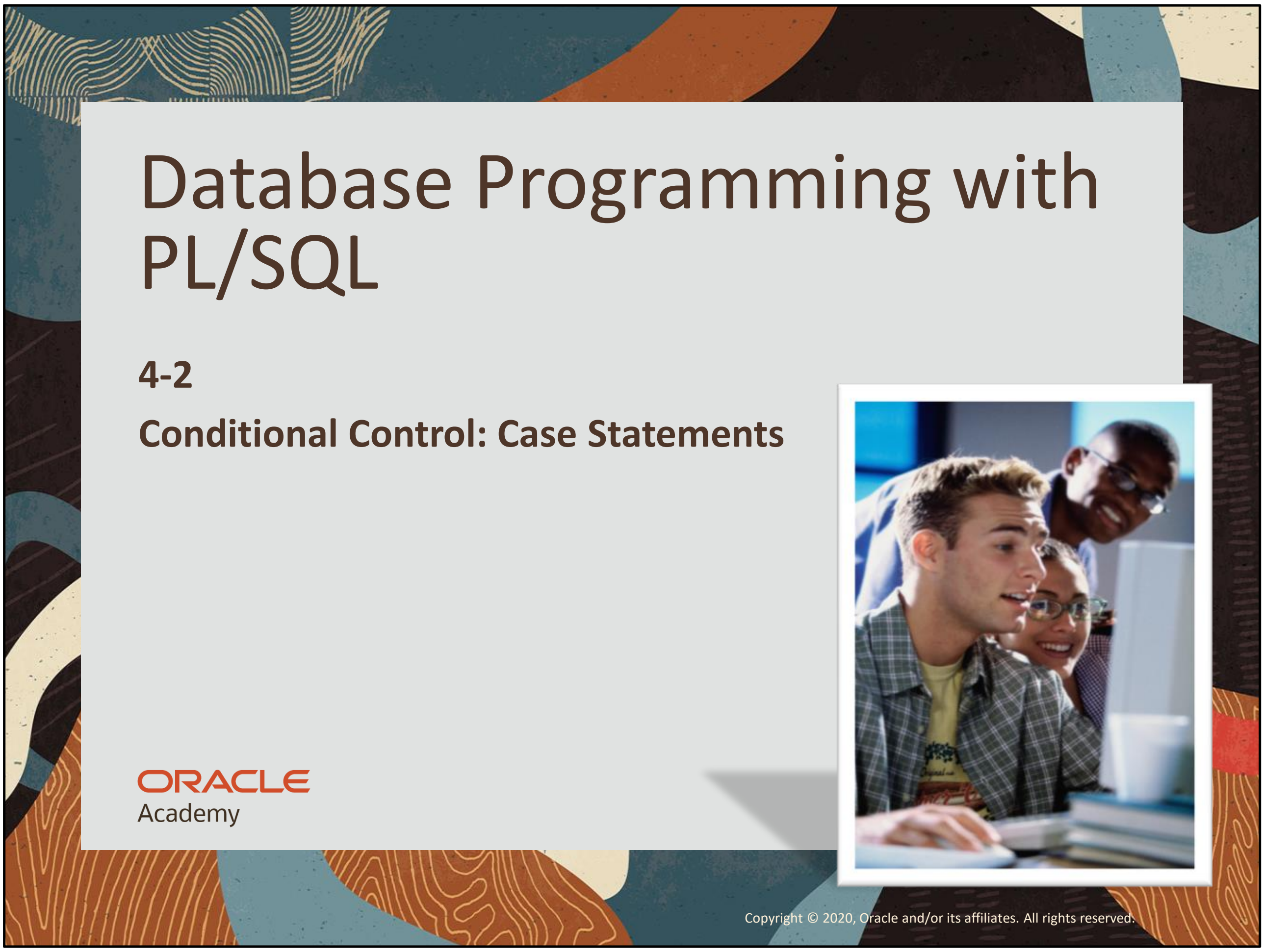
ORACLE

Academy



Database Programming with PL/SQL

4-2

Conditional Control: Case Statements

ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Construct and use CASE statements in PL/SQL
  - Construct and use CASE expressions in PL/SQL
  - Include the correct syntax to handle null conditions in PL/SQL CASE statements
  - Include the correct syntax to handle Boolean conditions in PL/SQL IF and CASE statements

Purpose

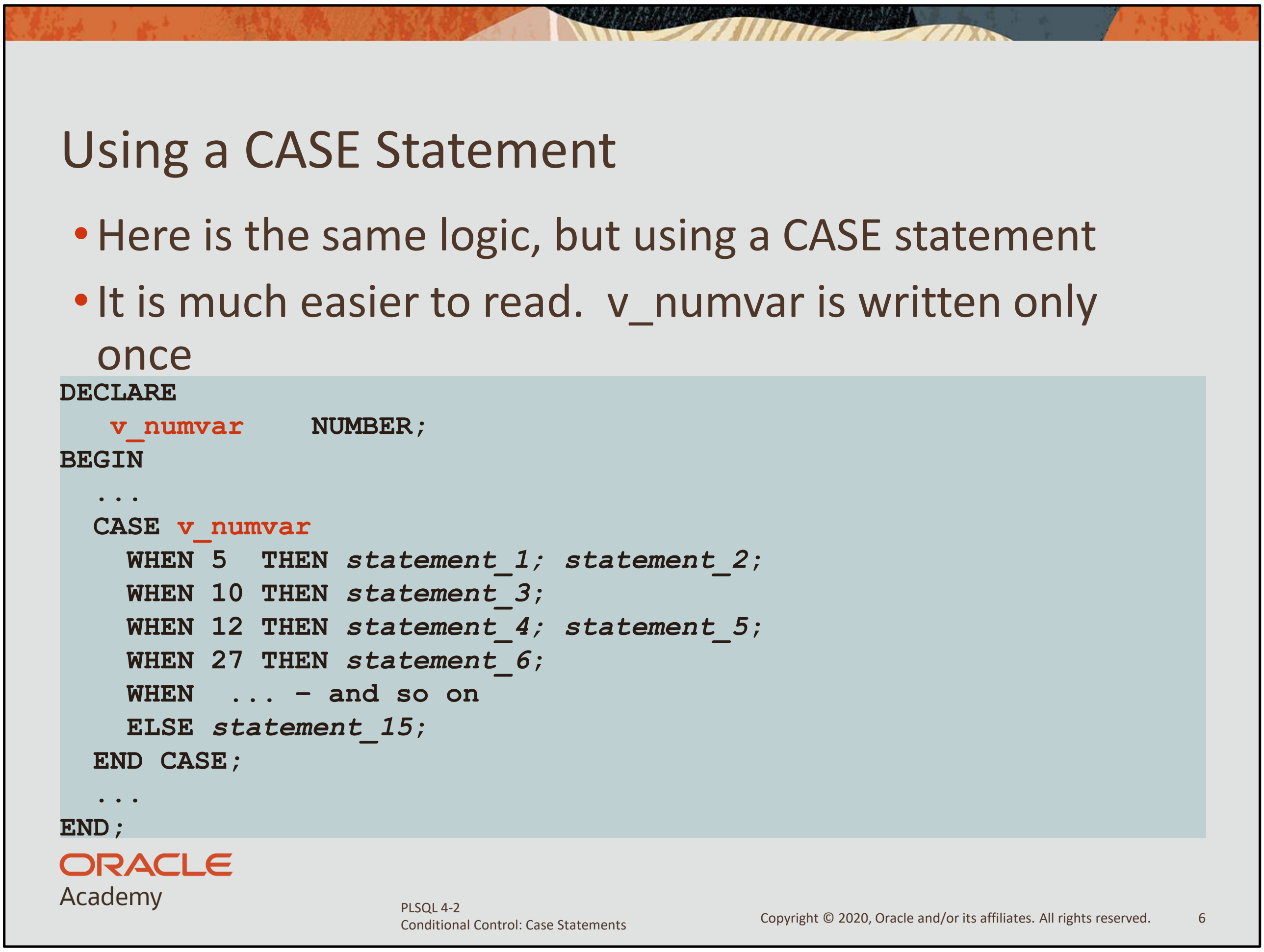
- In this lesson, you learn how to use CASE statements and CASE expressions in a PL/SQL block
- CASE STATEMENTS are similar to IF statements, but are often easier to write and easier to read
- CASE EXPRESSIONS work like functions to return one value from a number of values into a variable

Using a CASE Statement

- Look at this IF statement. What do you notice?
- All the conditions test the same variable v_numvar
- And the coding is very repetitive: v_numvar is coded many times

```
DECLARE
   v_numvar     NUMBER;
BEGIN
  ...
  IF     v_numvar = 5  THEN statement_1; statement_2;
    ELSIF v_numvar = 10 THEN statement_3;
    ELSIF v_numvar = 12 THEN statement_4; statement_5;
    ELSIF v_numvar = 27 THEN statement_6;
    ELSIF v_numvar ... – and so on
    ELSE  statement_15;
  END IF;    ...
END;
```

