct` values are null, "No commission and no manager" is displayed. Note that `TO_CHAR` function is applied so that all expressions are of the same data type.

Using the COALESCE Function (continued)

Example:

For the employees who do not get any commission, your organization wants to give a salary increment of $2,000 and for employees who get commission, the query should compute the new salary that is equal to the existing salary added to the commission amount.

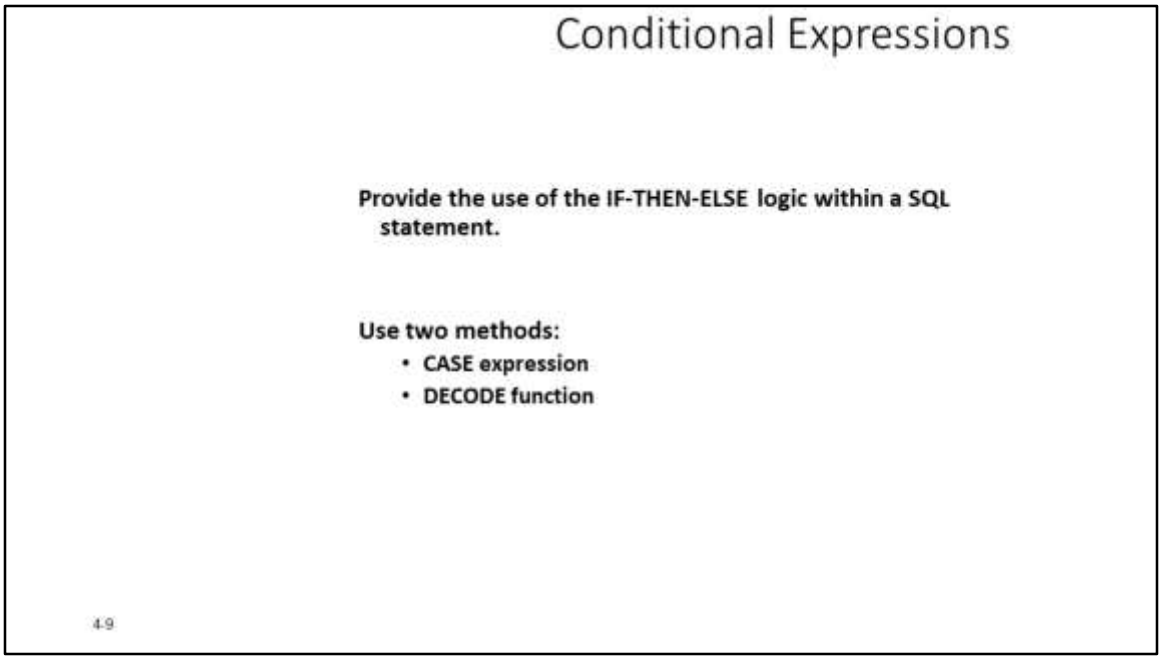
```
SELECT last_name, salary, commission_pct,
 COALESCE((salary+(commission_pct*salary)), salary+2000, salary)
  "New Salary"
FROM   employees;
```

Note: Examine the output. For employees who do not get any commission, the New Salary column shows the salary incremented by $2,000 and for employees who get commission, the New Salary column shows the computed commission amount added to the salary.

	LAST_NAME	SALARY	COMMISSION_PCT	New Salary
1	Whalen	4400	(null)	6400
2	Hartstein	13000	(null)	15000
3	Fay	6000	(null)	8000
4	Higgins	12000	(null)	14000
5	Gietz	8300	(null)	10300
6	King	24000	(null)	26000

...

