Assignment No .4

Title of Assignment: Unnamed PL/SQL code block: Use of Control structure and Exception handling ismandatory.

Suggested Problem statement:

Consider Tables:

1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status)
2. Fine(Roll_no, Date, Amt)

- Accept Roll_no and Name of Book from user.
- Check the number of days (from date of issue).
- If days are between 15 to 30 then fine amount will be Rs 5per day.
- If no. of days>30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 perday. After submitting the book, status will change from I to R.
- If condition of fine is true, then details will be stored into fine table.
- Also handles the exception by named exception handler or user define exception handler.

OR

Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

Note: Instructor will frame the problem statement for writing PL/SQL block in line with above statement

Course Objective:

Implement PL/SQL Code block for given requirements

Course Outcome:

C306.4 Implement PL/SQL Code block for given requirements

Software Required: - Mysql

Theory: -

A. Control Structures: - PL/SQL allows the use of an IF statement to control the execution of a block of code. In PL/SQL, the IF -THEN - ELSIF - ELSE - END IF construct in code blocks allow specifying certain conditions under which a specific block of code should be execute PL/SQL Control Structures are used to control flow of execution. PL/SQL provides different kinds of statements to provide such type of procedural capabilities. These statements are almost same as that of provided by other languages. The flow of control statements can be classified into the following categories:

- Conditional Control
- Iterative Control
- Sequential Control

Conditional Control:

PL/SQL allows the use of an IF statement to control the execution of a block of code. In PL/SQL, the IF -THEN - ELSIF - ELSE - END IF construct in code blocks allow specifying certain conditions under which a specific block of code should be executed.

Syntax:

```
IF < Condition > THEN
   < Action >
ELSIF <Condition> THEN
   < Action >
ELSE < Action >
END IF;
```

Iterative Control :

Iterative control indicates the ability to repeat or skip sections of a code block. A **loop** marks a sequence of statements that has to be repeated. The keyword loop has to be placed before the first statement in the sequence of statements to be repeated, while the keyword end loop is placed immediately after the last statement in the sequence. Once a loop begins to execute, it will go on forever. Hence a conditional statement that controls the number of times a loop is executed always accompanies loops. PL/SQL supports the following structures for iterative control:

Simple loop : In simple loop, the key word loop should be placed before the first statement in the sequence and the keyword end loop should be written at the end of the sequence to end the loop.

Syntax:

```
Loop

  < Sequence of statements >

 End loop;
```

1.WHILE loop

The while loop executes commands in its body as long as the condition remains true

Syntax:

```
WHILE < condition >
LOOP

    < Action >
END LOOP
```

2.The FOR Loop

The FOR loop can be used when the number of iterations to be executed are known.

Syntax :

```
FOR variable IN [REVERSE] start..end
 LOOP
   < Action >
 END LOOP;
```

The variable in the For Loop need not be declared. Also the increment value cannot be specified. The For Loop variable is always incremented by 1.

3.Sequential Control :

The GOTO Statement

The GOTO statement changes the flow of control within a PL/SQL block. This statement allows execution of a section of code, which is not in the normal flow of control. The entry point into such a block of code is marked using the tags «userdefined name». The GOTO statement can then make use of this user-defined name to jump into that block of code for execution.

Syntax :

```
GOTO jump;
....
<<jump>>
```

B. Exceptions

An Exception is an error situation, which arises during program execution. When an error occurs exception is raised, normal execution is stopped and control transfers to exception handling part. Exception handlers are routines written to handle the exception. The exceptions can be internally defined (system-defined or pre-defined) or User-defined exception.

Syntax:

```
EXCEPTION

      WHEN <ExceptionName> THEN

      <User Defined Action To Be Carried Out>
```

Predefined exception:

Predefined exception is raised automatically whenever there is a violation of Oracle coding rules. Predefined exceptions are those like ZERO_DIVIDE, which is raised automatically when we try to divide a number by zero. Other built-in exceptions are given below. You can handle unexpected Oracle errors using OTHERS handler. It can handle all raised exceptions that are not handled by any other handler. It must always be written as the last handler in exception block.

Exception	**Raised when....**
DUP_VAL_ON_INDEX	When you try to insert a duplicate value into a unique column.
INVALID_CURSOR	It occurs when we try accessing an invalid cursor.
INVALID_NUMBER	On usage of something other than number in place of number value.

LOGIN_DENIED	At the time when user login is denied.
TOO_MANY_ROWS	When a select query returns more than one row and the destination variable can take only single value.
VALUE_ERROR	When an arithmetic, value conversion, truncation, or constraint error occurs.
CURSOR_ALREADY_OPEN	Raised when we try to open an already open cursor.

Predefined exception handlers are declared globally in package STANDARD. Hence we need not have to define them rather just use them. The biggest advantage of exception handling is it improves readability and reliability of the code. Errors from many statements of code can be handles with a single handler. Instead of checking for an error at every point we can just add an exception handler and if any exception is raised it is handled by that. For checking errors at a specific spot it is always better to have those statements in a separate begin – end block.

User Defined Exception Handling:

To trap business rules being violated the technique of raising user-defined exceptions and then handling them, is used. User-defined error conditions must be declared in the declarative part of any PL/SQL block. In the executable part, a check for the condition that needs special attention is made. If that condition exists, the call to the user-defined exception is made using a RAISE statement. The exception once raised is then handled in the Exception handling section of the PL/SQL code block.

Syntax:

```
DECLARE
    < ExceptionName > EXCEPTION ;
 BEGIN
    <SQL Sentence >;
```

Conclusion:

Students are able to Implement PL/SQL Code block for given requirements

Activity to be Submitted by Students

1. Write a PL/SQL block which accepts a cleaner number and returns the cleaners name and salary and update this cleaner’s details by increasing salary by 10%. Use Exception Handling.

2. Write a PL/SQL block to find Largest of three numbers