ORACLE
Academy

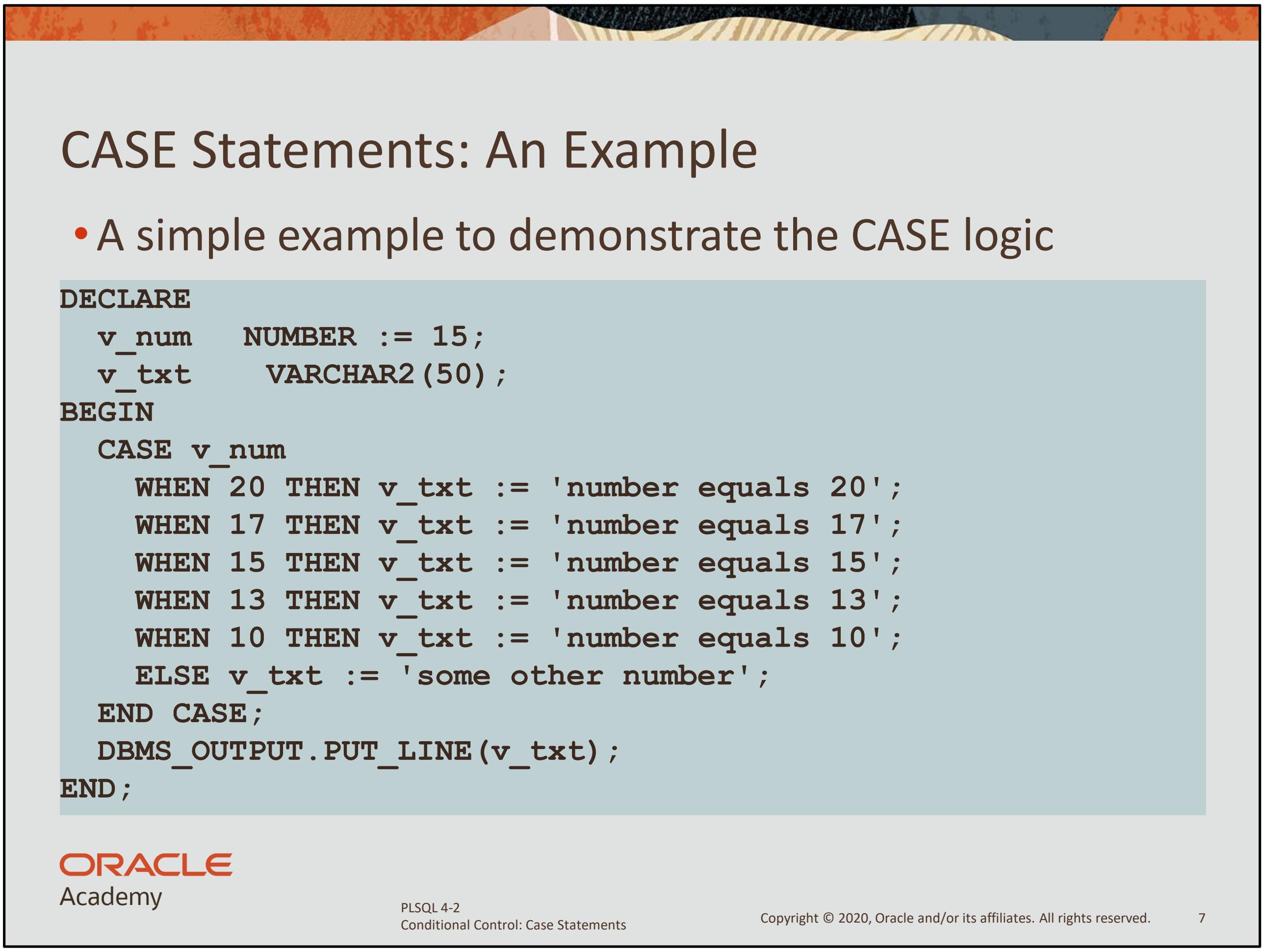


Using a CASE Statement

- Here is the same logic, but using a CASE statement
- It is much easier to read. v_numvar is written only once

```
DECLARE
   v_numvar     NUMBER;
BEGIN
  ...
  CASE v_numvar
    WHEN 5  THEN statement_1; statement_2;
    WHEN 10 THEN statement_3;
    WHEN 12 THEN statement_4; statement_5;
    WHEN 27 THEN statement_6;
    WHEN   ... – and so on
    ELSE statement_15;
  END CASE;
  ...
END;
```

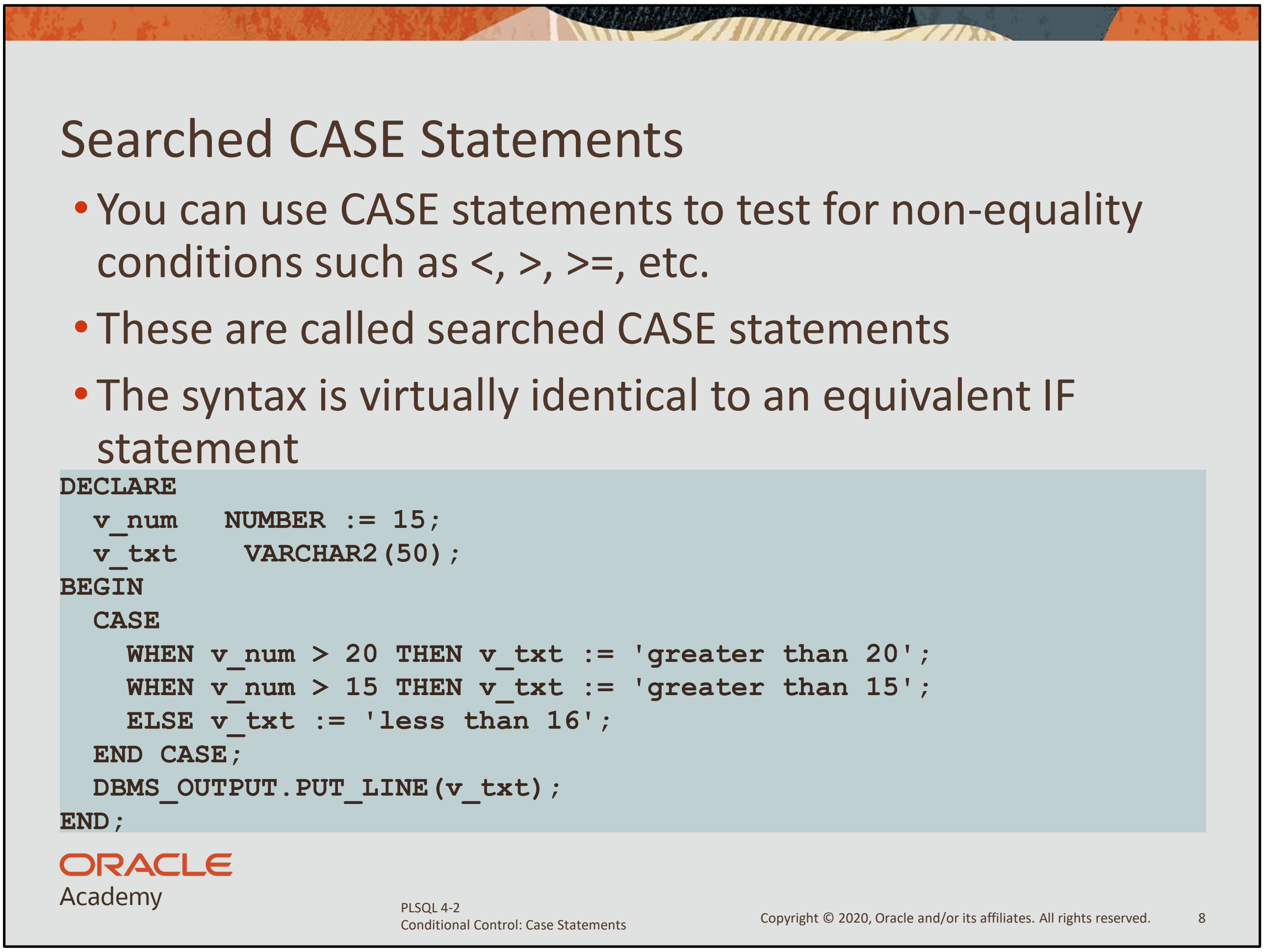
The WHEN clauses, “WHEN 5,” “WHEN 10,” “WHEN 12,” etc. mean: WHEN v_numvar is equal to 5, and WHEN v_numvar is equal to 10, etc.