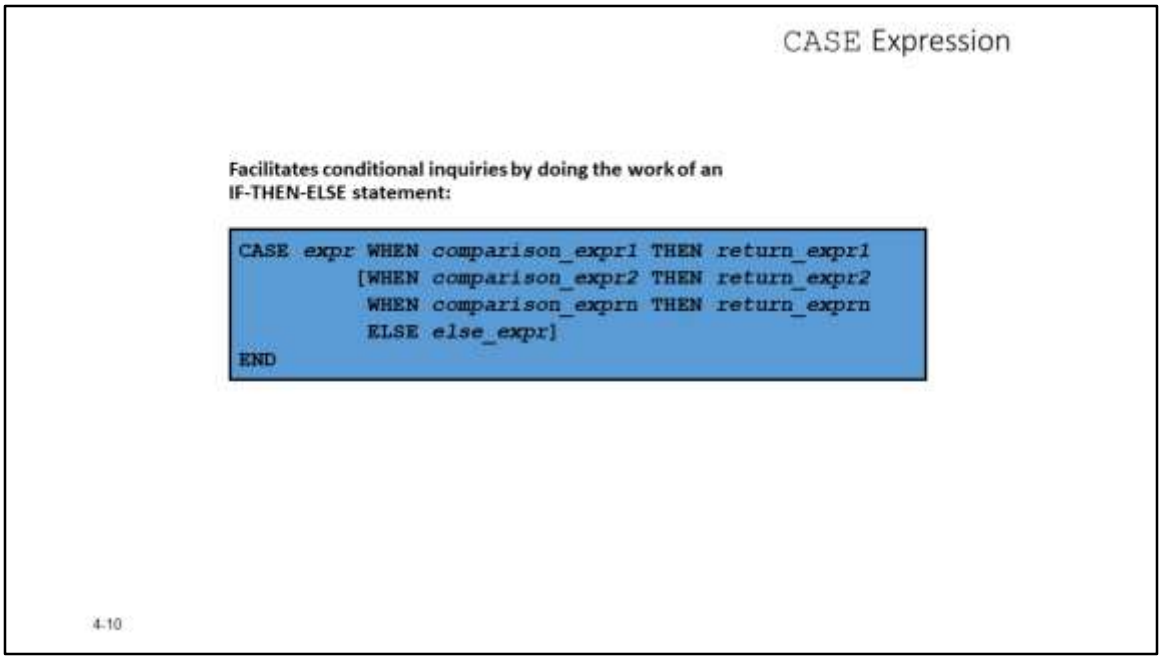
17	Zlotkey	10500	0.2	12600
18	Abel	11000	0.3	14300
19	Taylor	8600	0.2	10320
20	Grant	7000	0.15	8050



Conditional Expressions

The two methods that are used to implement conditional processing (`IF-THEN-ELSE` logic) in a SQL statement are the `CASE` expression and the `DECODE` function.

Note: The `CASE` expression complies with the ANSI SQL. The `DECODE` function is specific to Oracle syntax.



CASE Expression

CASE expressions allow you to use the IF-THEN-ELSE logic in SQL statements without having to invoke procedures.

In a simple CASE expression, the Oracle server searches for the first WHEN ... THEN pair for which expr is equal to comparison_expr and returns return_expr. If none of the WHEN ... THEN pairs meet this condition, and if an ELSE clause exists, the Oracle server returns else_expr. Otherwise, the Oracle server returns a null. You cannot specify the literal NULL for all the return_exprs and the else_expr.

The expressions expr and comparison_expr must be of the same data type, which can be CHAR, VARCHAR2, NCHAR, or NVARCHAR2. All of the return values (return_expr) must be of the same data type.

Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

```
Select order_id, order_status,order_total,
CASE order_status when 0 then 1.1*order_total
                 when 1 then 2*order_total
                 when 2 then 3*order_total
                 when 3 then 4*order_total
    ELSE order_status END   "REVISED SALARY"
FROM orders;
```

	ORDER_ID	ORDER_STATUS	ORDER_TOTAL	REVISED SALARY
1	2458	0	70647.34	77712.074
2	2397	1	42281.2	84562.4
3	2454	1	6653.4	13306.8
4	2354	0	46157	50882.7
5	2358	2	7826	23478
6	2381	3	23034.6	92138.4
7	2440	3	63693.44	254762.64

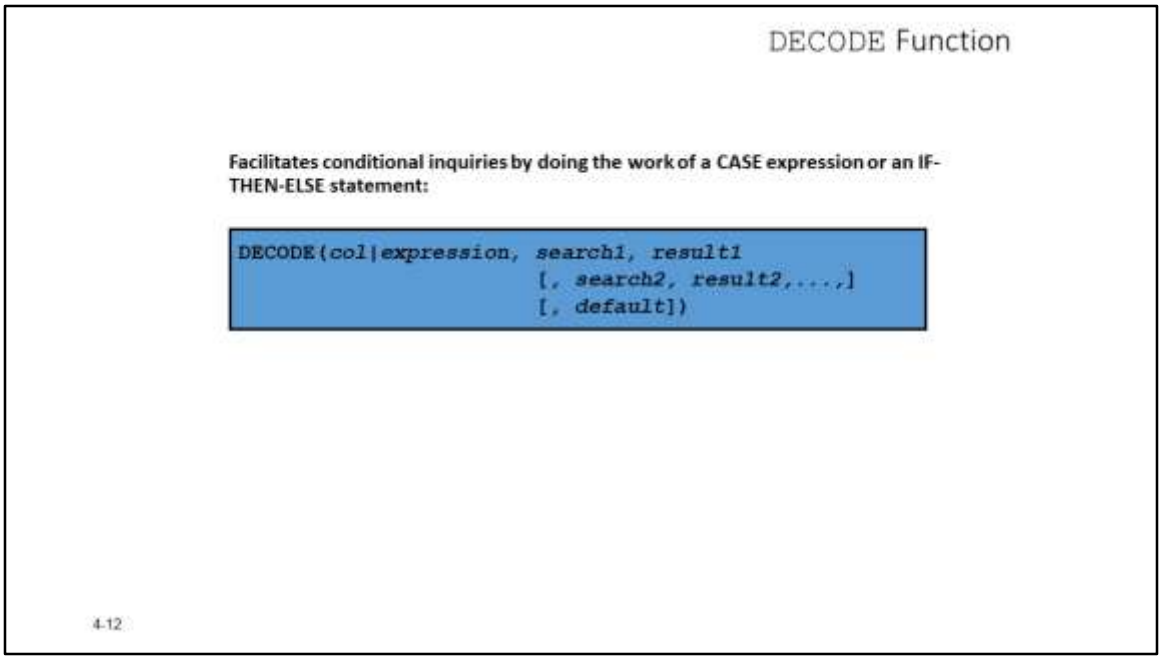
...

4-11

Using the `CASE` Expression

In the SQL statement in the slide, the value of `order_status` is decoded. If `order_status` is 0, then the order_total increase is 1.1; if order_status is `1`, the order_total increase is 2; if `order_status` is `3`, the order_total increase is 4. For all other order_status, there is no increase in order_total.

