NAME:SHREYA BANSAL
PRN: 23070521144
BATCH: B2

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.

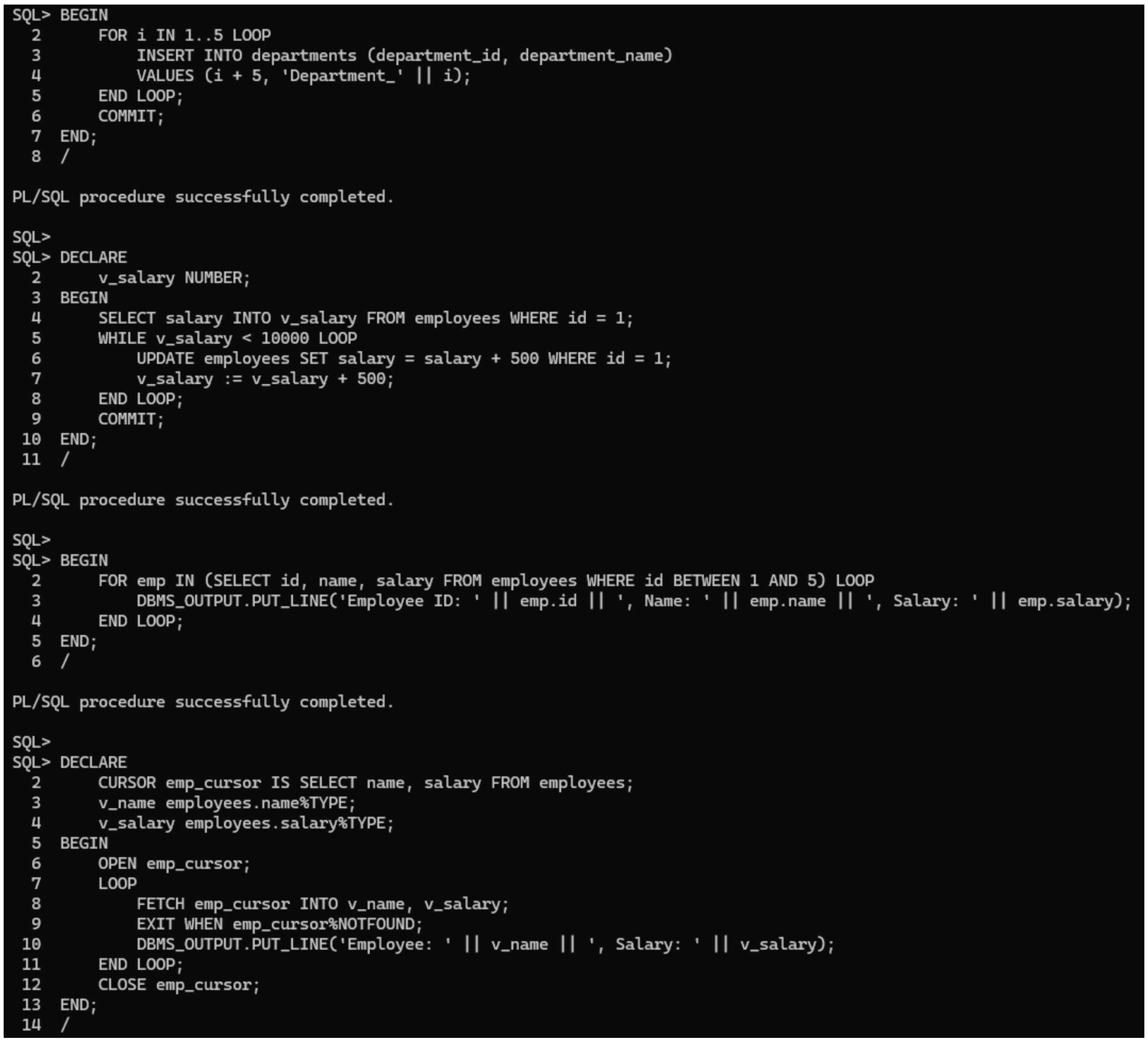
Example 4: Delete Employees with ID Greater Than 50

```
BEGIN

