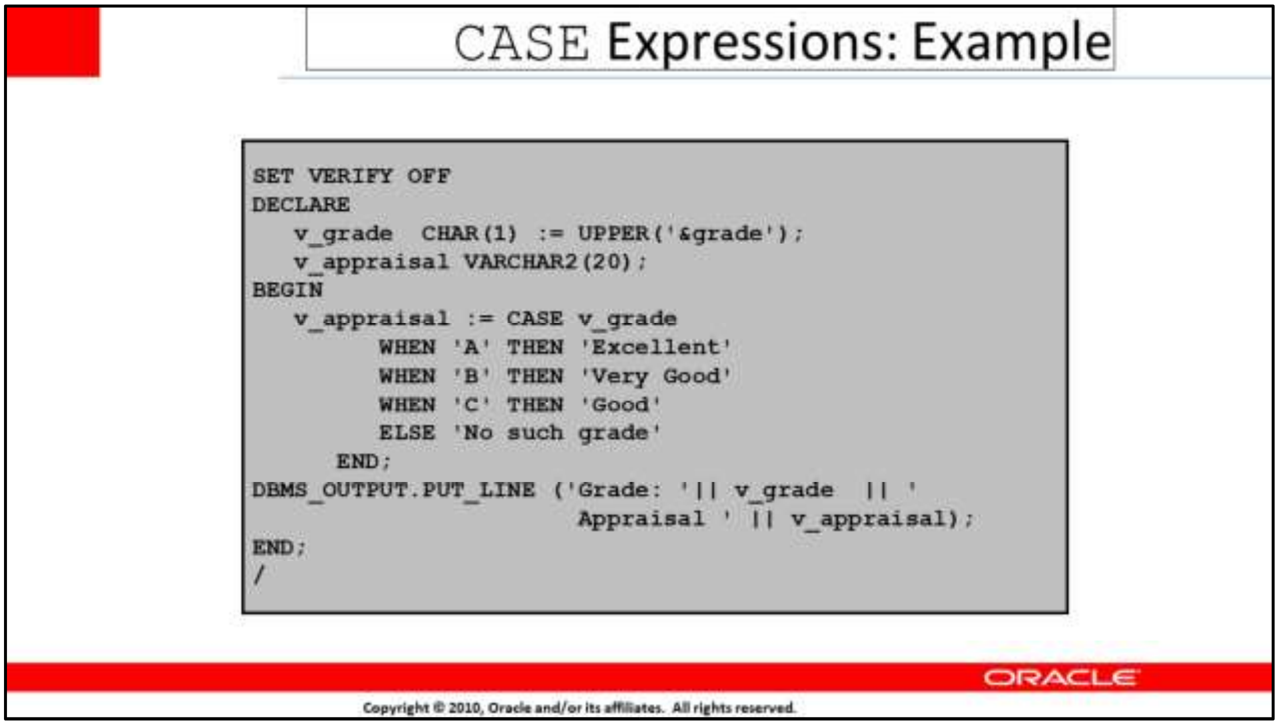


CASE Expressions: Example

```
SET VERIFY OFF
DECLARE
   v_grade  CHAR(1) := UPPER('&grade');
   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;
/
```

CASE Expressions

A CASE expression returns a result based on one or more alternatives. To return the result, the CASE expression uses a selector, which is an expression whose value is used to return one of several alternatives. The selector is followed by one or more WHEN clauses that are checked sequentially. The value of the selector determines which result is returned. If the value of the selector equals the value of a WHEN clause expression, that WHEN clause is executed and that result is returned.

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

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

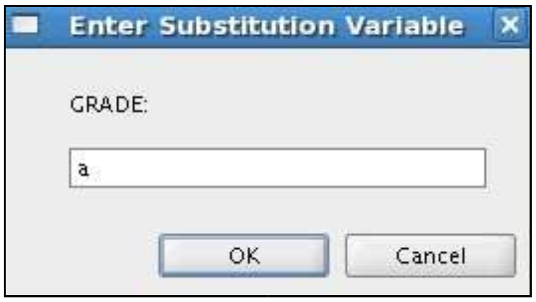
A searched CASE expression has no selector. Furthermore, the WHEN clauses in CASE expressions contain search conditions that yield a Boolean value rather than expressions that can yield a value of any type.

