Experiment No 23: Implement PL/SQL program based on Exception Handling (User-defined exceptions)

I. Practical Significance: PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR.

II. INDUSTRY / EMPLOYER EXPECTED OUTCOME:

To implement PL/SQL programs with exception handling using user-defined exceptions that leads to more reliable code.

III. COURSE LEVEL LEARNING OUTCOMES (COS):

CO4 - Implement PL/SQL codes for given application.

IV. LABORATORY LEARNING OUTCOME:

Implement PL/SQL program based on Exception Handling (User-defined exceptions).

V. Relevant Affective Domain related outcome(s)

a. Follow precautionary measures.

b. Follow installation steps.

c. Follow ethical practices.

VI. Relevant Theoretical Background

User defined exceptions: This type of users can create their own exceptions according to the need and to raise these exceptions explicitly raise command is used. Example: Divide non-negative integer x by y such that the result is greater than or equal to 1. From the given question we can conclude that there exist two exceptions Division by zero. If result is greater than or equal to 1 means y is less than or equal to x.

```
DECLARE
x int:=&x; /*taking value at run time*/
y int:=&y;
div_r float;
```

```
exp1 EXCEPTION;
exp2 EXCEPTION;
BEGIN
IF y=0 then
raise exp1;
ELSEIF y > x then
raise exp2;
ELSE
div_r:= x / y;
dbms_output.put_line('the result is '||div_r);
END IF;
EXCEPTION
WHEN exp1 THEN
dbms_output.put_line('Error');
dbms_output.put_line('division by zero not allowed');
WHEN exp2 THEN
dbms_output.put_line('Error');
dbms_output.put_line('y is greater than x please check the input');
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

1.Follow the rules while raising the exception.

2.Implement the logic for the given problem

IX. Result(s)

X. Practical related questions (Provide space for answers)

In this Practical we studied to implement PL/SQL Program based on Exception Handling.

* Practical related questions.

1] →

Defining exception in PL/SQL allows you to create custom error handling based on programs.

1. Declare the Exception:
In the declare section of your PL/SQL block or package.
2. Raise the Exception:
Use the RAISE statement in the executable part when a condition is met.
3. Handle the Exception:
In the Exception block specify what to do when the exception is raised.

Ex:

```
-- Declaring the exception.
   DECLARE
      e_high_Salary EXCEPTION;
-- Raise the exception.
   BEGIN
   IF Salary > 50000 THEIN
   RAISE e_high_Salary;
   END IF;
-- Handle the Exception
   EXCEPTION
```

```
WHEN e_high_Salary THEN
DBMS_OUTPUT.PUT_LINE ('Salary is too high');
END;
/
```

2] →

Creating Procedure for User defining Exception

Step1: Declare the Exception

```
CREATE OR REPLACE PROCEDURE check_Student_grade
(Student_grade IN NUMBER) AS
e_invalid_grad EXCEPTION;
```

Step2: Raise the Exception

```
BEGIN
IF Student_grade<0 OR Student_grade>100 THEN
RAISE e_Invalid_grade;
END IF;
```

Step3: Handle the Exception

```
WHEN e_invalid_grade THEN
DBMS_OUTPUT.PUT_line ('Error: Invalid grade!');
END;
/
```

* Exercise

1] →

```
DECLARE
x int := &x;
y int := &y;
div_r float;
exp1 EXCEPTION;
exp2 EXCEPTION;
BEGIN
IF y=0 then
raise exp1;
ELSEIF y>x then
raise exp2;
ELSE
div_r := x/y;
dbms_output.put_line ('the result is '|| div_r);
END;IF;
EXCEPTION
WHEN exp1 THEN
dbms_output.Put_line ('Error');
dbms_output.put_line ('division by zero not allowed');
WHEN exp2 THEN
dbms_output.put_line ('Error');
dbms_output.Put_line ('y is greater than x
                       Please check the input');
END;
/
```

2) →

```
DECLARE
e_invalid_id EXCEPTION;
Customer_id NUMBER;
BEGIN
Customer_id := &enter_customer_id;
IF Customer_id <=0 THEN
RAISE e_invalid_id;
ELSE
DBMS_OUTPUT.PUT_LINE ('Customer ID is
Valid:' || Customer_id);
END IF;
EXCEPTION
WHEN
 e_invalid_id THEN
DBMS_OUTPUT.PUT_Line ('Error: invalid
                       Customer ID');
END;
/
```