    FOR i IN (SELECT id FROM employees WHERE id > 50) LOOP

        DELETE FROM employees WHERE id = i.id;

    END LOOP;

    COMMIT;

END;

/
```

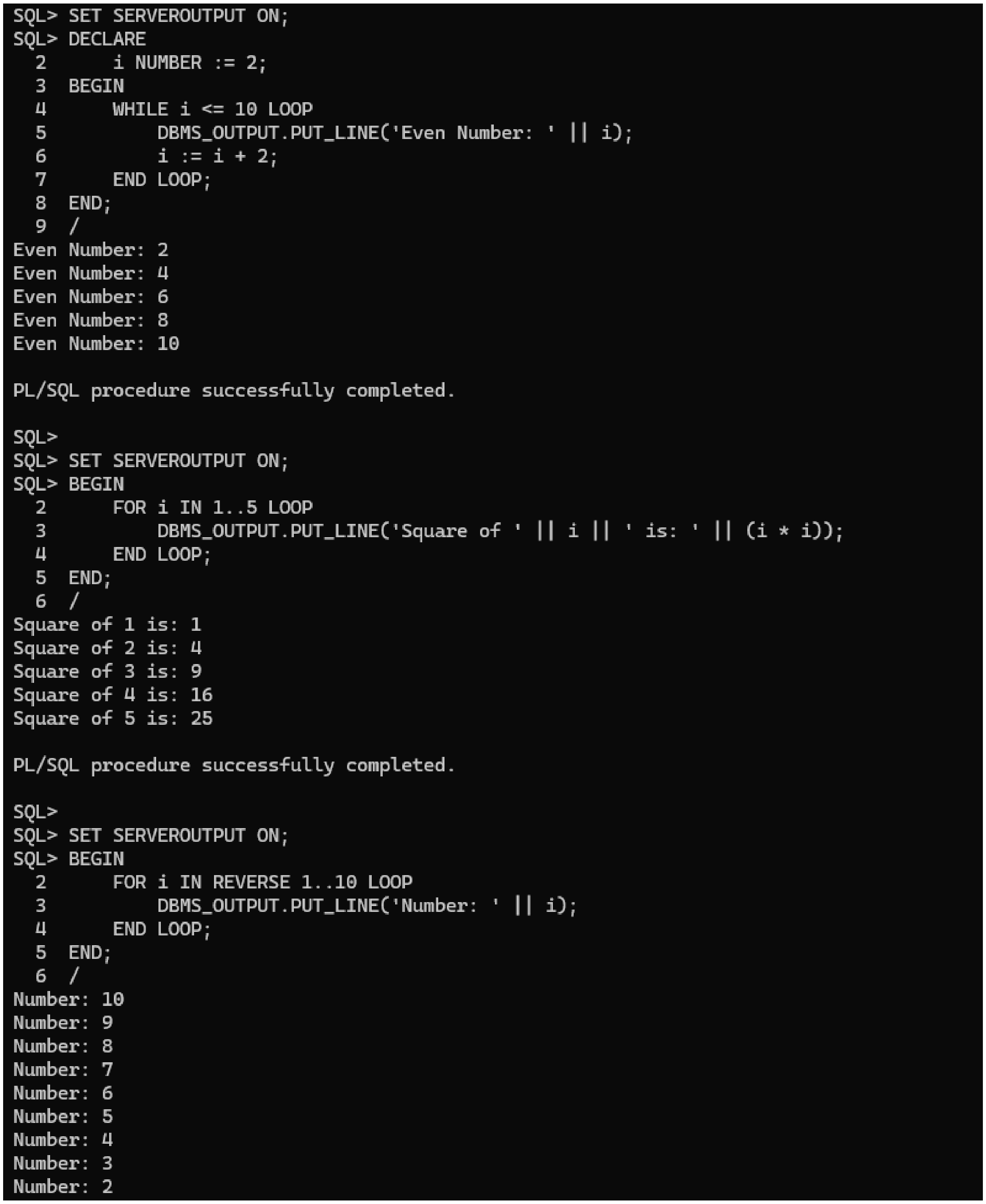
Explanation: Deletes employees with IDs greater than 50.