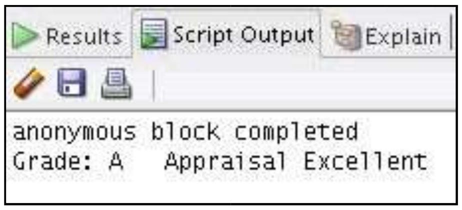
CASE Expressions: Example

In the example in the slide, the CASE expression uses the value in the v_grade variable as the expression. This value is accepted from the user by using a substitution variable. Based on the value entered by the user, the CASE expression returns the value of the v_appraisal variable based on the value of the v_grade value.

Result

When you enter a or A for v_grade, as shown in the Substitution Variable window, the output of the example is as follows:

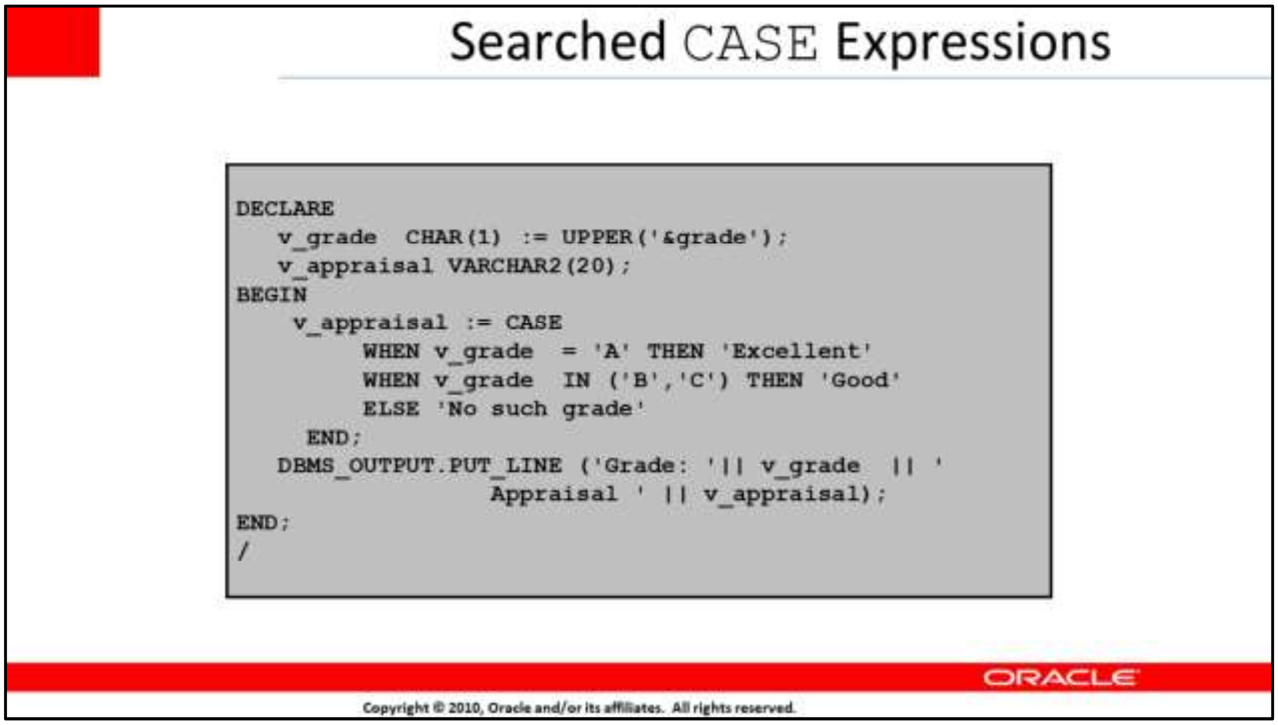




```
anonymous block completed
Grade: A   Appraisal Excellent
```

Searched CASE Expressions

In the previous example, you saw a single test expression, the v_grade variable. The WHEN clause compared a value against this test expression.