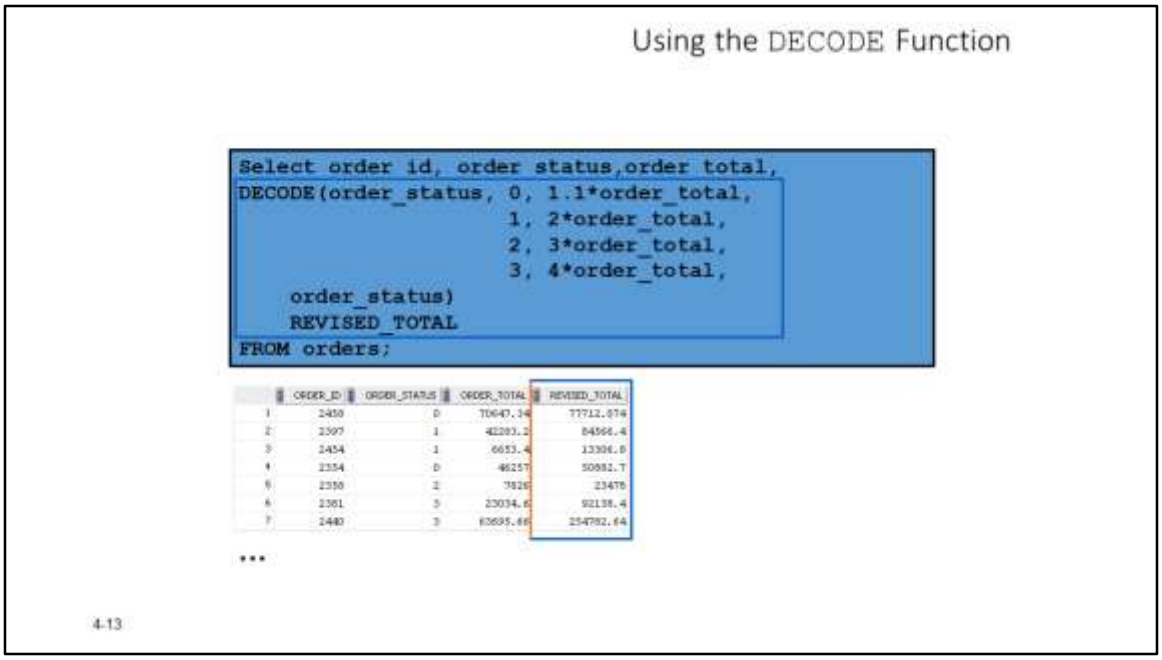
The same statement can be written with the `DECODE` function.



DECODE Function

The DECODE function decodes an expression in a way similar to the IF-THEN-ELSE logic that is used in various languages. The DECODE function decodes *expression* after comparing it to each *search* value. If the expression is the same as *search*, *result* is returned.

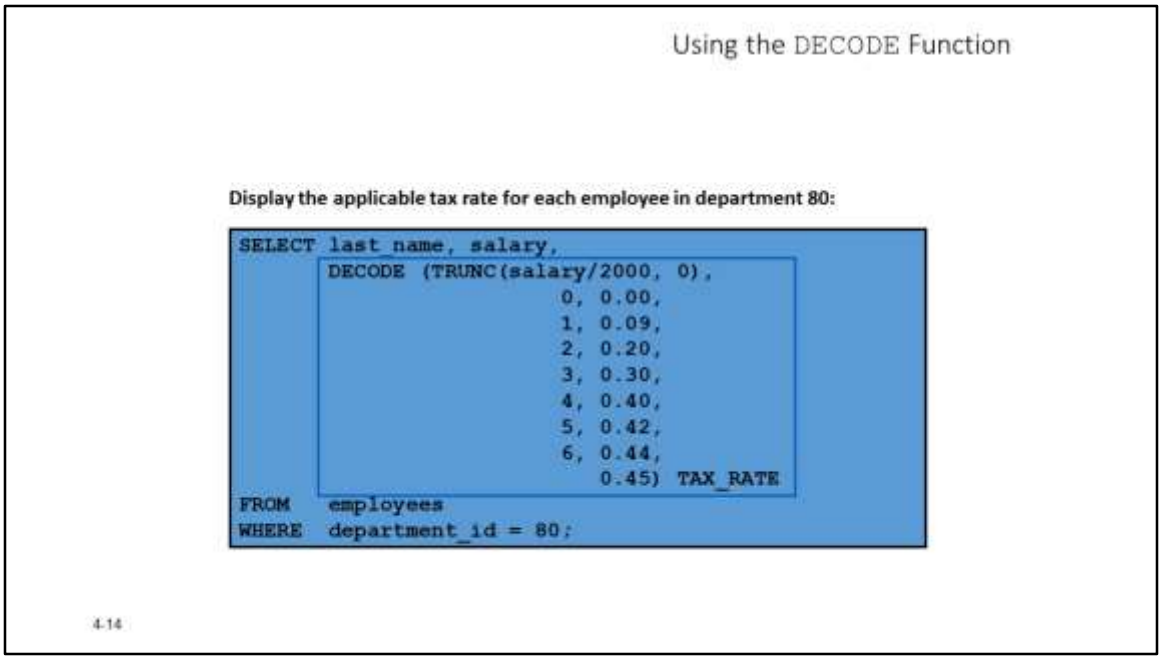
If the default value is omitted, a null value is returned where a search value does not match any of the result values.



