**Experiment-5**

**Aim:**

Create a simple PL/SQL programs which includes declaration section, executable section and exception –Handling section.

1. Student marks can be selected from the table and printed for those who secured first class and an exception can be raised if no records were found.
2. Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL block.

**Description**

* PL/SQL is a procedural language and which is extension to a non-procedural language SQL.
* A collection of executable statements is called block.
* PL/SQL is a block structured language.

**PL/SQL features**

* It is developed by oracle company in 1980’s
* It supports variables & constraints, SQL bind variables, PL/SQL bind and host variables.
* It supports error handling.
* It supports conditional statements(if,else,if-else).
* It supports interaction controls(while loop,for loop…).
* It supports sub-programs

1.Procedures

2.Functions

3.Packages

4.Triggers

* PL/SQL Block contains 3 sections

1.Declaration section

2.Executable section

3.Exception section`

**Block Structure**

**Syntax**

Declare

[Variable declaration];

Begin

<executable statements>;

Exception when <Exception name> then

<executable statements>;

End;

**Rules**

* Place semicolon(; ) at the end of PL/SQL statements.
* Use forward slash(/) to run PL/SQL block.
* In PL/SQL an error is called exception.
* Section keyword like declare,begin,exception should not contain semicolon(; ).

**PL/SQL variables**

* Variables are used to store data values that are used by PL/SQL.
* PL/SQL variables will support all built in data types and Boolean data types(TRUE/FALSE).
* Bind variables not support the Boolean data types.

**Bind variable:** The variable which is defined at SQL prompt called as bind variables.

**Syntax:**

Var <variable name> type;

Ex: Var name varchar(20);

**Asign values to bind variable**

EXEC: Name :=’ramu’;

**Print bind variable**

Print:Name

**Host variable**

**Syntax:**

Declare <variable name>Data type (size);

**Example:**

Declare

Veno number:=201;

Vname varchar(20):=’ramu’;

Vdate date default sysdate;

**DBMS \_OUTPUT.PUT\_LINE():-** It is used to display the user message and variable name.

Ex: dbms\_output.put\_line((‘employee name is’||Ename);

SET SERVEROUTPUT ON:It is used to activate dbms statements.

**Programs:**

(i) Student marks can be selected from the table and printed for those who secured first class and an exception can be raised if no records were found.

CREATE TABLE student\_marks (

student\_id NUMBER PRIMARY KEY,

student\_name VARCHAR2(50),

marks NUMBER

);

-- Insert sample data into the student\_marks table

INSERT INTO student\_marks VALUES (1, 'Alice', 75);

INSERT INTO student\_marks VALUES (2, 'Bob', 45);

INSERT INTO student\_marks VALUES (3, 'Charlie', 85);

INSERT INTO student\_marks VALUES (4, 'David', 55);

INSERT INTO student\_marks VALUES (5, 'Eve', 65);

-- Commit the changes

COMMIT;

DECLARE

-- Declare a cursor to select students with first class marks

CURSOR student\_cursor IS

SELECT student\_id, student\_name, marks

FROM student\_marks

WHERE marks >= 60;

-- Record variable to hold data from the cursor

student\_rec student\_cursor%ROWTYPE;

-- Variable to check if any rows are fetched

rows\_fetched BOOLEAN := FALSE;

BEGIN

-- Open the cursor

OPEN student\_cursor;

LOOP

-- Fetch each row into the record variable

FETCH student\_cursor INTO student\_rec;

-- Exit the loop when no more rows are fetched

EXIT WHEN student\_cursor%NOTFOUND;

-- Mark that rows have been fetched

rows\_fetched := TRUE;

-- Print the student details

DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || student\_rec.student\_id);

DBMS\_OUTPUT.PUT\_LINE('Student Name: ' || student\_rec.student\_name);

DBMS\_OUTPUT.PUT\_LINE('Marks: ' || student\_rec.marks);

DBMS\_OUTPUT.PUT\_LINE('---');

END LOOP;

-- Close the cursor

CLOSE student\_cursor;

-- Handle case when no rows are found

IF NOT rows\_fetched THEN

DBMS\_OUTPUT.PUT\_LINE('No students found with first class marks.');

END IF;

EXCEPTION

-- Handle any unexpected exceptions

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

/

1. Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL block.

CREATE TABLE stu\_record (

student\_id NUMBER PRIMARY KEY,

student\_name VARCHAR2(50),

marks NUMBER

);

-- Step 2: Insert data and use COMMIT, ROLLBACK, and SAVEPOINT

BEGIN

-- Insert the first student record

INSERT INTO stu\_record (student\_id, student\_name, marks)

VALUES (1, 'Alice', 75);

-- Commit the first transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('First record inserted and committed.');

-- Insert the second student record

INSERT INTO stu\_record (student\_id, student\_name, marks)

VALUES (2, 'Bob', 45);

-- Set a savepoint after the second insertion

SAVEPOINT savepoint\_bob;

DBMS\_OUTPUT.PUT\_LINE('Second record inserted, savepoint created.');

-- Insert the third student record

INSERT INTO stu\_record (student\_id, student\_name, marks)

VALUES (3, 'Charlie', 85);

-- Rollback to the savepoint

ROLLBACK TO savepoint\_bob;

DBMS\_OUTPUT.PUT\_LINE('Rolled back to savepoint. Third record not saved.');

-- Commit the second transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Second record committed.');

END;

/