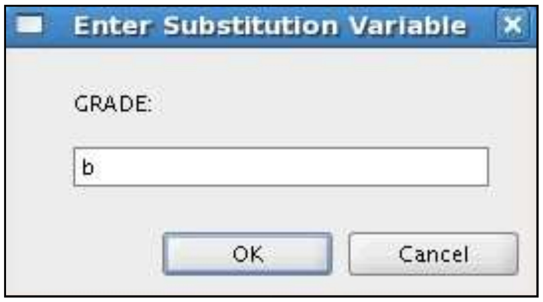
Searched CASE Expressions

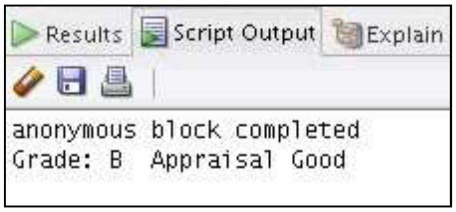
```
DECLARE
   v_grade  CHAR(1) := UPPER('&grade');
   v_appraisal VARCHAR2(20);
BEGIN
    v_appraisal := CASE
         WHEN v_grade  = 'A' THEN 'Excellent'
         WHEN v_grade  IN ('B','C') THEN 'Good'
         ELSE 'No such grade'
     END;
   DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade  || '
    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

In searched CASE statements, you do not have a test expression. Instead, the WHEN clause contains an expression that results in a Boolean value. The same example is rewritten in this slide to show searched CASE statements.

Result

The output of the example is as follows when you enter b or B for v_grade:



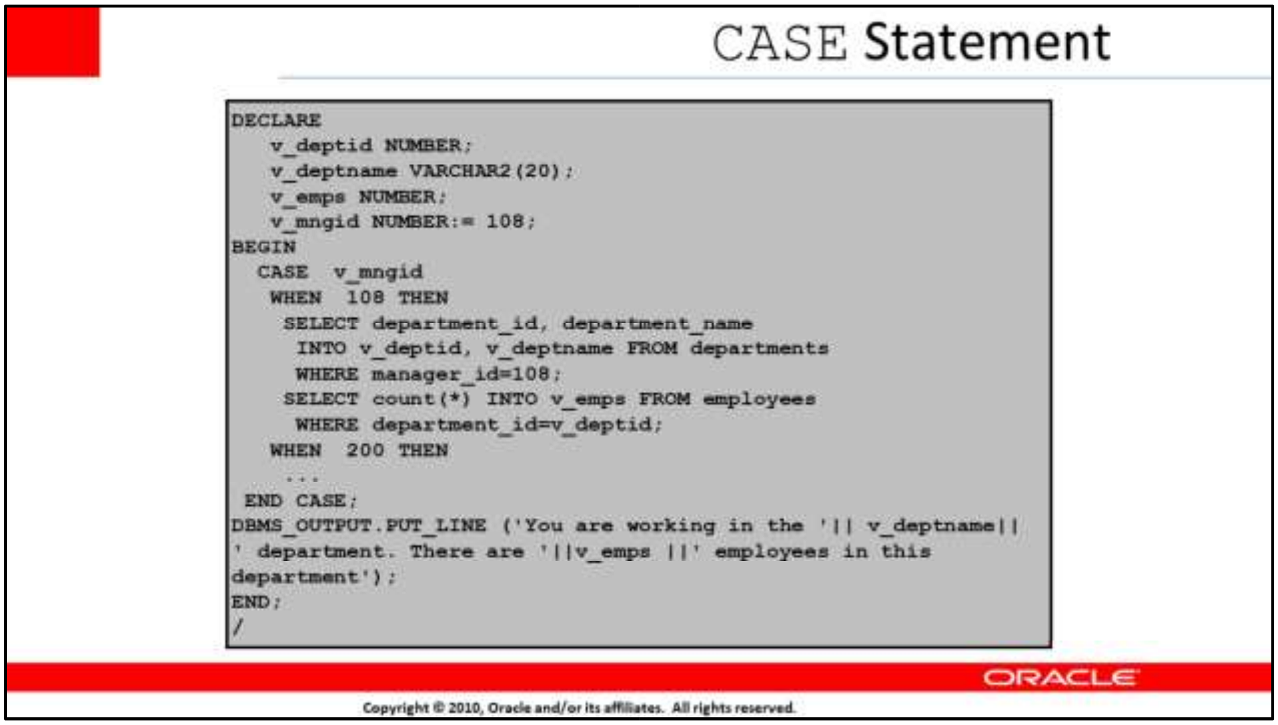