Using the `DECODE` Function

In the SQL statement in the slide, the value of `order_status` is decoded. If `order_status` is 0, then the order_total increase is 1.1; if order_status is `1`, the order_total increase is 2; if `order_status` is `3`, the order_total increase is 4. For all other order_status, there is no increase in order_total. The same statement can be expressed in pseudocode as an `IF-THEN-ELSE` statement:

```
IF order_status = 0   THEN   = order_status *1.10
IF order_status = 1   THEN order_status = order_status *2
IF order_status = 2   THEN order_status = order_status *3
IF order_status = 3   THEN order_status = order_status
ELSE  order_status = order_status
```



Using the DECODE Function (continued)

This slide shows another example using the DECODE function. In this example, you determine the tax rate for each employee in department 80 based on the monthly salary. The tax rates are as follows:

Monthly Salary Range	***Tax Rate***
$0.00–1,999.99	00%
$2,000.00–3,999.99	09%
$4,000.00–5,999.99	20%
$6,000.00–7,999.99	30%
$8,000.00–9,999.99	40%
$10,000.00–11,999.99	42%
$12,200.00–13,999.99	44%
$14,000.00 or greater	45%

	LAST_NAME	SALARY	TAX_RATE
1	Zlotkey	10500	0.42
2	Abel	11000	0.42
3	Taylor	8600	0.4