Loops with database Simple Tasks for Practice

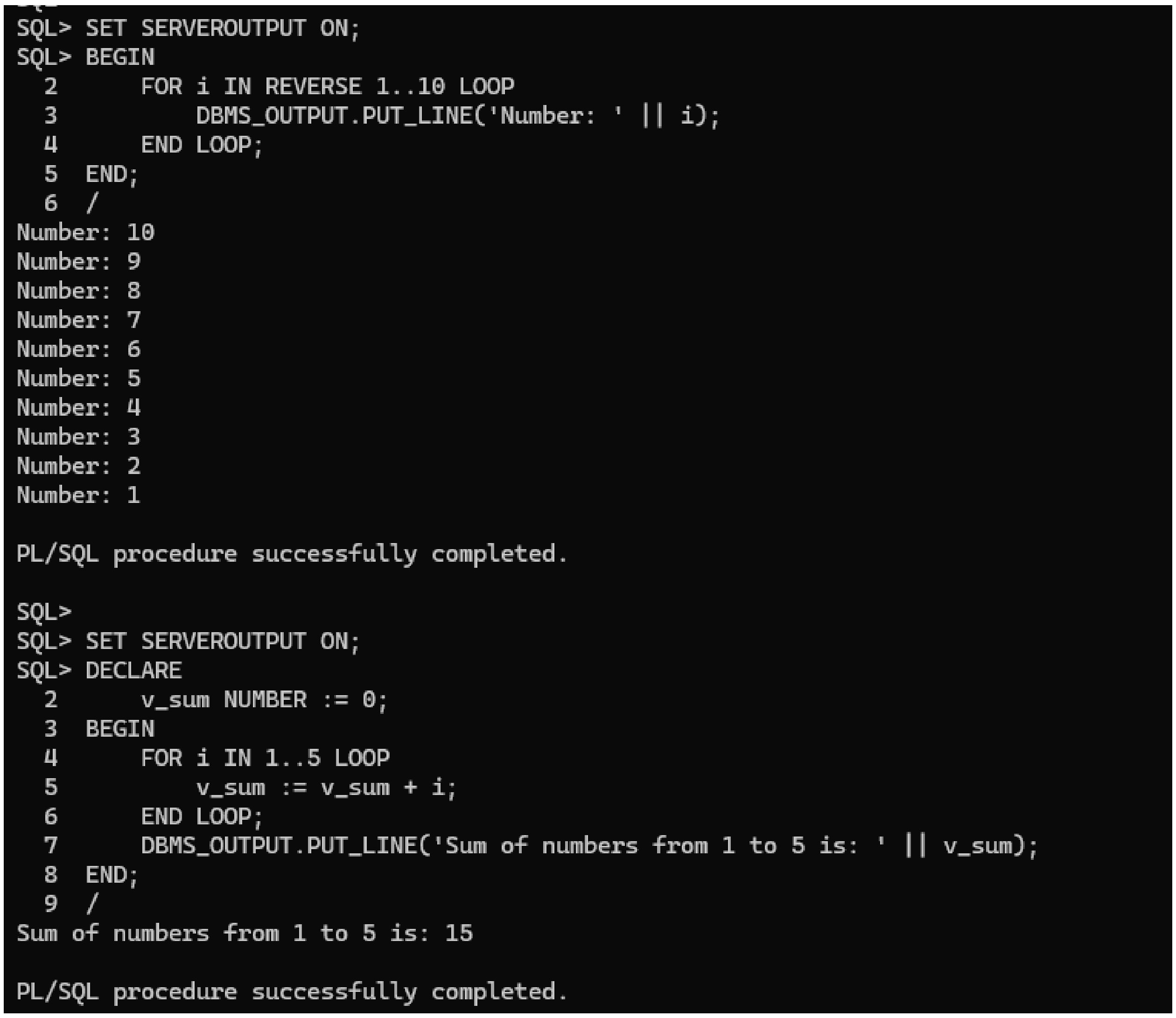
1. Write a **LOOP** to insert **5 new departments** into a `departments` table. 2. Modify the **WHILE LOOP** to **increase salaries** until they reach 10,000. 3. Write a **FOR LOOP** to display **employee details** for IDs 1 to 5. 4. Create a **cursor-based LOOP** that prints **employee names and salaries**.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



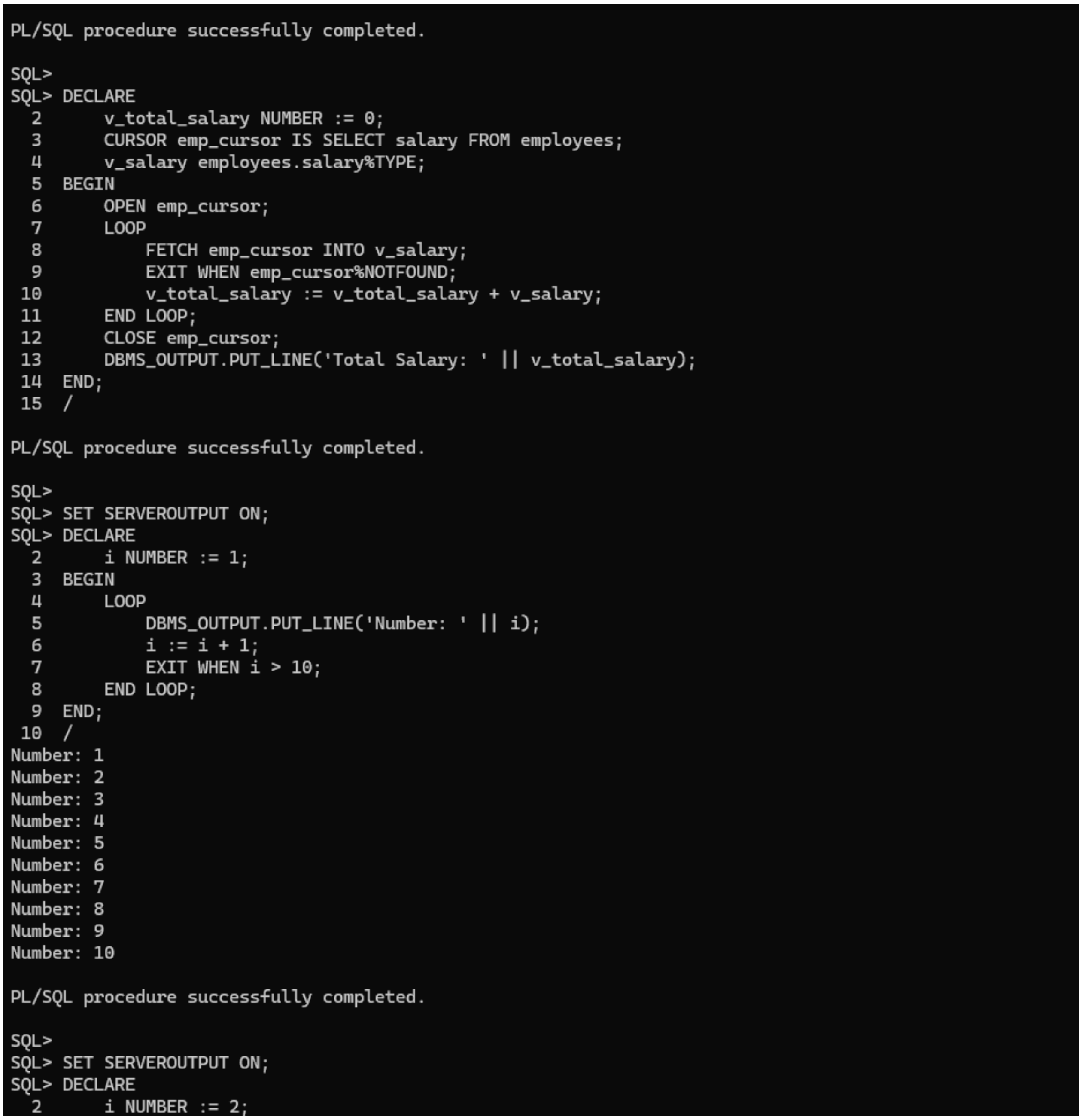
```
SQL> BEGIN
    FOR i IN 1..5 LOOP
        INSERT INTO departments (department_id, department_name)
        VALUES (i + 5, 'Department_' || i);
    END LOOP;
    COMMIT;
END;
/

PL/SQL procedure successfully completed.

SQL>
SQL> DECLARE
    v_salary NUMBER;
BEGIN
    SELECT salary INTO v_salary FROM employees WHERE id = 1;
    WHILE v_salary < 10000 LOOP
        UPDATE employees SET salary = salary + 500 WHERE id = 1;
        v_salary := v_salary + 500;
    END LOOP;
    COMMIT;
END;
/

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
    FOR emp IN (SELECT id, name, salary FROM employees WHERE id BETWEEN 1 AND 5) LOOP
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.id || ', Name: ' || emp.name || ', Salary: ' || emp.salary);
    END LOOP;
END;
/

PL/SQL procedure successfully completed.

