

**Name-Harshit Selarka**  
**section-B2**  
**prn-23070521131**

## **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;
```

```
/
```

### Output

Number
1
2
3
4
5

**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.

### Output

Number
1
2
3
4
5

## 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;
```

```
/
```

### Output

Number
1
2
3
4
5

**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;
```

```
/
```

### Output

Number
5
4
3
2
1

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

## Simple Tasks for Practice

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

### Output

Number
1
2
3
4
5
6
7
8
9
10

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

### Output

Even_Number
2
4
6
8
10

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

### Output

Number	Square
1	1
2	4
3	9
4	16
5	25

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

### Output

Number
10
9
8
7
6
5
4
3
2
1

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

### Output

Total_Sum
15

## 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  $\leq$  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.