CASE Statements: An Example

- A simple example to demonstrate the CASE logic

```
DECLARE
  v_num   NUMBER := 15;
  v_txt    VARCHAR2(50);
BEGIN
  CASE v_num
    WHEN 20 THEN v_txt := 'number equals 20';
    WHEN 17 THEN v_txt := 'number equals 17';
    WHEN 15 THEN v_txt := 'number equals 15';
    WHEN 13 THEN v_txt := 'number equals 13';
    WHEN 10 THEN v_txt := 'number equals 10';
    ELSE v_txt := 'some other number';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_txt);
END;
```



Searched CASE Statements

- You can use CASE statements to test for non-equality conditions such as <, >, >=, etc.
- These are called searched CASE statements
- The syntax is virtually identical to an equivalent IF statement

```
DECLARE
  v_num   NUMBER := 15;
  v_txt    VARCHAR2(50);
BEGIN
  CASE
    WHEN v_num > 20 THEN v_txt := 'greater than 20';
    WHEN v_num > 15 THEN v_txt := 'greater than 15';
    ELSE v_txt := 'less than 16';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_txt);
END;
```

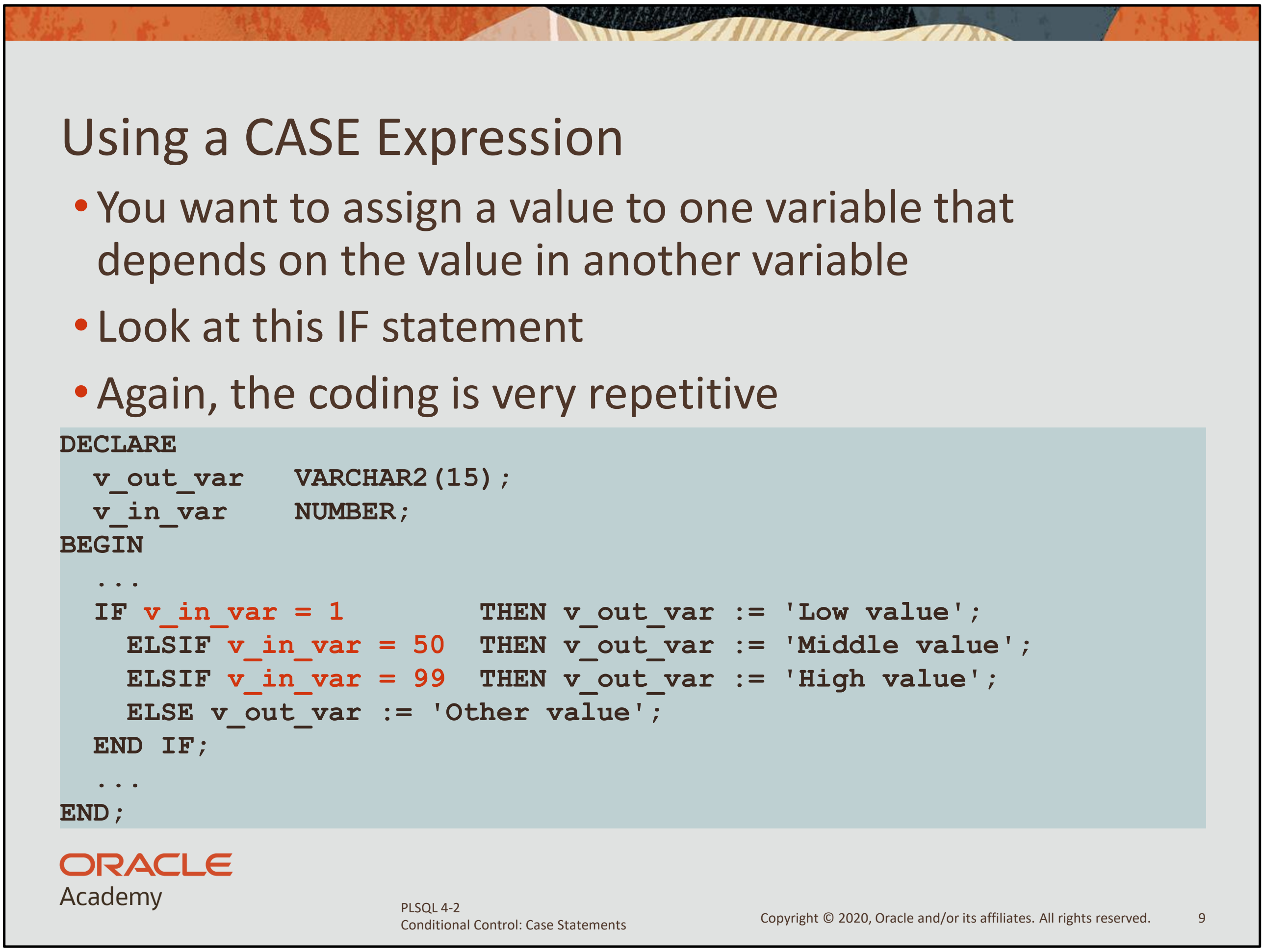
ORACLE
Academy

PLSQL 4-2
Conditional Control: Case Statements

8

The same logic from above, rewritten using an IF statement:

```
DECLARE
  v_num        NUMBER := 15;
  v_txt         VARCHAR2(50);
BEGIN
  IF v_num > 20 THEN v_txt := 'greater than 20';
    ELSIF v_num > 15 THEN v_txt := 'greater than 15';
    ELSE v_txt := 'less than 16';
  END IF;
  DBMS_OUTPUT.PUT_LINE(v_txt);
END;
```



Using a CASE Expression

- You want to assign a value to one variable that depends on the value in another variable
- Look at this IF statement
- Again, the coding is very repetitive

```
DECLARE
  v_out_var   VARCHAR2(15);
  v_in_var    NUMBER;
BEGIN
  ...
  IF v_in_var = 1        THEN v_out_var := 'Low value';
    ELSIF v_in_var = 50  THEN v_out_var := 'Middle value';
    ELSIF v_in_var = 99  THEN v_out_var := 'High value';
    ELSE v_out_var := 'Other value';
  END IF;
  ...
END;
```

ORACLE
Academy

Using a CASE Expression

- Here is the same logic, but using a CASE expression:

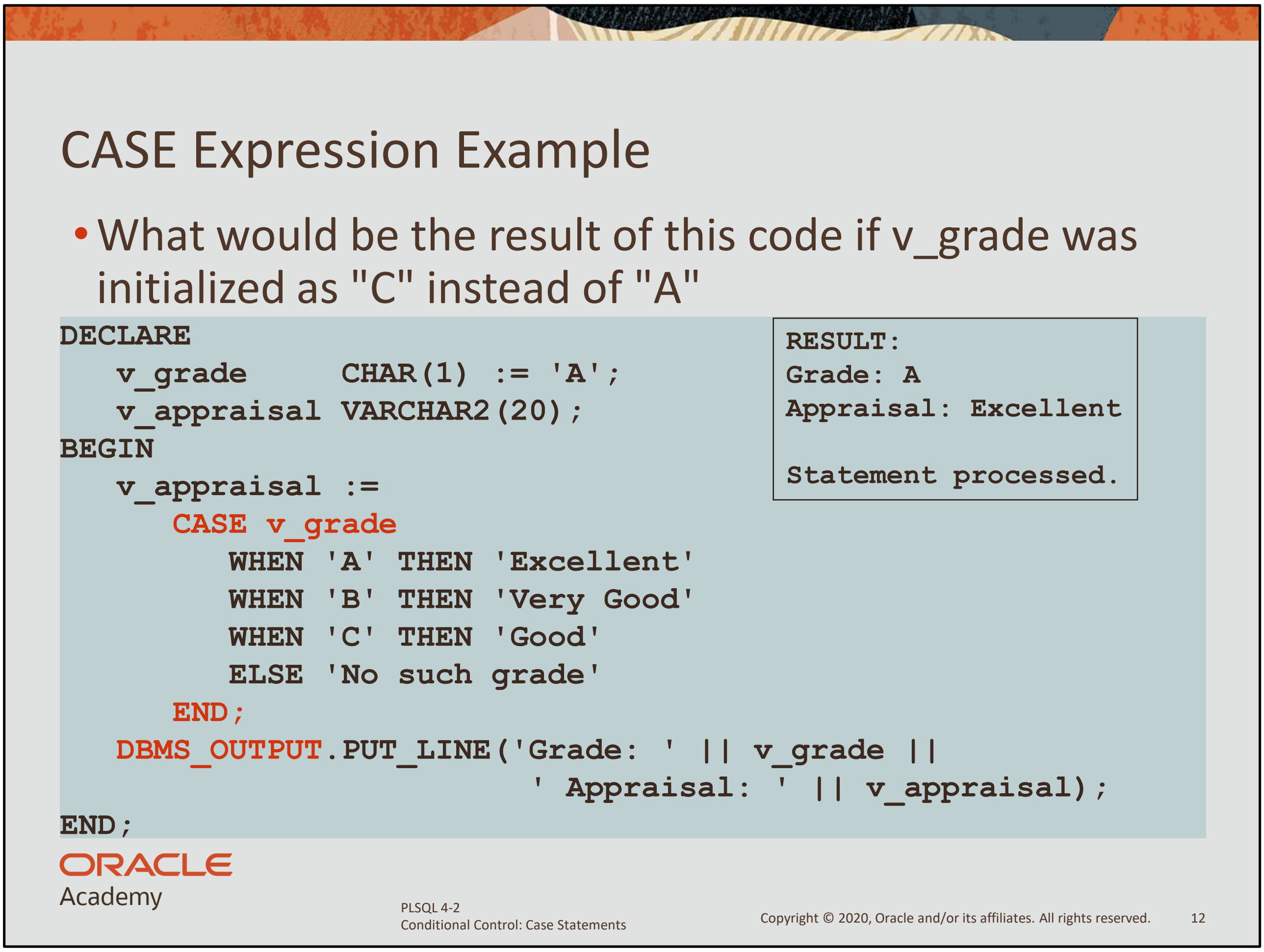
```
DECLARE
  v_out_var   VARCHAR2(15);
  v_in_var    NUMBER;
BEGIN
  ...
  v_out_var := CASE v_in_var
      WHEN 1  THEN 'Low value'
      WHEN 50 THEN 'Middle value'
      WHEN 99 THEN 'High value'
      ELSE         'Other value'
    END;
  ...
END;
```