```
anonymous block completed
Grade: B  Appraisal Good
```

CASE Statement

Recall the use of the IF statement. You may include n number of PL/SQL statements in the THEN clause and also in the ELSE clause. Similarly, you can include statements in the CASE statement, which is more readable compared to multiple IF and ELSIF statements.

How a CASE Expression Differs from a CASE Statement



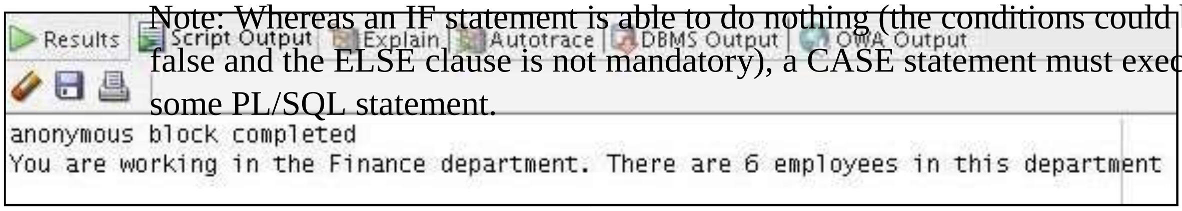
CASE Statement

```
DECLARE
   v_deptid NUMBER;
   v_deptname VARCHAR2(20);
   v_emps NUMBER;
   v_mngid NUMBER:= 108;
BEGIN
  CASE  v_mngid
   WHEN  108 THEN
    SELECT department_id, department_name
     INTO v_deptid, v_deptname FROM departments
     WHERE manager_id=108;
    SELECT count(*) INTO v_emps FROM employees
     WHERE department_id=v_deptid;
   WHEN  200 THEN
    ...
 END CASE;
DBMS_OUTPUT.PUT_LINE ('You are working in the '|| v_deptname||
' department. There are '||v_emps ||' employees in this
department');
END;
/
```

A CASE expression evaluates the condition and returns a value, whereas a CASE statement evaluates the condition and performs an action. A CASE statement can be a complete PL/SQL block. CASE statements end with ENDCASE;

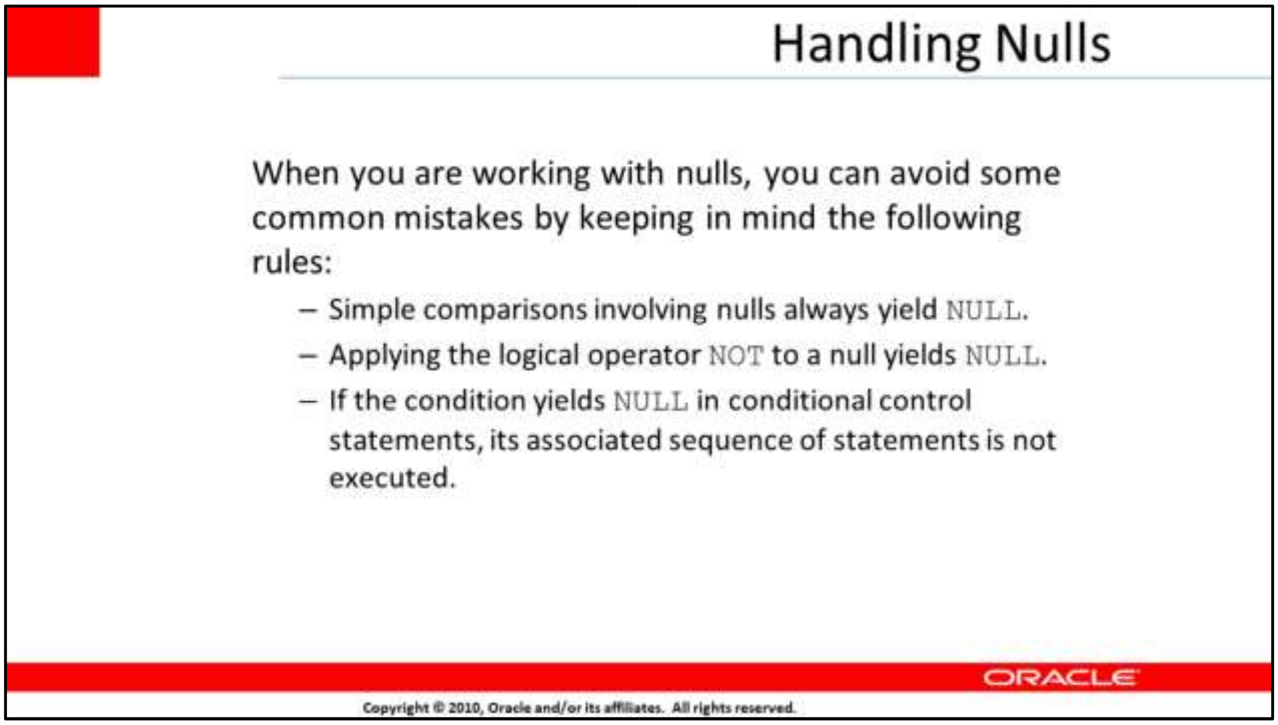
CASE expressions end with END;

The output of the slide code example is as follows:



