Exercises for Chapter 4: Conditional Control: `IF` Statements

The Labs below provide you with exercises and suggested answers with discussion related to how those answers resulted. The most important thing to realize is whether your answer works. You should figure out the implications of the answers here and what the effects are from any different answers you may come up with.

Lab 4.1 `IF` Statements

Answer the following questions:

`IF-THEN` Statements

In this exercise, you will use the `IF-THEN` statement to test whether the date provided by the user falls on the weekend, in other words, if the day happens to be Saturday or Sunday.

Create the following PL/SQL script:

For Example *ch04_6a.sql*

```
DECLARE
   v_date DATE := TO_DATE('&sv_user_date', 'DD-MON-YYYY');
   v_day  VARCHAR2(15);
BEGIN
   v_day := RTRIM(TO_CHAR(v_date, 'DAY'));

   IF v_day IN ('SATURDAY', 'SUNDAY')
   THEN
      DBMS_OUTPUT.PUT_LINE (v_date||' falls on weekend');
   END IF;
   DBMS_OUTPUT.PUT_LINE ('After IF statement…');
END;
```

In order to test this script fully, execute it twice. For the first run, enter ‘13-SEP-2014’, and for the second run, enter ‘16-SEP-2014’. Execute the script, and then answer the following questions:

a) What output was generated by the script (for both dates)?

Answer: The first output produced is for the date of 13-SEP-2014:

```
09/13/2014 00:00 falls on weekend
After IF statement…
```

The second output produced for the date is 16-SEP-2014:

```
After IF statement…
```

b) Explain why the output produced for the two dates is different.

Answer: When the value of 13-SEP-2014 is entered for the variable `v_date`, the day of the week is determined with the help of the functions `TO_CHAR` and `RTRIM`, and then assigned to the variable `v_day`.

Next, the following condition is evaluated:

```
v_day IN ('SATURDAY', 'SUNDAY')
```

Because the value of the variable `v_day` is ‘SATURDAY,’ the condition evaluates to TRUE, and the `DBMS_OUTPUT.PUT_LINE` statement in the body of the `IF-THEN` statement is executed. As a result, ‘09/13/2014 00:00 falls on weekend’ is displayed in the Dbms Output window. Next, the control of the execution is passed to the last `DBMS_OUTPUT.PUT_LINE` statement and ‘After IF statement…’ is displayed in the Dbms Output window.

For the second run, the variable `v_date` is assigned value of 16-SEP-2014. And just like in the previous run, the value of the variable `v_day` is derived from the value of the variable `v_date`. Next, the condition of the `IF-THEN` statement is evaluated. Because it evaluates to FALSE, the `IF-THEN` statement does not execute, and the control of the execution is passed to the last `DBMS_OUTPUT.PUT_LINE` statement, and ‘After `IF` statement…’ is displayed in the Dbms Output window.

c) Remove the RTRIM function from the assignment statement for the variable `v_day` as follows

```
v_day := TO_CHAR(v_date, 'DAY');
```

and run the script again, for ‘14-SEP-2014’ date. What output was generated in this case? Why?

Answer: The script should look similar to the following. Changes are highlighted in bold.

For Example *ch04_6b.sql*

```
DECLARE
   v_date DATE := TO_DATE('&sv_user_date', 'DD-MON-YYYY');
   v_day  VARCHAR2(15);
BEGIN
   v_day := TO_CHAR(v_date, 'DAY');

   IF v_day IN ('SATURDAY', 'SUNDAY')
   THEN
      DBMS_OUTPUT.PUT_LINE (v_date||' falls on weekend');
   END IF;
   DBMS_OUTPUT.PUT_LINE ('After IF statement…');
END;
```

When the value of 14-SEP-2014 is entered for the variable `v_date`, the new version of the script produces output as shown:

```
After IF statement…
```

In the original example, the variable `v_day` is calculated with the help of the statement,

```
RTRIM(TO_CHAR(v_date, 'DAY'))
```

First, the function `TO_CHAR` returns the day of the week padded with blanks. In this case, the size of the value retrieved by the function `TO_CHAR` is always 9 bytes. Next, the `RTRIM` function removes trailing spaces.

