**Practical 06 Part II**

**Introduction to Loops in PL/SQL**

Loops allow repeated execution of a block of statements. PL/SQL supports three types of loops: **BASIC LOOP** (Infinite Loop)

**WHILE LOOP** (Condition-based)

**FOR LOOP** (Counter-based)

**BASIC LOOP (Must use EXIT condition)**

A LOOP executes repeatedly until an EXIT condition is met.

**Example: Print numbers from 1 to 5 using LOOP**

SET SERVEROUTPUT ON;

DECLARE

i NUMBER := 1;

BEGIN

LOOP

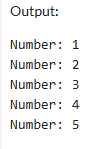
DBMS\_OUTPUT.PUT\_LINE('Number: ' || i);

i := i + 1;

EXIT WHEN i > 5; -- Exit condition

END LOOP;

END;

/ 

**Explanation:** The loop runs indefinitely until i becomes greater than 5.

**WHILE LOOP (Executes as long as condition is TRUE)** A WHILE loop checks a condition before executing the block.

**Example: Print numbers from 1 to 5 using WHILE LOOP**

SET SERVEROUTPUT ON;

DECLARE

i NUMBER := 1;

BEGIN

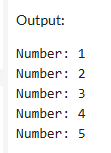
WHILE i <= 5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Number: ' || i);

i := i + 1;

END LOOP;

END;

/ 

**Explanation:** The loop runs as long as i <= 5. When i becomes 6, it stops.

**FOR LOOP (Counter-based)**

A FOR loop runs a fixed number of times.

**Example: Print numbers from 1 to 5 using FOR LOOP**

SET SERVEROUTPUT ON;

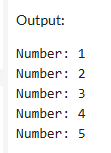
BEGIN

FOR i IN 1..5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Number: ' || i);

END LOOP;

END;

/

**Explanation:** The loop runs automatically from 1 to 5, eliminating the need for a manual counter.

**REVERSE FOR LOOP**

A FOR loop can count **backward** using REVERSE.

**Example: Print numbers from 5 to 1 using FOR LOOP** SET SERVEROUTPUT ON;

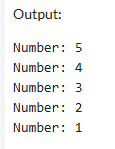
BEGIN

FOR i IN REVERSE 1..5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Number: ' || i);

END LOOP;

END;

/ 

**Explanation:** The loop counts **down** from 5 to 1.

**Simple Tasks for Practice**

Write a **BASIC LOOP** to print numbers from 1 to 10.

Modify the **WHILE LOOP** to print **even numbers** from 2 to 10.

Write a **FOR LOOP** to print the **square of numbers** from 1 to 5.

Create a **REVERSE FOR LOOP** that prints numbers from 10 to 1.

Write a loop that **calculates the sum of numbers from 1 to 5**.

**LOOPS USECASES IN DBMS**

**BASIC LOOP (Must use EXIT condition)** The LOOP statement runs indefinitely unless explicitly stopped with an EXIT condition.

**Example 1: Insert 5 Records into a Table Using LOOP** BEGIN

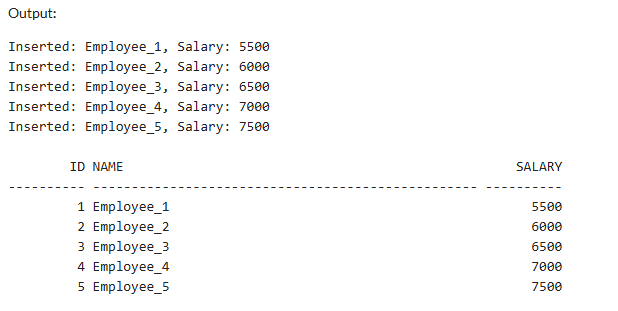
FOR i IN 1..5 LOOP

INSERT INTO employees (id, name, salary) VALUES (i, 'Employee\_' || i, 5000 + (i \* 500));

END LOOP;

COMMIT;

END;

/ 

**Explanation:** Inserts 5 employees with incrementing salaries.

**Example 2: Fetch and Display Employee Names Using LOOP** DECLARE

v\_name employees.name%TYPE;