```
anonymous block completed
You are working in the Finance department. There are 6 employees in this department
```

Note: Whereas an IF statement is able to do nothing (the conditions could be all false and the ELSE clause is not mandatory), a CASE statement must execute some PL/SQL statement.



Handling Nulls

When you are working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

- Simple comparisons involving nulls always yield NULL.
- Applying the logical operator NOT to a null yields NULL.
- If the condition yields NULL in conditional control statements, its associated sequence of statements is not executed.

ORACLE

Handling Nulls

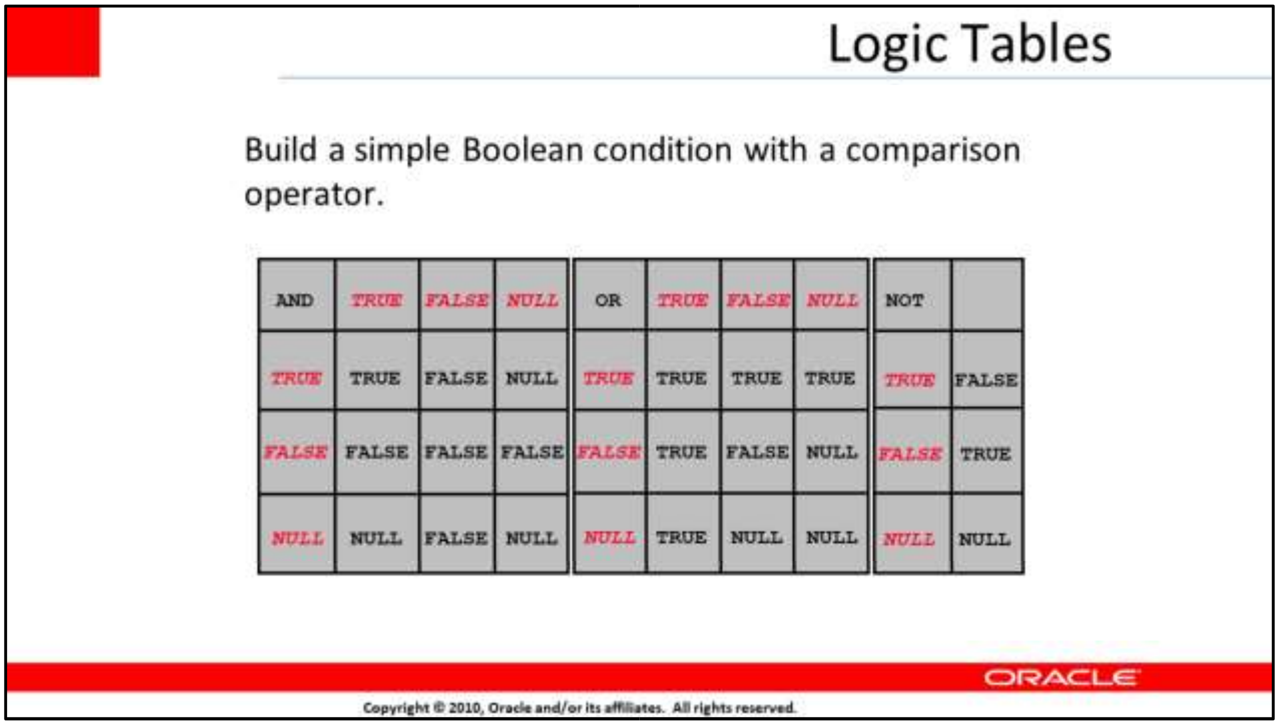
Consider the following example:

```
x := 5; y
:= NULL;
...
IF x != y THEN  -- yields NULL, not TRUE
   -- sequence_of_statements that are not executed END
IF;
```

You may expect the sequence of statements to execute because x and y seem unequal. But nulls are indeterminate. Whether or not x is equal to y is unknown. Therefore, the IF condition yields NULL and the sequence of statements is bypassed. a := NULL; b := NULL;