In the statement,

```
v_day := TO_CHAR(v_date, 'DAY')
```

the `TO_CHAR` function is used without the `RTRIM` function. Therefore, trailing blanks are not removed after the day of the week has been derived. As a result, the condition of the `IF-THEN` statement evaluates to FALSE even though given date falls on the weekend. As a result, the control of the execution is passed to the last `DBMS_ OUTPUT.PUT_LINE` statement.

d) Rewrite this script using the `LIKE` operator instead of the `IN` operator, so that it produces the same results for the dates specified earlier.

Answer: The script should look similar to the following. Newly modified statements are shown in bold.

For Example *ch04_6c.sql*

```
DECLARE
   v_date DATE := TO_DATE('&sv_user_date', 'DD-MON-YYYY');
   v_day  VARCHAR2(15);
BEGIN
   v_day := TO_CHAR(v_date, 'DAY');

   IF v_day LIKE 'S%'
   THEN
      DBMS_OUTPUT.PUT_LINE (v_date||' falls on weekend');
   END IF;
   DBMS_OUTPUT.PUT_LINE ('After IF statement…');
END;
```

Since both days, Saturday and Sunday, are the only days of the week that start with the letter ‘S’, there is no need to spell out the names of the days or specify any additional characters for the `LIKE` operator.

When this version of the script is executed for the original dates of 13-SEP-2014 and 16-SEP-2014 it produces the same output as the original example:

```
09/13/2014 00:00 falls on weekend
After IF statement…
```

And

```
After IF statement…
```

`IF-THEN-ELSE` Statement

In this exercise, you will use the `IF-THEN-ELSE` statement to check how many students are enrolled in course number 25, section 1. If there are 15 or more students enrolled, section 1 of course number 25 is full. Otherwise, section 1 of course number 25 is not full and more students can register for it. In both cases, a message should be displayed to the user indicating whether section 1 is full. Try to answer the questions before you run the script. Once you have answered the questions, run the script and check your answers.

Create the following PL/SQL script:

For Example *ch04_7a.sql*

```
DECLARE
   v_total NUMBER;
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment e
     JOIN section s USING (section_id)
    WHERE s.course_no = 25
      AND s.section_no = 1;

   -- check if section 1 of course 25 is full
   IF v_total >= 15
   THEN
      DBMS_OUTPUT.PUT_LINE ('Section 1 of course 25 is full');
   ELSE
      DBMS_OUTPUT.PUT_LINE ('Section 1 of course 25 is not full');
   END IF;
   -- control resumes here
END;
```

Try to answer the following questions first and then execute the script:

a) What `DBMS_OUTPUT.PUT_LINE` statement is executed if there are 15 students enrolled in section 1 of course number 25?

   Answer: If there are 15 or more students enrolled in section 1 of course number 25, the first `DBMS_OUTPUT.PUT_LINE` statement is executed and message ‘Section 1 of course 25 is full’ is displayed in the Dbms Output window. This is because the condition

   ```
   v_total >= 15
   ```

   evaluates to `TRUE`, and as a result, the statement

   ```
   DBMS_OUTPUT.PUT_LINE ('Section 1 of course 25 is full');
   ```

   is executed.

b) What `DBMS_OUTPUT.PUT_LINE` statement is executed if there are 3 students enrolled in section 1 of course number 25?

Answer: If there are 3 students enrolled in section 1 of course number 25, the second `DBMS_OUTPUT.PUT_LINE` statement is executed and the message 'Section 1 of course 25 is not full' is displayed in the Dbms Output window. This is because the condition

```
v_total >= 15
```

evaluates to `FALSE`, and control of the execution is passed to the `ELSE` part on the `IF-THEN-ELSE` statement. As a result, the statement

```
DBMS_OUTPUT.PUT_LINE ('Section 1 of course 25 is not full');
```

is executed.

c) What `DBMS_OUTPUT.PUT_LINE` statement is executed if there is no section 1 for course number 25?

