Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)
School of Computing
B.Tech. – Information Technology

VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211IT201

Course Name : Database System Concepts

Slot No : S 1 2 L 5

DBMS TASK - 6 REPORT

TASK:-**PL/SQL Procedures, functions, Loops**

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU28684	24UECS0058	YEKAMBARAM NANDHA KISHORE

ABSTRACT

The objective of this task is to study and implement various **PL/SQL control structures**, **procedures**, and **functions** to perform conditional, iterative, and modular programming inside Oracle.

PL/SQL enhances SQL with procedural capabilities such as conditional branching using IF, iterative loops (FOR, WHILE, LOOP), and modular constructs (PROCEDURE and FUNCTION).

This exercise demonstrates the use of control structures like **IF — THEN**, **CASE – WHEN**, **GOTO**, **NULL**, **FOR loops**, and **REVERSE loops**, along with examples of creating and executing stored procedures and functions.

Aim:

To implement PL/SQL Procedures, Functions and loops on Number theory and business scenarios.

Procedure: PL/SQL is a combination of SQL along with the procedural features of programming languages.

Questions:

1. Write a PL/SQL block that calculates the average age of players and displays the result.

```
SET SERVEROUTPUT ON;

DECLARE
    v_avg_age NUMBER(5,2);
BEGIN
    SELECT AVG(Age)
    INTO v_avg_age
    FROM Player;
```

```
    DBMS_OUTPUT.PUT_LINE('Average Age of Players = ' || v_avg_age);
END;
/
```

OUTPUT:

Average Age of Players = 31.67

Q2. PL/SQL Block – Insert a New Player Record

```
SET SERVEROUTPUT ON;

DECLARE
BEGIN
    INSERT INTO Player (PlayerID, PlayerName, Age, Role, TeamID)
    VALUES (105, 'Rohit Sharma', 36, 'Batsman', 102);

    DBMS_OUTPUT.PUT_LINE('New Player inserted successfully.');
END;
/
```

Output:

```
New Player inserted successfully.
```

Q3. Function – Total Matches Played by a Specific Player

```
CREATE OR REPLACE FUNCTION TotalMatchesByPlayer(p_playerID IN NUMBER)
RETURN NUMBER
IS
    v_total NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_total
    FROM Match_Player
    WHERE PlayerID = p_playerID;

    RETURN v_total;
END;
/
```

To call the function:

```
SET SERVEROUTPUT ON;
DECLARE
   v_count NUMBER;
BEGIN
   v_count := TotalMatchesByPlayer(1001);
   DBMS_OUTPUT.PUT_LINE('Total matches played: ' || v_count);
END;
/
```

OUTPUT:

Total matches played: 2

Q4. Procedure – Retrieve Even-Numbered Player IDs

```
CREATE OR REPLACE PROCEDURE GetEvenPlayerIDs IS
    CURSOR c1 IS
        SELECT PlayerID, PlayerName
        FROM Player
        WHERE MOD(PlayerID, 2) = 0;
BEGIN
    FOR rec IN c1 LOOP
        DBMS_OUTPUT.PUT_LINE('PlayerID: ' || rec.PlayerID || '  Name: ' ||
rec.PlayerName);
    END LOOP;
END;
/
```

To call the procedure:

```
BEGIN
    GetEvenPlayerIDs;
END;
/
```

OUTPUT:

```
PlayerID: 1002  Name: Ruturaj Gaikwad
PlayerID: 1004  Name: Rohit Sharma
PlayerID: 1006  Name: Baba Aparajith
```

Q5. PL/SQL - IF-THEN Example (Pass/Fail based on Marks)

```
SET SERVEROUTPUT ON;

DECLARE
    v_marks NUMBER;
    v_sid NUMBER := &StudentID;  -- Input student ID
BEGIN
    SELECT marks INTO v_marks FROM Student WHERE s_id = v_sid;

    IF v_marks > 50 THEN
        DBMS_OUTPUT.PUT_LINE('PASS');
    ELSE
        DBMS_OUTPUT.PUT_LINE('FAIL');
    END IF;
```

```
END;
/
```

OUTPUT:

PASS

Q6. Create Table and Insert Employee Data

```
CREATE TABLE Employees (
    Emp_ID   NUMBER PRIMARY KEY,
    Emp_Name VARCHAR2(100),
    Age      NUMBER,
    Salary   NUMBER,
    DOJ      DATE
);
```

```
SQL> DESC EMPLOYEES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMP_ID                                    NOT NULL NUMBER
 EMP_NAME                                           VARCHAR2(100)
 AGE                                                NUMBER
 SALARY                                             NUMBER
 DOJ                                                DATE
```

```
INSERT INTO Employees VALUES (1, 'Alice', 30, 50000, DATE '2015-06-01');
INSERT INTO Employees VALUES (2, 'Bob', 35, 60000, DATE '2012-03-15');
INSERT INTO Employees VALUES (3, 'Carol', 28, 45000, DATE '2019-11-22');
INSERT INTO Employees VALUES (4, 'Dave', 42, 80000, DATE '2010-01-05');

    EMP_ID
----------
EMP_NAME
--------------------------------------------------------------------------------
       AGE     SALARY DOJ
---------- ---------- ---------
         1
Alice
        30      50000 01-JUN-15

         2
Bob
        35      60000 15-MAR-12

         3
Carol
        28      45000 22-NOV-19
```

```
         4
Dave
        42      80000 05-JAN-10
```

Q7. Function – Average Age of Employees

```
CREATE OR REPLACE FUNCTION get_avg_age
RETURN NUMBER
IS
    v_avg_age NUMBER;
BEGIN
    SELECT AVG(Age)
    INTO v_avg_age
    FROM Employees;

    RETURN ROUND(v_avg_age, 2);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/
```

Call the function:

```
SET SERVEROUTPUT ON;

DECLARE
    v_avg NUMBER;
```

```
BEGIN
    v_avg := get_avg_age;
    DBMS_OUTPUT.PUT_LINE('Average age = ' || NVL(TO_CHAR(v_avg), 'NULL'));
END;
/
```

OUTPUT:

```
Average age = 33.75
```

Q8. Simple Numeric FOR Loop

```
BEGIN
    FOR i IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE('i = ' || i);
    END LOOP;
END;
/
```

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

Q9. Cursor FOR Loop – List Employee Details

```
CREATE OR REPLACE PROCEDURE list_emp_names IS
BEGIN
    FOR r IN (SELECT Emp_ID, Emp_Name, Salary FROM Employees ORDER BY Emp_ID)
    LOOP
```

```
 DBMS_OUTPUT.PUT_LINE(r.Emp_ID || ' - ' || r.Emp_Name || ' : ' || r.Salary);


 END LOOP;

END list_emp_names;

/


-- Call the procedure

BEGIN

    list_emp_names;

END;

/
```

OUTPUT:

1 - Alice : 50000

2 - Bob : 60000

3 - Carol : 45000

4 - Dave : 80000

RESULT:

THE QUERIES ARE EXECUTED SUCCESFULLY