ORACLE
Academy

CASE Expression Syntax

- A CASE expression is different from a CASE statement because it selects one of a number of results and assigns it to a variable
- A CASE expression ends with END not END CASE
- In the syntax, expressionN can be a literal value or an expression such as (v_other_var * 2)

```
variable_name :=
  CASE selector
    WHEN expression1 THEN result1
    WHEN expression2 THEN result2
    ...
    WHEN expressionN THEN resultN
   [ELSE resultN+1]
  END;
```



CASE Expression Example

- What would be the result of this code if v_grade was initialized as "C" instead of "A"

```
DECLARE
   v_grade     CHAR(1) := 'A';
   v_appraisal VARCHAR2(20);
BEGIN
   v_appraisal :=
      CASE v_grade
         WHEN 'A' THEN 'Excellent'
         WHEN 'B' THEN 'Very Good'
         WHEN 'C' THEN 'Good'
         ELSE 'No such grade'
      END;
   DBMS_OUTPUT.PUT_LINE('Grade: ' || v_grade ||
                         ' Appraisal: ' || v_appraisal);
END;
```

```
RESULT:
Grade: A
Appraisal: Excellent

Statement processed.
```

ORACLE
Academy

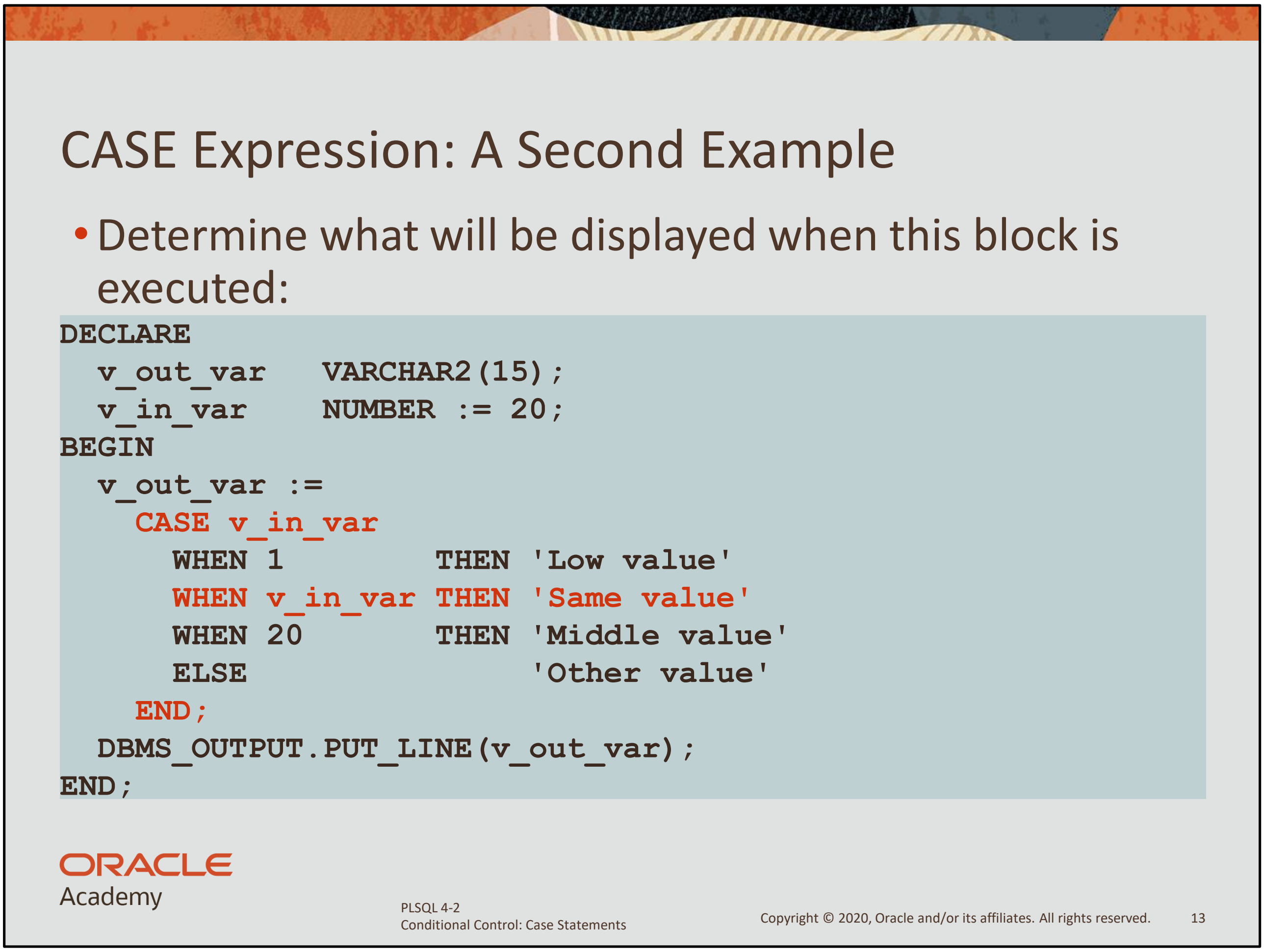
PLSQL 4-2
Conditional Control: Case Statements

Answer:

Grade: C

Appraisal: Good

Statement processed.