CURSOR emp\_cursor IS SELECT name FROM employees;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_name;

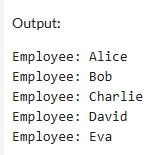
EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || v\_name);

END LOOP;

CLOSE emp\_cursor;

END;

/ 

**Explanation:** Uses a cursor to fetch and print employee names one by one.

**Example 3: Delete Employees with Salary Below 3000 Using LOOP** DECLARE

CURSOR emp\_cursor IS SELECT id FROM employees WHERE salary < 3000; v\_id employees.id%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_id;

EXIT WHEN emp\_cursor%NOTFOUND;

DELETE FROM employees WHERE id = v\_id;

END LOOP;

CLOSE emp\_cursor;

COMMIT;

END;

/

**Explanation:** Deletes employees earning less than 3000.

**Example 4: Update Salaries Using LOOP**

DECLARE

CURSOR emp\_cursor IS SELECT id FROM employees;

v\_id employees.id%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_id;

EXIT WHEN emp\_cursor%NOTFOUND;

UPDATE employees SET salary = salary + 1000 WHERE id = v\_id; END LOOP;

CLOSE emp\_cursor;

COMMIT;

END;

/

**Explanation:** Increases salaries by 1000 for all employees.

**WHILE LOOP (Executes as long as the condition is TRUE) Example 1: Print Employee Names While ID ≤ 5**

DECLARE

v\_id NUMBER := 1;

v\_name employees.name%TYPE;

BEGIN

WHILE v\_id <= 5 LOOP

SELECT name INTO v\_name FROM employees WHERE id = v\_id; DBMS\_OUTPUT.PUT\_LINE('Employee: ' || v\_name);

v\_id := v\_id + 1;

END LOOP;

END;

/

**Explanation:** Fetches and prints employee names for IDs 1 to 5.

**Example 2: Insert Employees Until a Certain Count**

DECLARE

v\_count NUMBER := 0;

BEGIN

WHILE v\_count < 5 LOOP

INSERT INTO employees (id, name, salary) VALUES (v\_count + 10, 'New\_Employee', 4000);

v\_count := v\_count + 1;

END LOOP;

COMMIT;

END;

/

**Explanation:** Inserts 5 new employees.

**Example 3: Fetch and Display Employees with Salary Above 6000** DECLARE

CURSOR emp\_cursor IS SELECT name FROM employees WHERE salary > 6000;

v\_name employees.name%TYPE;

BEGIN

OPEN emp\_cursor;

FETCH emp\_cursor INTO v\_name;

WHILE emp\_cursor%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || v\_name);

FETCH emp\_cursor INTO v\_name;

END LOOP;

CLOSE emp\_cursor;

END;

/

**Explanation:** Fetches employees earning more than 6000.

**Example 4: Deduct Salary Until Minimum Threshold** DECLARE

v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary FROM employees WHERE id = 1; WHILE v\_salary > 3000 LOOP

UPDATE employees SET salary = salary - 500 WHERE id = 1; v\_salary := v\_salary - 500;

END LOOP;

COMMIT;

END;

/

**Explanation:** Deducts salary until it reaches 3000.

**FOR LOOP (Counter-based loop, runs a fixed number of times)**

**Example 1: Insert 10 Employees Using FOR LOOP**

BEGIN

FOR i IN 1..10 LOOP

INSERT INTO employees (id, name, salary) VALUES (i + 100, 'Emp\_' || i, 6000);

END LOOP;

COMMIT;

END;

/

**Explanation:** Inserts 10 employees with unique IDs.

**Example 2: Display First 5 Employees**

BEGIN

FOR emp IN (SELECT name FROM employees WHERE ROWNUM <= 5) LOOP DBMS\_OUTPUT.PUT\_LINE('Employee: ' || emp.name);

END LOOP;

END;