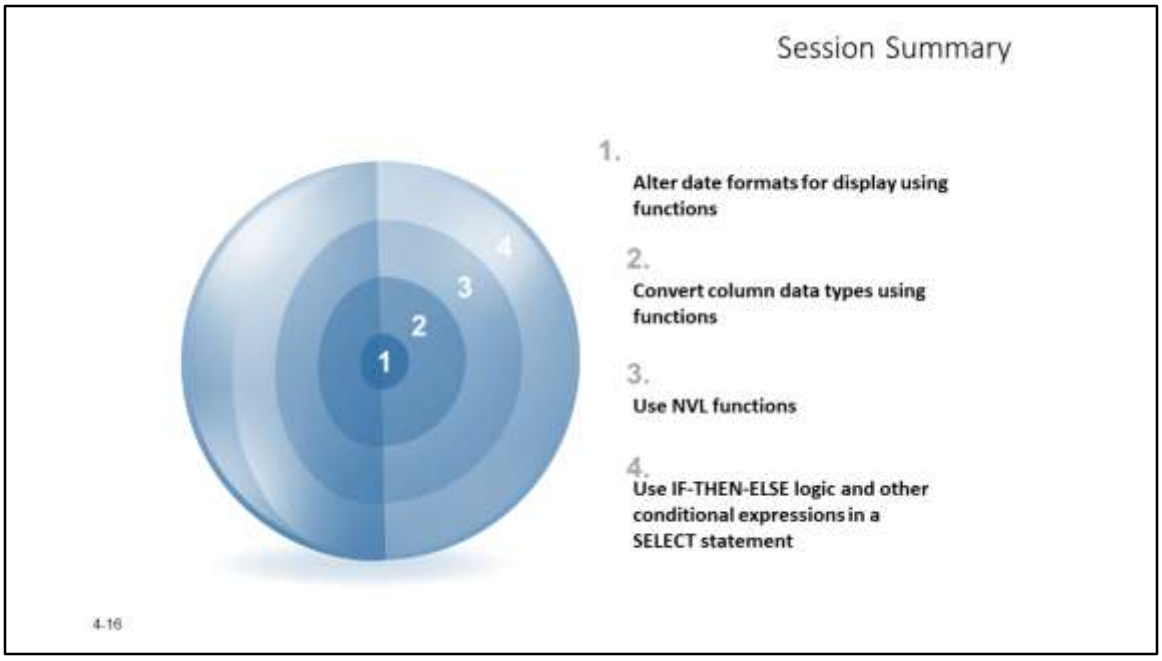
Quiz

The TO_NUMBER function converts either character strings or date values to a number in the format specified by the optional format model.

- True
- False

4-15

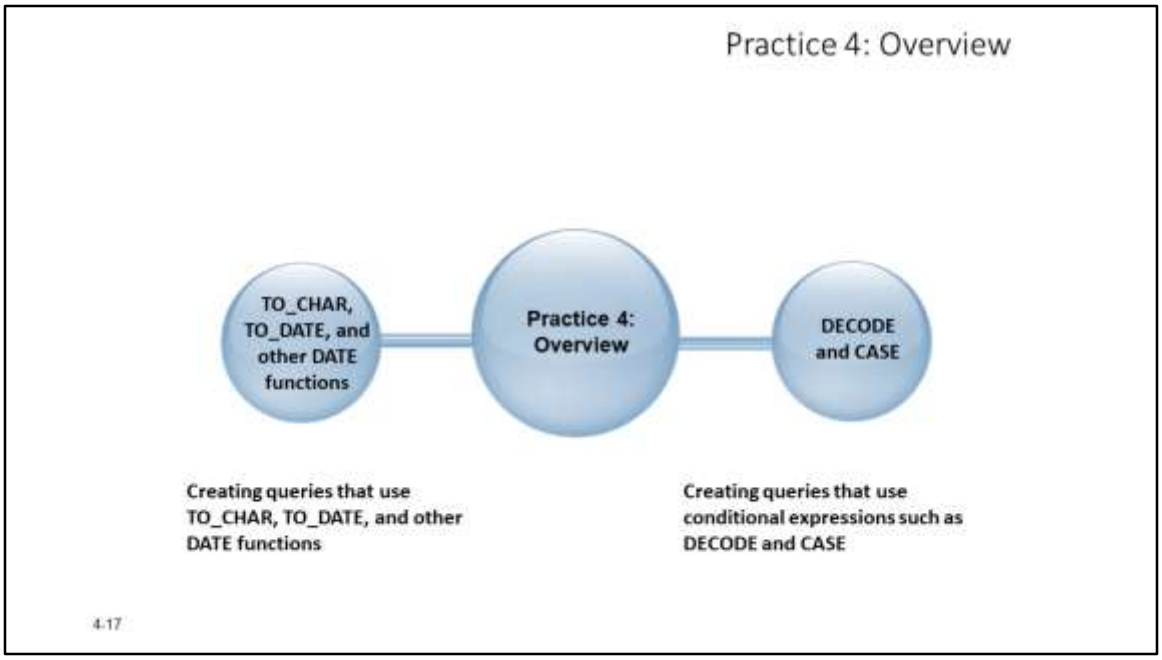
Answer: 2



Session Summary

Remember the following:

- Conversion functions can convert character, date, and numeric values: `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- There are several functions that pertain to nulls, including `NVL`, `NVL2`, `NULLIF`, and `COALESCE`.
- The `IF-THEN-ELSE` logic can be applied within a SQL statement by using the `CASE` expression or the `DECODE` function.



Practice 4: Overview

This practice provides a variety of exercises using `TO_CHAR` and `TO_DATE` functions, and conditional expressions such as `DECODE` and `CASE`. Remember that for nested functions, the results are evaluated from the innermost function to the outermost function.