```
...
IF a = b THEN  -- yields NULL, not TRUE
   -- sequence_of_statements that are not executed END
IF;
```

In the second example, you may expect the sequence of statements to execute because a and b seem equal. But, again, equality is unknown, so the IF condition yields NULL and the sequence of statements is bypassed.



Logic Tables

Build a simple Boolean condition with a comparison operator.

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

Logic Tables

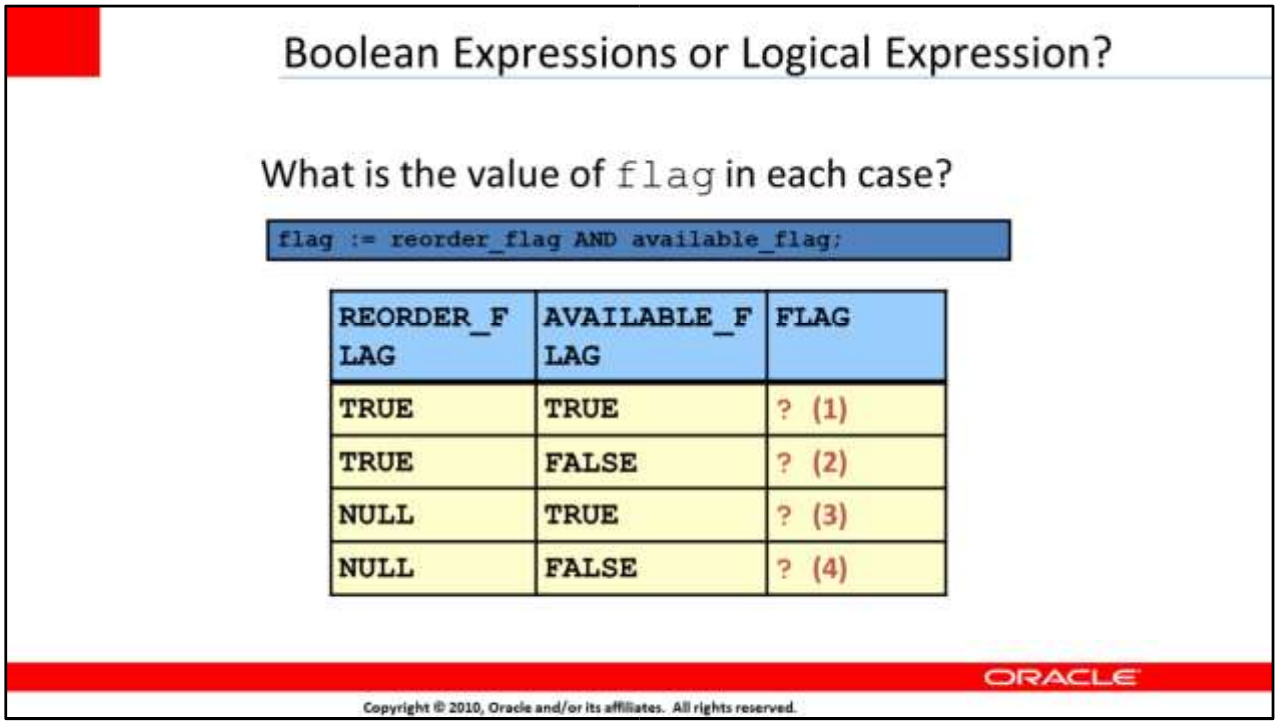
You can build a simple Boolean condition by combining number, character, and date expressions with comparison operators.

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.

In the logic tables shown in the slide:

- •FALSE takes precedence in an AND condition, and TRUE takes precedence in an OR condition
- •AND returns TRUE only if both of its operands are TRUE
- •OR returns FALSE only if both of its operands are FALSE
- •NULLANDTRUE always evaluates to NULL because it is not known whether the second operand evaluates to TRUE

Note: The negation of NULL (NOTNULL) results in a null value because null values are indeterminate.



Boolean Expressions or Logical Expression?

What is the value of `flag` in each case?

```
flag := reorder_flag AND available_flag;
```

REORDER_FLAG	AVAILABLE_FLAG	FLAG
TRUE	TRUE	? (1)
TRUE	FALSE	? (2)
NULL	TRUE	? (3)
NULL	FALSE	? (4)

Boolean Expressions or Logical Expression?

The AND logic table can help you to evaluate the possibilities for the Boolean condition in the slide.

Answers

1.	TRUE
2.	FALSE
3.	NULL
4.	FALSE