/

**Explanation:** Prints the first 5 employee names.

**Example 3: Increase Salaries in a Range**

BEGIN

FOR i IN 1..10 LOOP

UPDATE employees SET salary = salary + 500 WHERE id = i; END LOOP;

COMMIT;

END;

/

**Explanation:** Increases salaries of employees with IDs 1 to 10.

**Example 4: Delete Employees with ID Greater Than 50** BEGIN

FOR i IN (SELECT id FROM employees WHERE id > 50) LOOP DELETE FROM employees WHERE id = i.id;

END LOOP;

COMMIT;

END;

/

**Explanation:** Deletes employees with IDs greater than 50.

**Loops with database Simple Tasks for Practice**

1. Write a **LOOP** to insert **5 new departments** into a departments table. 2. Modify the **WHILE LOOP** to **increase salaries** until they reach 10,000. 3. Write a **FOR LOOP** to display **employee details** for IDs 1 to 5. 4. Create a **cursor-based LOOP** that prints **employee names and salaries**. 5. Write a loop that **calculates the total salary** of all employees.

-- Step 1: Create the departments table

CREATE TABLE departments (

department\_id NUMBER PRIMARY KEY,

department\_name VARCHAR2(100)

);

-- Step 2: Create the employees table

CREATE TABLE employees (

id NUMBER PRIMARY KEY,

name VARCHAR2(100),

salary NUMBER,

department\_id NUMBER,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

-- Step 3: Insert 5 new departments into the departments table using a LOOP

BEGIN

FOR i IN 1..5 LOOP

INSERT INTO departments (department\_id, department\_name)

VALUES (i, 'Department\_' || i);

END LOOP;

COMMIT;

END;

/

-- Step 4: Insert 5 employees for testing

BEGIN

FOR i IN 1..5 LOOP

INSERT INTO employees (id, name, salary, department\_id)

VALUES (i, 'Employee\_' || i, 4000 + (i \* 500), i); -- Assigning unique salary to each employee

END LOOP;

COMMIT;

END;

/

-- Step 5: WHILE LOOP to increase salaries until they reach 10,000

DECLARE

v\_salary employees.salary%TYPE;

BEGIN

-- Assume we are starting with the salary of employee with ID = 1

BEGIN

SELECT salary INTO v\_salary FROM employees WHERE id = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employee found with ID = 1');

RETURN; -- Exit the block if no data is found

END;

-- WHILE LOOP to increase salary until it reaches 10,000

WHILE v\_salary < 10000 LOOP

v\_salary := v\_salary + 500; -- Increase salary by 500 each time

UPDATE employees SET salary = v\_salary WHERE id = 1;

COMMIT;

END LOOP;

END;

/

-- Step 6: FOR LOOP to display employee details for IDs 1 to 5

DECLARE

v\_id employees.id%TYPE;

v\_name employees.name%TYPE;

v\_salary employees.salary%TYPE;

BEGIN

FOR i IN 1..5 LOOP

BEGIN

SELECT id, name, salary INTO v\_id, v\_name, v\_salary

FROM employees WHERE id = i;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employee found with ID = ' || i);

CONTINUE; -- Continue to the next iteration

END;

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_id || ', Name: ' || v\_name || ', Salary: ' || v\_salary);

END LOOP;

END;

/

-- Step 7: Cursor-based LOOP to print employee names and salaries

DECLARE

CURSOR emp\_cursor IS

SELECT name, salary FROM employees;

v\_name employees.name%TYPE;

v\_salary employees.salary%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_name, v\_salary;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || v\_name || ', Salary: ' || v\_salary);

END LOOP;

CLOSE emp\_cursor;

END;

/

-- Step 8: Loop to calculate the total salary of all employees

DECLARE

v\_total\_salary NUMBER := 0;

CURSOR emp\_cursor IS

SELECT salary FROM employees;

v\_salary employees.salary%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_salary;

EXIT WHEN emp\_cursor%NOTFOUND;

v\_total\_salary := v\_total\_salary + v\_salary;

END LOOP;

CLOSE emp\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Total Salary of All Employees: ' || v\_total\_salary);

END;

/