Answer: If there is no section 1 for course number 25, the second `DBMS_OUTPUT.PUT_LINE` statement is executed and the message 'Section 1 of course 25 is not full' is displayed in the Dbms Output window. This is because the `COUNT` function used in the `SELECT` statement

```
SELECT COUNT(*)
  INTO v_total
  FROM enrollment e
  JOIN section s USING (section_id)
 WHERE s.course_no = 25
   AND s.section_no = 1;
```

returns 0, and the condition of the `IF-THEN-ELSE` statement evaluates to `FALSE`. Therefore, the control of the execution is passed to the `ELSE` part of the `IF-THEN-ELSE` statement, and the second `DBMS_OUTPUT.PUT_LINE` statement is executed and its message is displayed in the Dbms Output window.

d) How would you change this script so that if there are less than 15 students enrolled in section 1 of course number 25, a message indicating how many students can still be enrolled is displayed?

Answer: The script should look similar to this script. Newly added statements are highlighted in bold.

For Example *ch04_7b.sql*

```
DECLARE
   v_total    NUMBER;
   v_students NUMBER;
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment e
     JOIN section s USING (section_id)
    WHERE s.course_no = 25
      AND s.section_no = 1;

   -- check if section 1 of course 25 is full
   IF v_total >= 15
   HEN
      DBMS_OUTPUT.PUT_LINE ('Section 1 of course 25 is full');
   ELSE
      v_students := 15 - v_total;
      DBMS_OUTPUT.PUT_LINE (v_students||
         ' students can still enroll into section 1 of course 25');
   END IF;
   -- control resumes here
END;
```

Notice that if the IF-THEN-ELSE statement evaluates to FALSE, the statements associated with the ELSE part are executed. In this case, the value of the variable v_total is subtracted from 15. The result of this operation indicates how many more students can enroll in section 1 of course number 25.

Lab 4.2 ELSIF Statements

In this exercise, you will use an ELSIF statement to display a letter grade for a student registered for a specific section of course number 25.

Create the following PL/SQL script:

For Example *ch04_8a.sql*

```
DECLARE
   v_student_id   NUMBER := 102;
   v_section_id   NUMBER := 89;
   v_final_grade  NUMBER;
   v_letter_grade CHAR(1);
BEGIN
   SELECT final_grade
     INTO v_final_grade
     FROM enrollment
    WHERE student_id = v_student_id
      AND section_id = v_section_id;

   IF v_final_grade BETWEEN 90 AND 100
   THEN
      v_letter_grade := 'A';
   ELSIF v_final_grade BETWEEN 80 AND 89
   THEN
      v_letter_grade := 'B';
   ELSIF v_final_grade BETWEEN 70 AND 79
   THEN
      v_letter_grade := 'C';
   ELSIF v_final_grade BETWEEN 60 AND 69
   THEN
      v_letter_grade := 'D';
   ELSE
      v_letter_grade := 'F';
   END IF;

   -- control resumes here
   DBMS_OUTPUT.PUT_LINE ('Letter grade is: '||v_letter_grade);
```

```
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such student or section');
END;
```

Note, that you may need to change the values for the variables `v_student_id` and `v_section_id` as you see fit in order to test some of your answers.

Try to answer the following questions first, and then execute the script:

a) What letter grade will be displayed in the Dbms Output window:

   i) if the value of `v_final_grade` is equal to 85?
   ii) if the value of `v_final_grade` is NULL?
   iii) if the value of `v_final_grade` is greater than 100?

Answers:

i) If the value of `v_final_grade` is equal to 85, the value "B" of the letter grade will be displayed in the Dbms Output window.

The conditions of the `ELSIF` statement are evaluated in sequential order. The first condition

```
v_final_grade BETWEEN 90 AND 100
```

evaluates to FALSE, and control is passed to the first `ELSIF` part of the `ELSIF` statement. Then, the second condition

```
v_final_grade BETWEEN 80 AND 89
```

evaluates to `TRUE`, and the letter "B" is assigned to the variable `v_letter_grade`. The control of the execution is then passed to first executable statement after `END IF` statement, and the message 'Letter grade is: B' is displayed in the Dbms Output window.

ii) If the value of `v_final_grade` is `NULL`, value "F" of the letter grade will be displayed of the screen.

If the value of the variable `v_final_grade` is undefined or `NULL`, then all conditions of the `ELSIF` statement evaluate to `NULL` (notice, they do not evaluate to `FALSE`). As a result, the `ELSE` part of the `ELSIF` statement is executed, and letter "F" is assigned to the variable `v_letter_grade`. Thus, the message 'Letter grade is: F' is displayed in the Dbms Output window.

iii) If the value of `v_final_grade` is greater than 100, value "F" of the letter grade will be displayed of the screen.

