Experiment No 22: Implement PL/SQL program based on Exception Handling (Pre-defined exceptions) application

I. **Practical Significance:** Creating a PL/SQL program with exception handling using pre-defined exceptions allows PL/SQL code to deal with unexpected errors or problems that might happen while it runs. This makes PL/SQL program more reliable and prevents it from crashing or giving wrong results when things go wrong unexpectedly. This practical allows students to implement PL/SQL program based on pre-defined exceptions.

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

To implement PL/SQL programs with exception handling using pre-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 (Pre-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**

An *exception* in PL/SQL is a problem or error that happens during the execution of a program, which disrupts the normal flow of the code. The exception block in PL/SQL is important because it helps manage unexpected errors during program execution. It allows you to handle these errors gracefully, preventing program crashes and ensuring a smoother user experience.

Exception Name	**Description**
NO_DATA_FOUND	Raised when a SELECT INTO statement returns no rows.
TOO_MANY_ROWS	Raised when a SELECT INTO statement returns more than one row.

ZERO_DIVIDE	Raised when a division by zero occurs.
INVALID_NUMBER	Raised when trying to convert a character string to a number fails.
VALUE_ERROR	Raised when an arithmetic, conversion, truncation, or constraint error occurs.

Example:

```
DECLARE
temp number;
BEGIN
SELECT p_name into temp from student where p_name='Ameya';
dbms_output.put_line('the p_name is '||temp);
EXCEPTION
WHEN value_error THEN
dbms_output.put_line('Error');
dbms_output.put_line('Change data type of temp to varchar(20)');
END;
```

VALUE_ERROR exception is raised WHEN a statement is executed that resulted in an arithmetic, numeric, string, conversion, or constraint error. This error mainly results from programmer error or invalid data input. Here, it is raised because temp is declared with the datatype number.*temp* datatype should have been varchar2

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.Define the PL/SQL block structure

2.Implement the logic for the given problem

IX. Result(s)

In this Practical we Studied to implement PL/SQL Program based on Exception Handling application

Whod

*Practical related questions.

1. ⟶

- User defined exception

1. These are custom exception that you create based on your Specific business logic requirements
2. You define them using the Exception keyword in the DECLARE Section of a PL/SQL block.
3. These exceptions need to be explicity raised using the RAISE Statement when a Specific condition is met.

- Predefined Exception

1. These are standard exceptions that PL/SQL provides by default.
2. They handle Common errors like NO_DATA_FOUND (when a query returns no rows) or ZERO_DIVIDE (when an attempt is made to divide by zero).
3. They are automatically raised by the System when certain error conditions occur.

2] ⟶

Predefined exception are defined internally & have own name. It is declare in the standard package predefined exception are implicity by oracle like internally defined exception.

We can catch predefined exception using exception handling block & perform the appropriated action for it.

Ex:

```
DECLARE
 V_num1  number (6):=5;
 V_num2  number (6):=0;
 BEGIN
 V_num 1: =V_num 1 / v_num2;
DBMS output .PUT_line (11 v_num1);
Exception
WHEN zero_divided then
DBMS_OUTPUT. PUT_LINE ("Error acquire due to
                        divided by 0");
 END;
/
```

OUTPUT:

```
Error acquire due to divided by 0.
```

* Exercise

1] ⟶

```
DECLARE
num1  NUMBER;
num2  NUMBER;
result  NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('Enter the First number:');
num1:= &num1;
DBMS_OUTPUT. PUT_LINE ('Enter the Second number:');
num2 := &num2;
BEGIN
result := num1/num2;
DBMS_OUTPUT.PUT_LINE ('Result:' || result);
Exception
WHEN Zero_DIVIDE THEN
DBMS_OUTPUT.PUT_LINE ('Error: Division by zero
                                   is not allowed.');
END;
END;
/
```

2] $\longrightarrow$

```
DECLARE
 emp_id  NUMBER;
 emp_Salary  NUMBER;
BEGIN
 emp_id := &emp_id;
BEGIN
 SELECT Salary  INTO emp_salary
  FROM employees
  WHERE  employee_id = emp_id;
 DBMS_OUTPUT.PUT_LINE ('Salary of Employee ID'
 || emp_id || ' is:' || emp_salary);
 Exception
 WHEN   NO_DATA _FOUND THEN
 DBMS OUTPUT. PUT_LINE ('Employee ID not
                                  Found');
 END;
 END;
 /
```