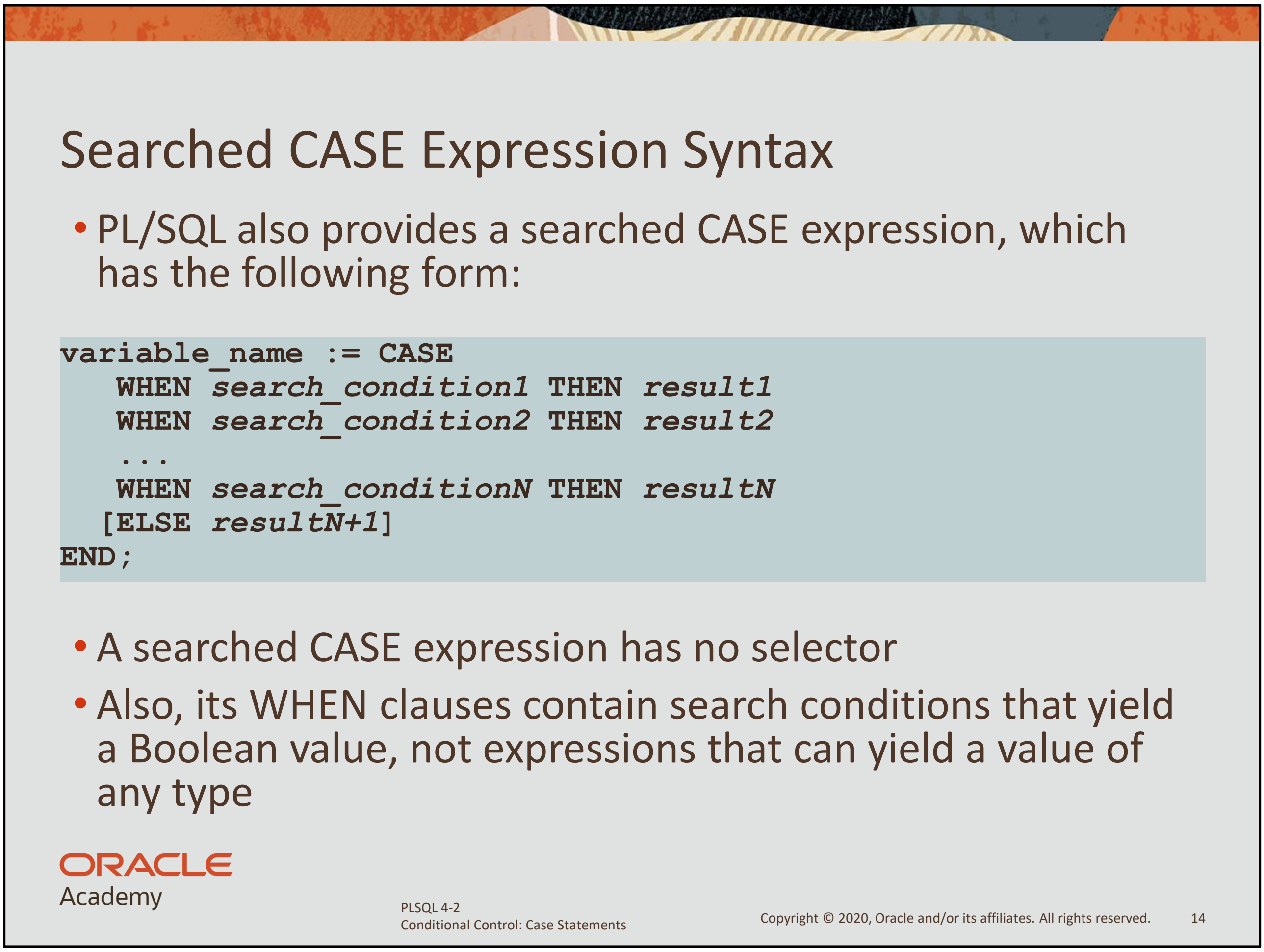


CASE Expression: A Second Example

- Determine what will be displayed when this block is executed:

```
DECLARE
  v_out_var   VARCHAR2(15);
  v_in_var    NUMBER := 20;
BEGIN
  v_out_var :=
    CASE v_in_var
      WHEN 1         THEN 'Low value'
      WHEN v_in_var THEN 'Same value'
      WHEN 20        THEN 'Middle value'
      ELSE                'Other value'
    END;
  DBMS_OUTPUT.PUT_LINE(v_out_var);
END;
```

Answer: "Same value" will be displayed. Remember, read the highlighted WHEN clause as, "When v_in_var is equal to v_in_var, then assign "Same value" to v_out_var and skip to the line following the CASE expression end." Since v_in_var is equal to itself, the code will display the phrase, "Same value."



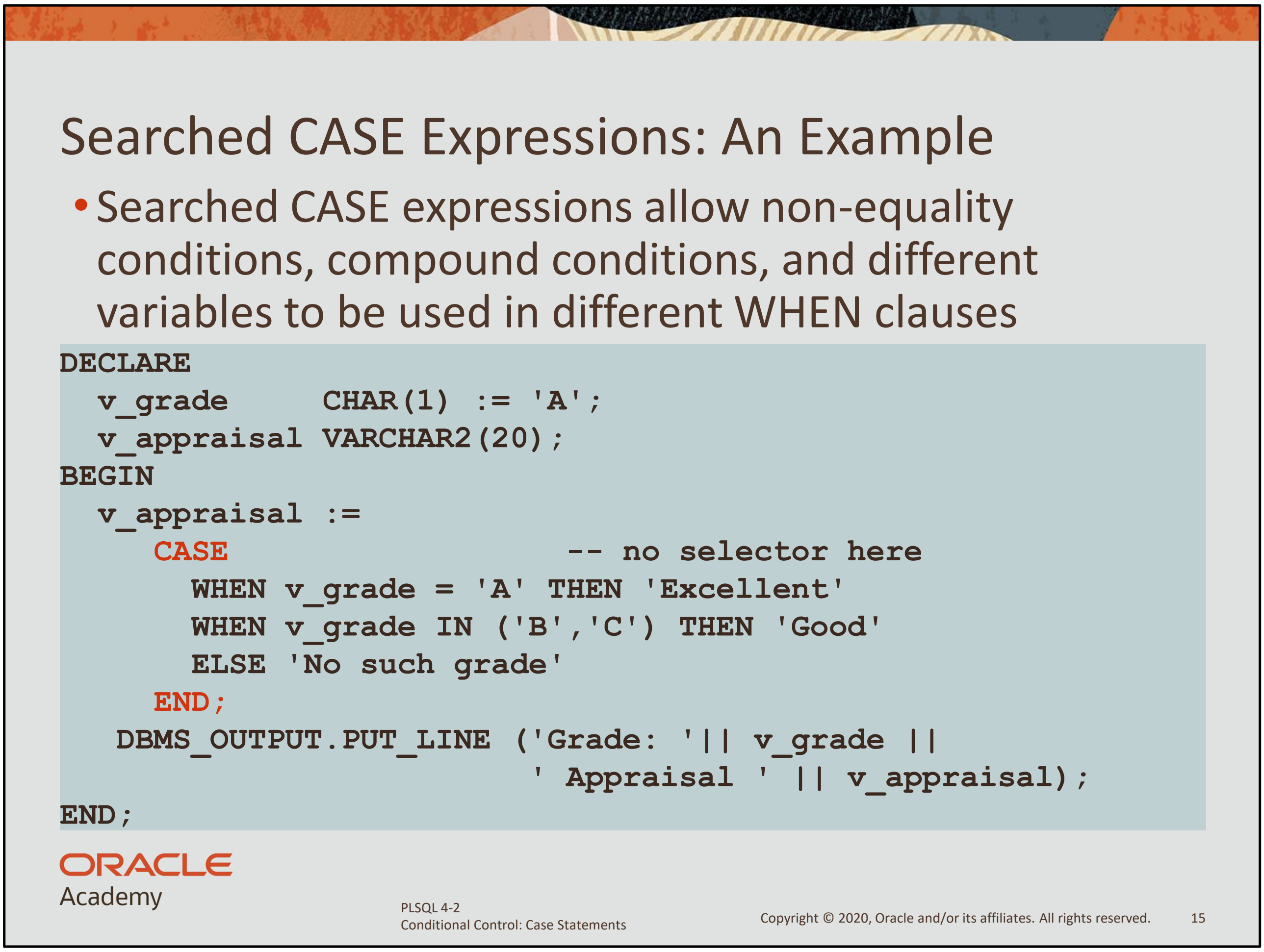
Searched CASE Expression Syntax

- PL/SQL also provides a searched CASE expression, which has the following form:

```
variable_name := CASE
   WHEN search_condition1 THEN result1
   WHEN search_condition2 THEN result2
   ...
   WHEN search_conditionN THEN resultN
  [ELSE resultN+1]
END;
```

- A searched CASE expression has no selector
- Also, its WHEN clauses contain search conditions that yield a Boolean value, not expressions that can yield a value of any type

Searched CASE expressions are more flexible, allowing non-equality conditions (and compound conditions) to be tested and different variables to be used in different WHEN clauses.