The conditions specified for the `ELSIF` statement cannot handle a value of the variable `v_final_grade` greater than 100. So, for any student whose letter grade should be A+, will result in a letter grade of "F." After the `ELSIF` statement has terminated, message 'The letter grade is: F' is displayed in the Dbms Output window.

b) How would you change this script so that a message 'v_final_grade is null' is displayed in the Dbms Output window if `v_final_grade` is `NULL`?

Answer: The script should look similar to this script. Changes are shown in bold.

For Example *ch04_8b.sql*

```
DECLARE
   v_student_id   NUMBER := 102;
   v_section_id   NUMBER := 89;
   v_final_grade  NUMBER;
   v_letter_grade CHAR(1);
BEGIN
   SELECT final_grade
     INTO v_final_grade
     FROM enrollment
    WHERE student_id = v_student_id
      AND section_id = v_section_id;

   IF v_final_grade IS NULL
   THEN
      DBMS_OUTPUT.PUT_LINE ('v_final_grade is null');
   ELSIF v_final_grade BETWEEN 90 AND 100
   THEN
      v_letter_grade := 'A';
   ELSIF v_final_grade BETWEEN 80 AND 89
   THEN
      v_letter_grade := 'B';
   ELSIF v_final_grade BETWEEN 70 AND 79
   THEN
      v_letter_grade := 'C';
   ELSIF v_final_grade BETWEEN 60 AND 69
   THEN
      v_letter_grade := 'D';
   ELSE
      v_letter_grade := 'F';
   END IF;

   -- control resumes here
   DBMS_OUTPUT.PUT_LINE ('Letter grade is: '||v_letter_grade);

EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such student or section');
END;
```

Note that one more condition has been added to the `ELSIF` statement. In this case, the condition

```
v_final_grade BETWEEN 90 AND 100
```

becomes the first `ELSIF` condition. Now, if the value of `v_final_grade` is `NULL`, the message 'v_final_grade is null' is displayed in the Dbms Output window. However, because there is no value assigned to the variable `v_letter_grade`, the message 'Letter grade is:' is displayed as well.

c) How would you change the script to define a letter grade without specifying the upper limit of the final grade? In the statement, `v_final_grade BETWEEN 90 and 100`, number 100 is the upper limit.

Answer: The script should look similar to following. Changes are shown in bold.

For Example *ch04_8c.sql*

```
DECLARE
   v_student_id   NUMBER := 102;
   v_section_id   NUMBER := 89;
   v_final_grade  NUMBER;
   v_letter_grade CHAR(1);
BEGIN
   SELECT final_grade
     INTO v_final_grade
     FROM enrollment
    WHERE student_id = v_student_id
      AND section_id = v_section_id;

   IF v_final_grade >= 90
   THEN
      v_letter_grade := 'A';
   ELSIF v_final_grade >= 80
   THEN
      v_letter_grade := 'B';
   ELSIF v_final_grade >= 70
   THEN
      v_letter_grade := 'C';
   ELSIF v_final_grade >= 60
   THEN
      v_letter_grade := 'D';
   ELSE
      v_letter_grade := 'F';
   END IF;

   -- control resumes here
   DBMS_OUTPUT.PUT_LINE ('Letter grade is: '||v_letter_grade);

EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no such student or section');
END;
```

In this example, there is no upper limit specified for the variable `v_final_grade` because the `BETWEEN` operator has been replaced with the >= operator. Thus, this script is able to handle a value of the variable `v_final_grade` that is greater than 100. Instead of assigning letter ‘F’ to the variable `v_letter_grade` (in version ch01_8a.sql of the script), the letter ‘A’ is assigned to the variable `v_letter_grade`. As a result, this script produces more accurate results.

Lab 4.3 Nested `IF` Statements

In this exercise, you will use nested IF statements. This script will convert the value of a temperature from one system to another. If the temperature is supplied in Fahrenheit, it will be converted to Celsius, and vice versa.

