Experiment No 18: Implement PL/SQL program using Conditional Statements

I. **Practical Significance:** Conditional statements in PL/SQL are like decision-makers in PL/SQL code. They help to make choices based on different conditions. For example, if something is true, do one thing, if not, do something else. This flexibility makes PL/SQL code more powerful and capable of handling different situations. This practical allows students to write a PL/SQL program using Conditional Statements- if, if then else, nested if, if elsif else.

II. **INDUSTRY / EMPLOYER EXPECTED OUTCOME:**

To implement conditional statements in PL/SQL to make better decisions and to improve problem-solving skills.

III. **COURSE LEVEL LEARNING OUTCOMES (COS):**CO4

- impiement PL/SQL codes for given appiication.

IV. **LABORATORY LEARNING OUTCOME:**

Implement PL/SQL program using Conditional Statements.

V. **Relevant Affective Domain related outcome(s)**

a. Follow precautionary measures.

b. Follow installation steps.

c. Follow ethical practices.

VI. **Relevant Theoretical Background**

Sr. No.	Title	Explanation	Syntax	Example Program

1	If	Checks a condition and executes a block of code if it's true.	IF condition THEN statement; END IF;	DECLARE
				x NUMBER := 10;
				BEGIN IF x > 5 THEN DBMS_OUTPUT.PUT_LINE ('x is greater than 5');
				END IF;
				END;

2	If-Then-Else	Executes one block of code if a condition	IF condition THEN	DECLARE x NUMBER := 3;

		is true and another if it's false.	statement; ELSE statement2 ; END IF;	BEGIN
				IF x > 5 THEN DBMS_OUTPUT.PUT_ LINE('x is greater than 5');
				ELSE DBMS_OUTPUT.PUT_ LINE('x is not greater than 5'); END IF;
				END;

3	If-Elsif	Allows checking multiple conditions in sequence and executing correspondin g blocks of code.	IF condition1 THEN statement1 ; ELSIF condition2 THEN statement2 ; ELSE statement3 ; END IF;	DECLARE
				x NUMBER:= 3;
				BEGIN
				IF x > 5 THEN DBMS_OUTPUT.PUT_ LINE('x is greater than 5');
				ELSIF x = 5 THEN DBMS_OUTPUT.PUT_ LINE('x is equal to 5');
				ELSE DBMS_OUTPUT.PUT_ LINE('x is less than 5');
				END IF;
				END;

4	Nested If	Uses one or more if statements inside another if statement.	IF condition1 THEN IF condition2 THEN statement;
			END IF;
			END IF;
				DECLARE
				x NUMBER := 3;
				y NUMBER := 10;
				BEGIN IF x > 5 THEN
				IF y > 5 THEN DBMS_OUTPUT.PUT_ LINE('Both x and y are greater than 5');
				END IF;

				END IF; END;

VII. Required Resources/apparatus/equipment with specifications

Sr.No	Equipment Name with Broad Specifications	Relevant LLO Number
1	Computer system with all necessary components like; motherboard, random access memory (RAM), read-only memory (ROM), internal hard disk drives, Mouse, Keyboard, and RDBMS applications such as Oracle Express Edition,MySql,SQLite,Oracle Apex etc.	All

VIII. Procedure

Implement PL/SQL program based on the given problem

IX. Result(s)

In this Practical we Studied to implement PL/SQL program using conditional Statement.

Practical related questions (Provide space for answers)

Note: Below are a few sample questions for reference. Teacher must design more suchquestions to ensure the achievement of identified CO.

1. List conditional statement in PL/SQL.
2. Describe any three conditional statements in PL/SQL

(Space for answer)

1) ⟶

(I) IF

(II) IF - Then - Else

(III) IF - Elseif

2] ⟶

1] IF

Checks a condition and executes a block of code if it's true

Syntax:

```
IF Condition
  THEN
  Statement;
  END IF;
```

Ex:

```
DECLARE
x NUMBER := 10;
BEGIN IF x > 5 THEN
DBMS_OUTPUT.PUT_LINE ('x is greater than 5');
END IF;
END;
```

2] IF - Then - Else

Executes one block of code if a condition is true and another if it's false.

Syntax:

```
IF
Condition Then
Statement;
ELSE
```

Statement 2
END IF;

Ex:

```
DECLARE
X NUMBER := 3
BEGIN
IF x > 5 THEN
DBMS_OUTPUT.PUT_LINE ('x is greater than 5');
ELSE
DBMS_OUTPUT.PUT_LINE ('x is not greater than 5');
END IF;
END;
/
```

3] Nested if

Uses one or more IF statements inside another IF statement.

Syntax:

```
IF
condition 1
THEN IF
Condition 2
Then
Statement;
END IF;
END IF;
```

Ex :

```
DECLARE
    X NUMBER := 3;
    y NUMBER := 10;
    BEGIN IF x > 5 THEN IF y > 5 THEN
    DBMS_OUTPUT.PUT_LINE ('Both x & y are greater than 5');
    END IF;
    END IF;
    END;
/
```

* Exercise

1] ⟶

```
DECLARE
x NUMBER := 30;
BEGIN IF x>0 THEN
DBMS_OUTPUT.PUT_LINE ('Number is Positive');
END IF;
END;
/
```

2] ⟶

```
DECLARE
age NUMBER := 21;
BEGIN IF age > 18 THEN
DBMS_output.PUT_LINE ('You can Vote');
else
DBMS_output.PUT_LINE ('You cannot Vote');
END IF;
END;
/
```

output:

You can Vote.

12 →

```
DECLARE
x NUMBER := 95;
BEGIN
if x>=75 THEN
DBMS_OUTPUT.Put_LINE ('Distinction');
else if x>=60 and x<=75 THEN
DBms_output.PUT_LINE ('First Class');
else if x>=45 and x<60 THEN
DBms_output.PUT_LINE ('Second Class');
else if x>=40 and x<45 THEN
DBms_Output.PUT_LINE ('Pass Class');
else
DBms_output.PUT_LINE ('Fail');
END if;
END if;
END if;
END if;
END;
/
```

Output:

Distinction.

(iv) Nested If are the Four conditional statement in PL/SQL.

X. **Exercise**

1. Write a PL/SQL program that checks if a given number is positive, and if it is, prints "Number is positive".
2. Write a PL/SQL program that asks the user for their age and then prints "You can vote" if they are over 18, and "You cannot vote" otherwise.
3. Write a PL/SQL program that asks the user for percentage and then assigns grades based on the following conditions:

   Distinction (>=75%)

   First Class (>=60 and <75)

   Second Class (>=45 and <60)

   Pass Class (>=40 and <45)

   Fail (<40)

XI. **References/Suggestions for further reading: include websites/links**

1. https://www.youtube.com/watch?v=yGU4YfSSjdM