Searched CASE Expressions: An Example

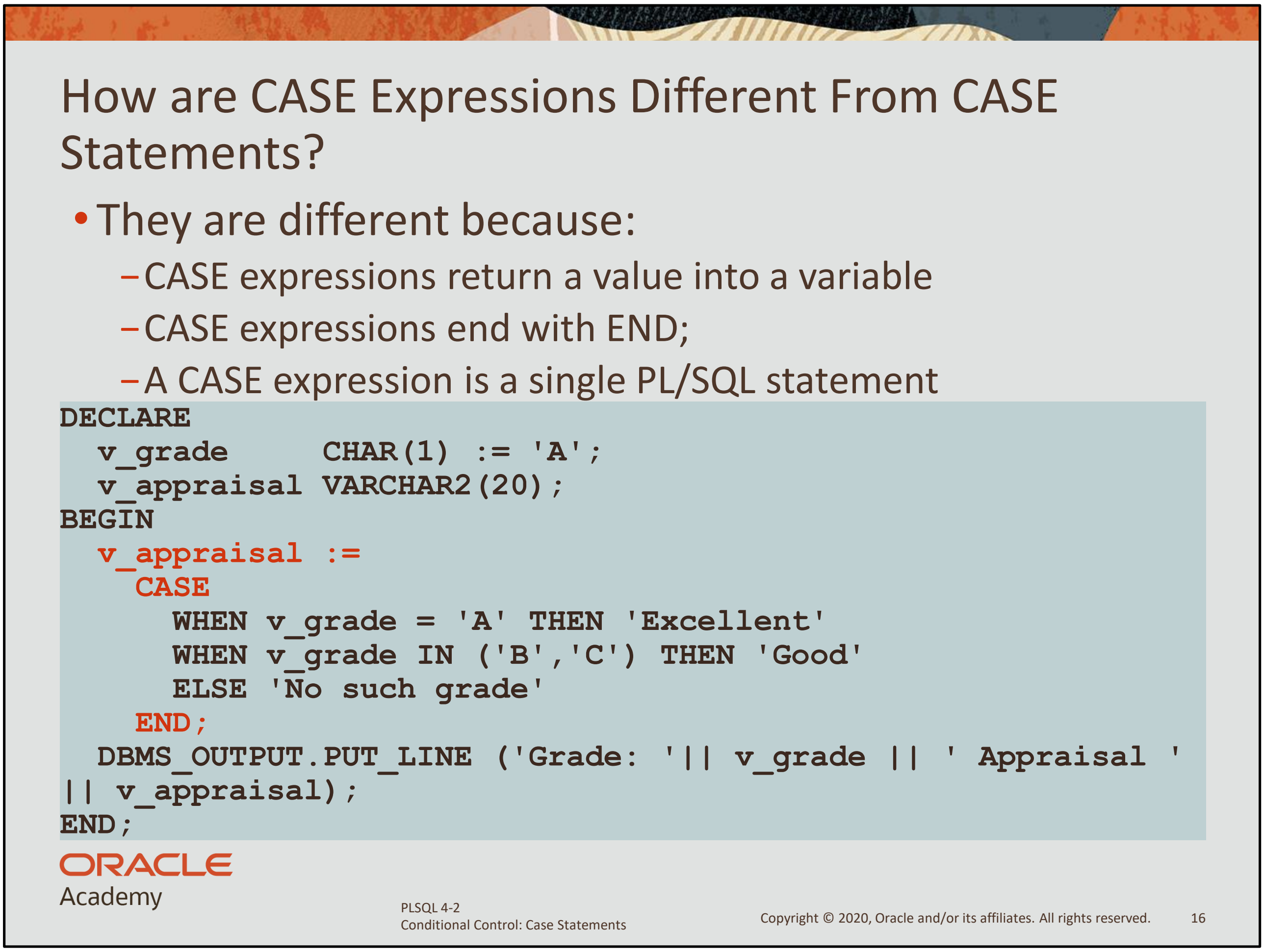
- Searched CASE expressions allow non-equality conditions, compound conditions, and different variables to be used in different WHEN clauses

```
DECLARE
  v_grade     CHAR(1) := 'A';
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal :=
     CASE                  -- no selector here
       WHEN v_grade = 'A' THEN 'Excellent'
       WHEN v_grade IN ('B','C') THEN 'Good'
       ELSE 'No such grade'
     END;
   DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade ||
                         ' Appraisal ' || v_appraisal);
END;
```

ORACLE
Academy

We could have used a non-searched CASE expression. The example in the slide could be written as:

```
v_appraisal :=
  CASE v_ grade
    WHEN ‘A’ THEN ‘Excellent’
    WHEN ‘B’ THEN ‘Good’
    WHEN ‘C’ THEN ‘Good’
    ELSE ‘No such grade’
  END;
```



How are CASE Expressions Different From CASE Statements?

- They are different because:
  - CASE expressions return a value into a variable
  - CASE expressions end with END;
  - A CASE expression is a single PL/SQL statement

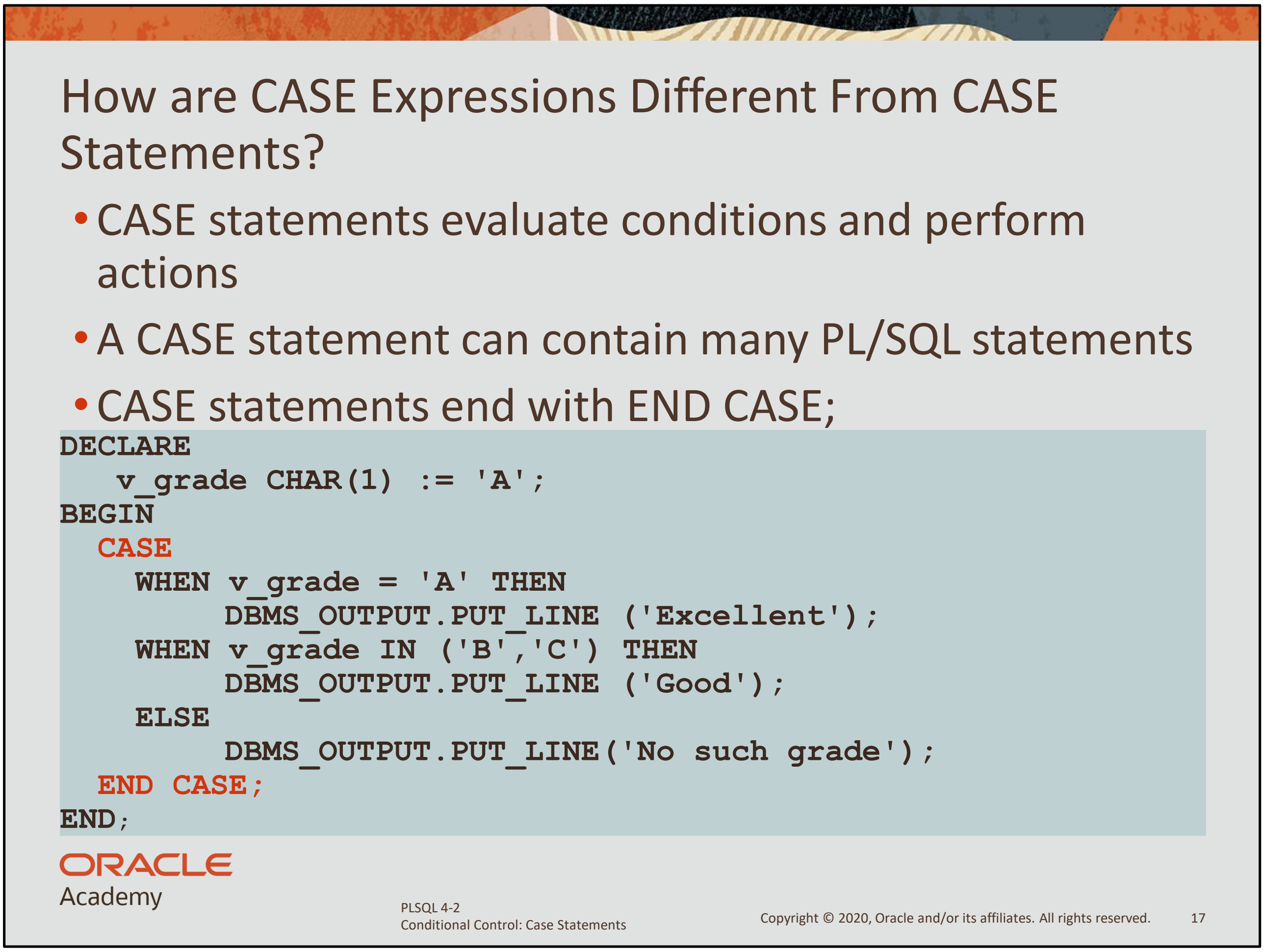
```
DECLARE
  v_grade     CHAR(1) := 'A';
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal :=
    CASE
      WHEN v_grade = 'A' THEN 'Excellent'
      WHEN v_grade IN ('B','C') THEN 'Good'
      ELSE 'No such grade'
    END;
  DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade || ' Appraisal '
|| v_appraisal);
END;
```

ORACLE
Academy

CASE expressions are actually functions, which always return exactly one value.