Create the following PL/SQL script:

For Example *ch04_9a.sql*

```
DECLARE
   v_temp_in   NUMBER := &sv_temp_in;
   v_scale_in  CHAR   := '&sv_scale_in';
   v_temp_out  NUMBER;
   v_scale_out CHAR;
BEGIN
   IF v_scale_in != 'C' AND v_scale_in != 'F'
   THEN
      DBMS_OUTPUT.PUT_LINE ('This is not a valid scale');
   ELSE
      IF v_scale_in = 'C'
      THEN
         v_temp_out  := ( (9 * v_temp_in) / 5 ) + 32;
         v_scale_out := 'F';
      ELSE
         v_temp_out  := ( (v_temp_in - 32) * 5 ) / 9;
         v_scale_out := 'C';
      END IF;
      DBMS_OUTPUT.PUT_LINE ('New scale is: '||v_scale_out);
      DBMS_OUTPUT.PUT_LINE ('New temperature is: '||v_temp_out);
   END IF;
END;
```

Execute the script, and then answer the following questions:

a) What output is generated by the script if the value of 100 is entered for the temperature, and the letter 'C' is entered for the scale?

Answer: The output should look like the following:

```
New scale is: F
New temperature is: 212
```

Once the values for the variables v_temp_in and v_scale_in have been provided, the condition

```
v_scale_in != 'C' AND v_scale_in != 'F'
```

of the outer IF statement evaluates to FALSE, and control of the execution is passed to the ELSE part of the outer IF statement. Next, the condition

```
v_scale_in = 'C'
```

of the inner IF statement evaluates to TRUE, and the values of the variables v_temp_out and v_scale_out are calculated. Control of the execution is then passed back to the outer IF statement, and the new value for the temperature and the scale are displayed in the Dbms Output window.

b) Run this script without providing a value for the temperature. What output will be generated by the script in this case? Why?

Answer: If the value for the temperature is not entered, the script will not compile at all. The compiler will try to assign a value to the variable v_temp_in with the help of the substitution variable. Because the value for v_temp_in has not been entered, the assignment statement will fail, and the following error message will be displayed.

```
ORA-06550: line 2, column 26:
PLS-00103: Encountered the symbol ";" when expecting one of
the following:
   ( - + case mod new not null <an identifier>
   <a double-quoted delimited-identifier> <a bind variable>
   continue avg count current exists max min prior sql stddev
   sum variance execute forall merge time timestamp interval
   date <a string literal with character set specification>
   <a number> <a single-quoted SQL string> pipe
   <an alternatively-quoted string literal with character set
   specification>
```

You have probably noticed that even though the mistake seems small and insignificant, the error message is fairly long and confusing.

c) Try to run this script providing an invalid letter for the temperature scale, for example, letter 'V'. What message will be displayed in the Dbms Output window? Why?

Answer: If an invalid letter is entered for the scale, the message 'This is not a valid scale' will be displayed in the Dbms Output window.

The condition of the outer IF statement will be evaluated to TRUE. As a result, the inner IF statement will not be executed at all, and the message 'This is not a valid scale' will be displayed in the Dbms Output window. This is illustrated further by the output below:

```
This is not a valid scale
```

d) Rewrite this script so that if an invalid letter is entered for the scale, the variable v_temp_out is initialized to 0 and the variable v_scale_out is initialized to C.

Answer: The script should look similar to the following script. Newly added and modified statements are shown in bold. Notice that the last two DBMS_OUTPUT.PUT_LINE statements have been moved from the body of the outer IF statement.

For Example *ch04_9b.sql*

```
DECLARE
   v_temp_in   NUMBER := &sv_temp_in;
   v_scale_in  CHAR   := '&sv_scale_in';
   v_temp_out  NUMBER;
   v_scale_out CHAR;
BEGIN
   IF v_scale_in != 'C' AND v_scale_in != 'F'
   THEN
      DBMS_OUTPUT.PUT_LINE ('This is not a valid scale');
      v_temp_out  := 0;
      v_scale_out := 'C';
   ELSE
      IF v_scale_in = 'C'
      THEN
```

```
         v_temp_out  := ( (9 * v_temp_in) / 5 ) + 32;
         v_scale_out := 'F';
      ELSE
         v_temp_out  := ( (v_temp_in - 32) * 5 ) / 9;
         v_scale_out := 'C';
      END IF;
   END IF;
   DBMS_OUTPUT.PUT_LINE ('New scale is: '||v_scale_out);
   DBMS_OUTPUT.PUT_LINE ('New temperature is: '||v_temp_out);
END;
```