A common, built-in function is SYSDATE. Just like a CASE expression, SYSDATE assigns one value to a variable.

v_date := SYSDATE;



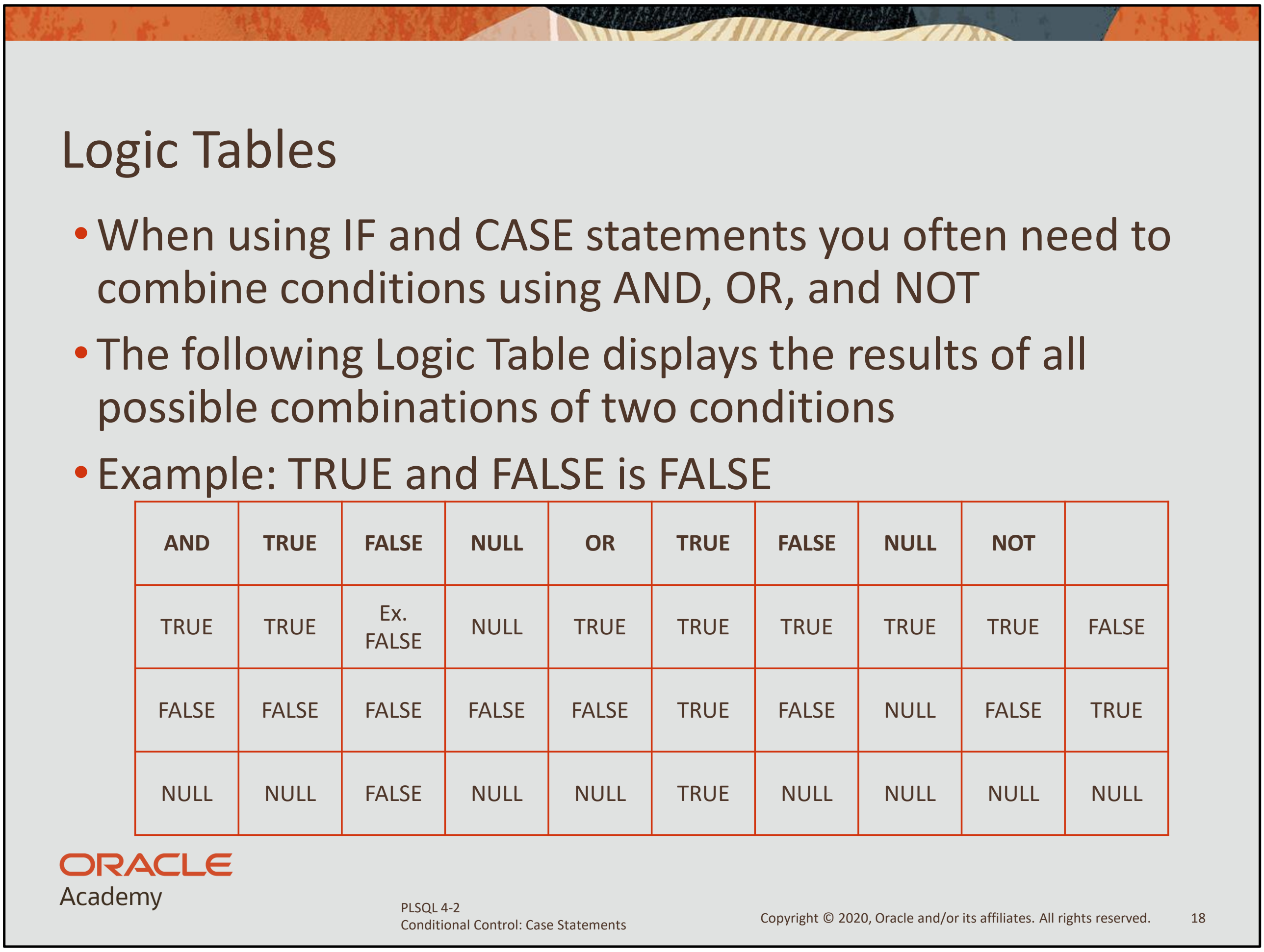
How are CASE Expressions Different From CASE Statements?

- CASE statements evaluate conditions and perform actions
- A CASE statement can contain many PL/SQL statements
- CASE statements end with END CASE;

```
DECLARE
   v_grade CHAR(1) := 'A';
BEGIN
  CASE
    WHEN v_grade = 'A' THEN
         DBMS_OUTPUT.PUT_LINE ('Excellent');
    WHEN v_grade IN ('B','C') THEN
         DBMS_OUTPUT.PUT_LINE ('Good');
    ELSE
         DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
```

ORACLE Academy

A CASE expression has only one terminating semicolon at the END;, while a CASE statement has a semicolon after each executable statement in a WHEN clause.



Logic Tables

- When using IF and CASE statements you often need to combine conditions using AND, OR, and NOT
- The following Logic Table displays the results of all possible combinations of two conditions
- Example: TRUE and FALSE is FALSE

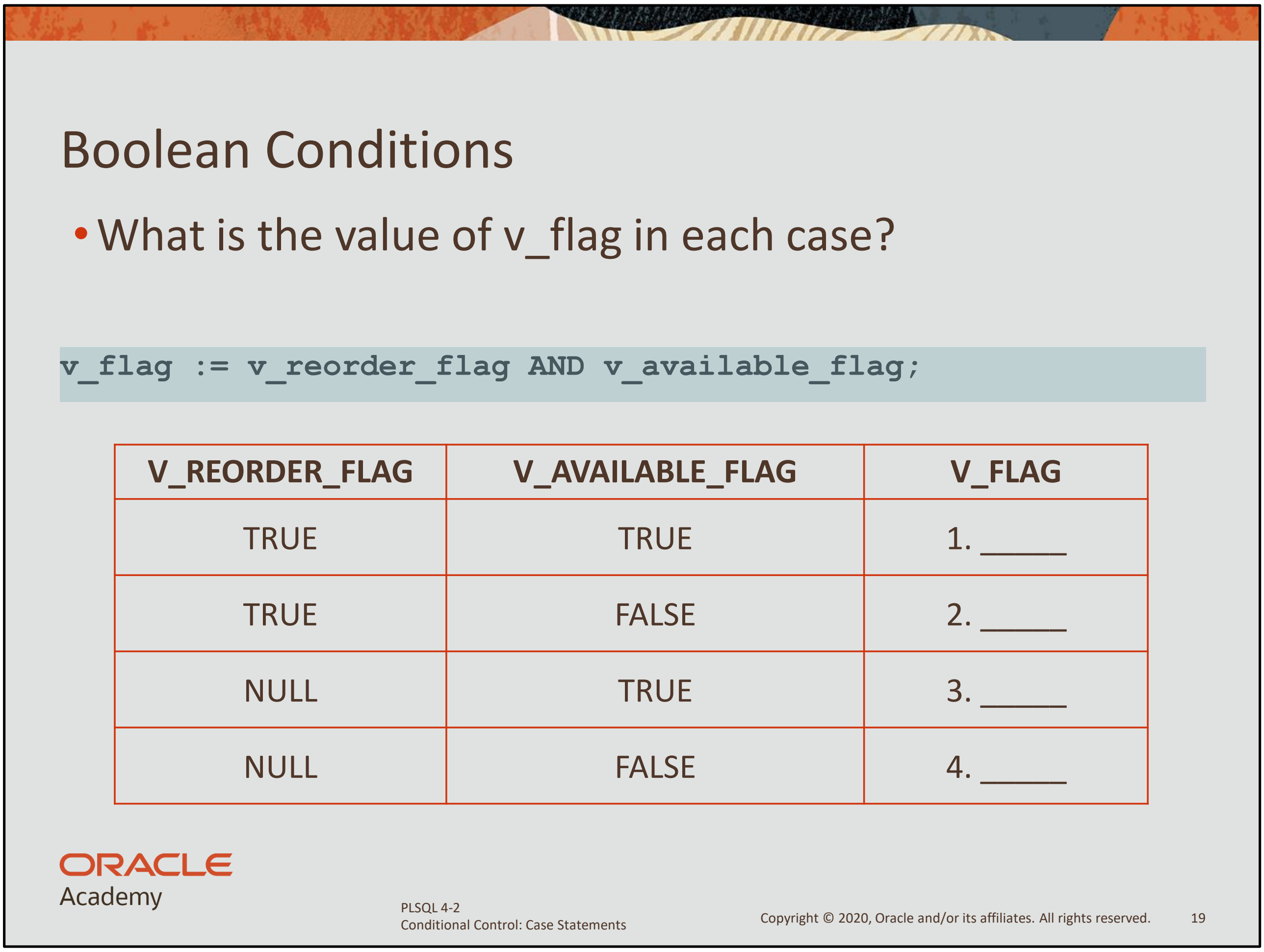
AND	TRUE	FALSE	NULL	OR	TRUE	FALSE	NULL	NOT	
TRUE	TRUE	Ex. FALSE	NULL	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	NULL	FALSE	TRUE
NULL	NULL	FALSE	NULL	NULL	TRUE	NULL	NULL	NULL	NULL

ORACLE
Academy

You can build a simple Boolean condition by combining number, character, or date expressions with comparison operators.

You can build a complex Boolean condition by combining simple Boolean conditions with the logical operators AND, OR, and NOT. The logical operators are used to check the Boolean variable values and return TRUE, FALSE, or NULL.

Note: The negation of NULL (NOT NULL) results in a null value because null values are indeterminate.



Boolean Conditions

- What is the value of v_flag in each case?

```
v_flag := v_reorder_flag AND v_available_flag;
```

V_REORDER_FLAG	V_AVAILABLE_FLAG	V_FLAG
TRUE	TRUE	1. _____
TRUE	FALSE	2. _____
NULL	TRUE	3. _____
NULL	FALSE	4. _____

The AND Logic Table on the previous slide can help you evaluate the possibilities for the Boolean condition in this slide.

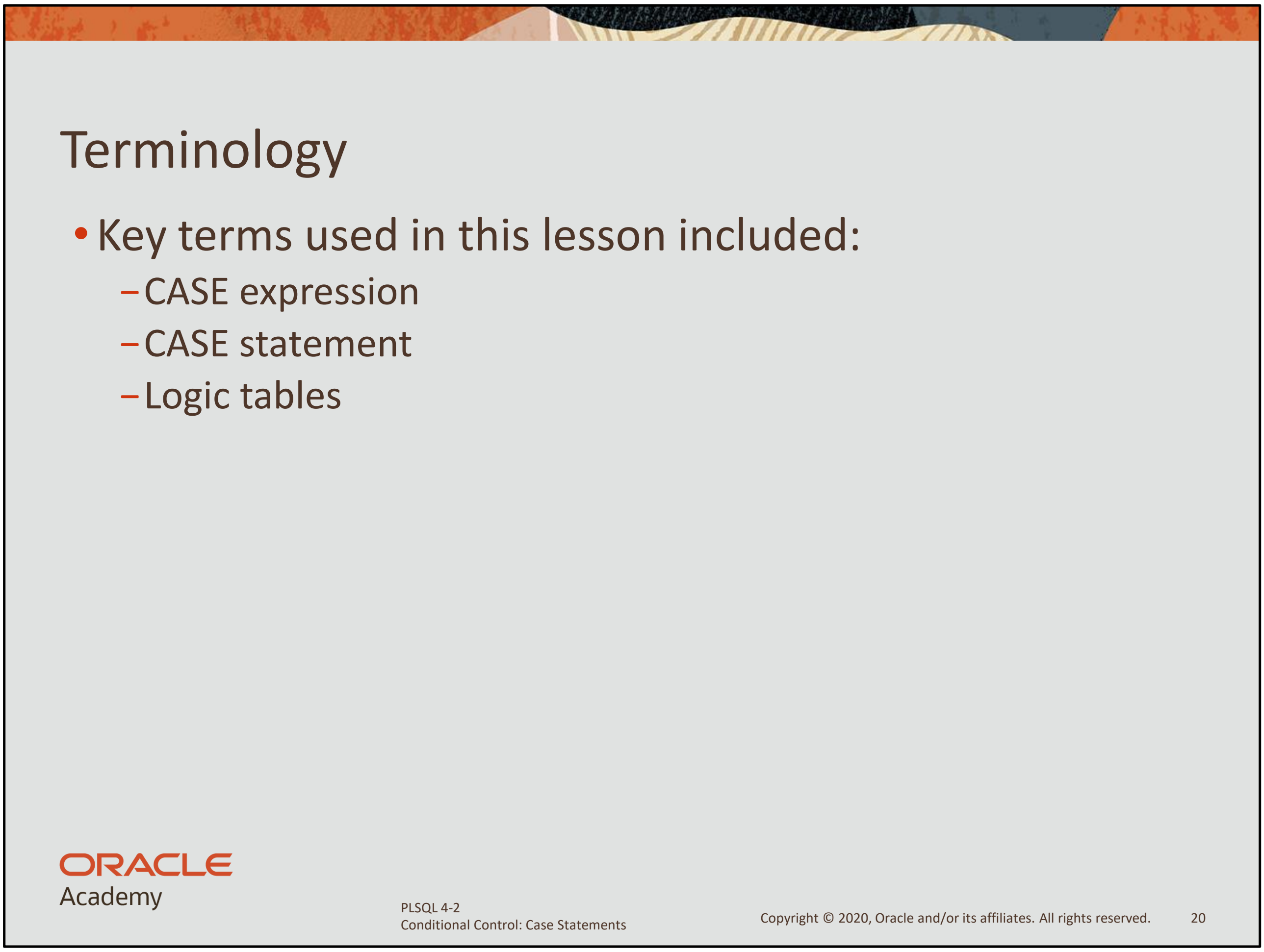
Answers:

1. TRUE

2. FALSE

3. NULL

4. FALSE

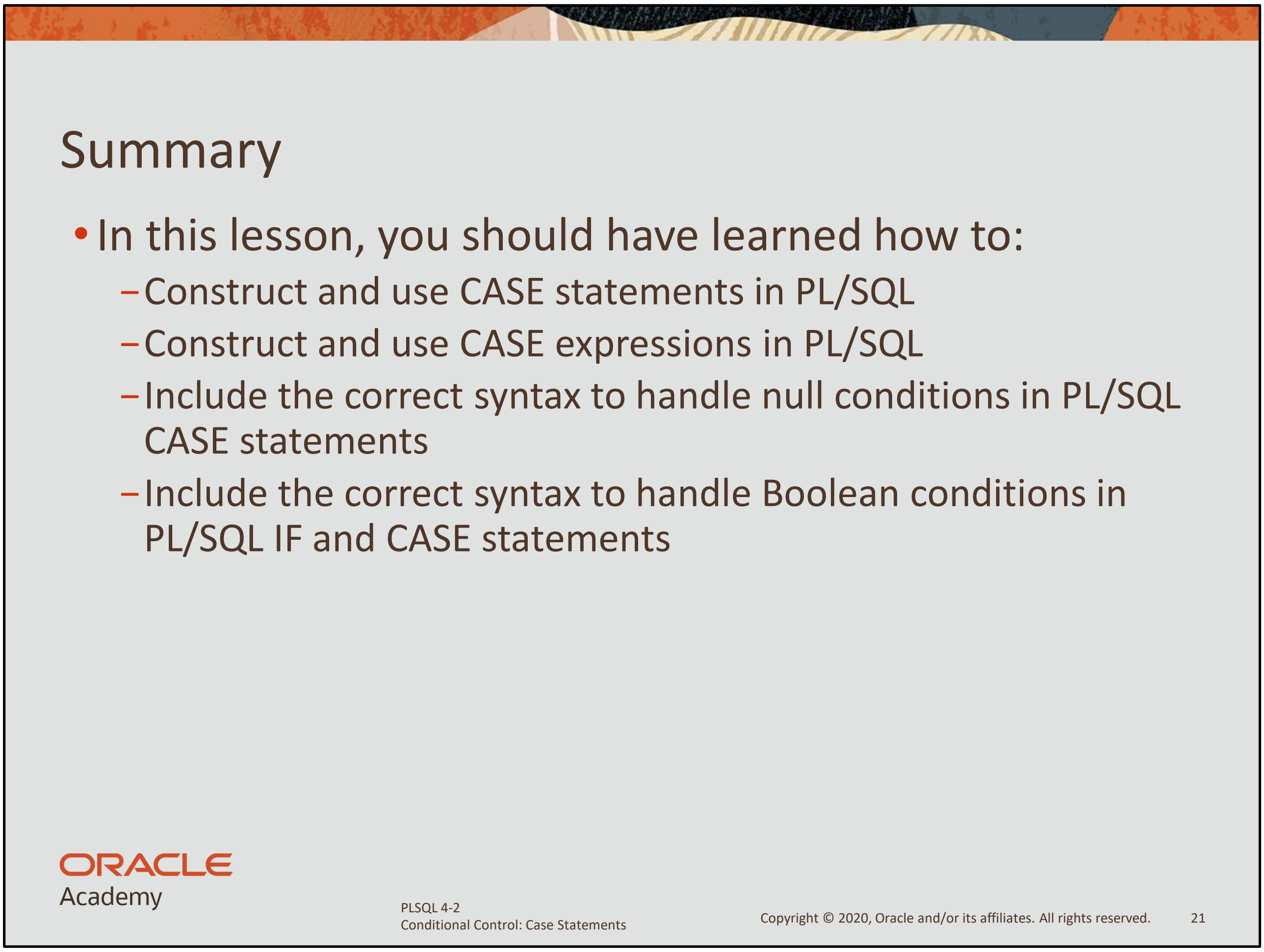


Terminology

- Key terms used in this lesson included:
  - CASE expression
  - CASE statement
  - Logic tables

ORACLE
Academy

- CASE expression – An expression that selects a result and returns it into a variable.
- CASE statement – A block of code that performs actions based on conditional tests.
- Logic Tables – Shows the results of all possible combinations of two conditions.



Summary

- In this lesson, you should have learned how to:
  - Construct and use CASE statements in PL/SQL
  - Construct and use CASE expressions in PL/SQL
  - Include the correct syntax to handle null conditions in PL/SQL CASE statements
  - Include the correct syntax to handle Boolean conditions in PL/SQL IF and CASE statements

ORACLE
Academy