This version of the script produces output as follows:

```
This is not a valid scale.
New scale is: C
New temperature is: 0
```

Try It Yourself

The projects in this section are meant to have you use all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

1) Create the following script. For a given instructor, determine how many sections he or she is teaching. If the number is greater than or equal to 3, display a message saying that the instructor needs a vacation. Otherwise, display a message saying how many sections this instructor is teaching.

   Answer: The script should look similar to the following:

   For Example *ch04_10a.sql*

```
DECLARE
   v_instructor_id NUMBER := &sv_instructor_id;
   v_total NUMBER;
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM section
    WHERE instructor_id = v_instructor_id;

   -- check if instructor teaches 3 or more sections
   IF v_total >= 3
   THEN
      DBMS_OUTPUT.PUT_LINE ('This instructor needs a vacation');
   ELSE
      DBMS_OUTPUT.PUT_LINE ('This instructor teaches '||v_total||' sections');
   END IF;
   -- control resumes here
END;
```

The script above declares two variables, `v_instructor_id` and `v_total`. Both variables are used by the `SELECT INTO` statement to check the number of sections taught by an instructor. Next, the `IF-THEN-ELSE` statement evaluates whether an instructor teaches 3 or more sections. If a particular instructor teaches three or more sections, the condition of the `IF-THEN-ELSE` statement evaluates to `TRUE`, and the message ‘This instructor needs a vacation’ is displayed in the Dbms Output window. In the opposite case, the message stating how many sections instructor is teaching is displayed.

Assume that value 101 is provided for instructor ID at the runtime. Then the script produces output as follows:

```
This instructor needs a vacation
```

2) Execute the two PL/SQL blocks below and explain why they produce different output for the same value of the variable `v_num`.

For Example *ch04_11a.sql*

```
-- Block 1
DECLARE
   v_num NUMBER := NULL;
BEGIN
   DBMS_OUTPUT.PUT_LINE ('PL/SQL Block 1');
   IF v_num > 0
   THEN
      DBMS_OUTPUT.PUT_LINE ('v_num is greater than 0');
   ELSE
      DBMS_OUTPUT.PUT_LINE ('v_num is not greater than 0');
   END IF;
END;
/

-- Block 2
DECLARE
   v_num NUMBER := NULL;
BEGIN
   DBMS_OUTPUT.PUT_LINE ('PL/SQL Block 2');
   IF v_num > 0
   THEN
      DBMS_OUTPUT.PUT_LINE ('v_num is greater than 0');
   END IF;
   IF NOT (v_num > 0)
   THEN
      DBMS_OUTPUT.PUT_LINE ('v_num is not greater than 0');
   END IF;
END;
/
```

Answer: Consider outputs produced by the preceding scripts:

```
PL/SQL Block 1
v_num is not greater than 0

PL/SQL Block 2
```

The outputs produced by Block 1 and Block 2 are different, even though in both examples the variable `v_num` has been defined as `NULL`.

First, take a closer look at the IF-THEN-ELSE statement used in Block 1:

```
IF v_num > 0
THEN
   DBMS_OUTPUT.PUT_LINE ('v_num is greater than 0');
ELSE
   DBMS_OUTPUT.PUT_LINE ('v_num is not greater than 0');
END IF;
```

The condition v_num > 0 evaluates to FALSE because NULL has been assigned to the variable v_num. As a result, the control of the execution is transferred to the ELSE part of the IF-THEN-ELSE statement. So, the message 'v_num is not greater than 0' is displayed in the Dbms Output window.

Second, take a closer look at the IF-THEN statements used in Block 2:

```
IF v_num > 0
THEN
   DBMS_OUTPUT.PUT_LINE ('v_num is greater than 0');
END IF;
IF NOT (v_num > 0)
THEN
   DBMS_OUTPUT.PUT_LINE ('v_num is not greater than 0');
END IF;
```

For both IF-THEN statements their conditions evaluate to FALSE, and as a result none of the messages are in the Dbms Output window.