SQL>
SQL> 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;
/
```



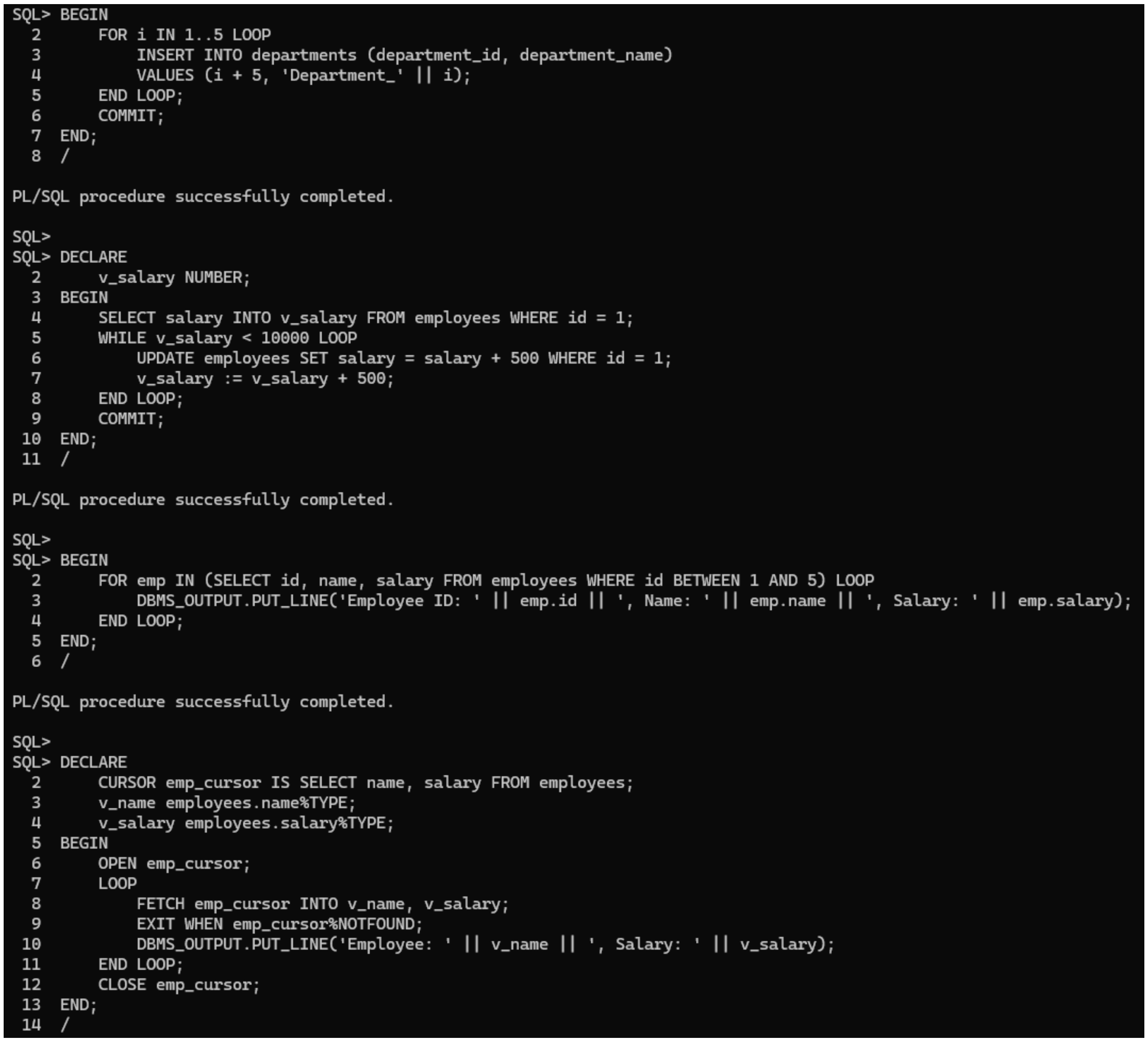
```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
         i NUMBER := 2;
     BEGIN
         WHILE i <= 10 LOOP
             DBMS_OUTPUT.PUT_LINE('Even Number: ' || i);
             i := i + 2;
         END LOOP;
     END;
     /
Even Number: 2
Even Number: 4
Even Number: 6
Even Number: 8
Even Number: 10

PL/SQL procedure successfully completed.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
         FOR i IN 1..5 LOOP
             DBMS_OUTPUT.PUT_LINE('Square of ' || i || ' is: ' || (i * i));
         END LOOP;
     END;
     /
Square of 1 is: 1
Square of 2 is: 4
Square of 3 is: 9
Square of 4 is: 16
Square of 5 is: 25

PL/SQL procedure successfully completed.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
         FOR i IN REVERSE 1..10 LOOP
             DBMS_OUTPUT.PUT_LINE('Number: ' || i);
         END LOOP;
     END;
     /
Number: 10
Number: 9
Number: 8
Number: 7
Number: 6
Number: 5
Number: 4
Number: 3
Number: 2
```



```
SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
         FOR i IN REVERSE 1..10 LOOP
             DBMS_OUTPUT.PUT_LINE('Number: ' || i);
         END LOOP;
     END;
     /
Number: 10
Number: 9
Number: 8
Number: 7
Number: 6
Number: 5
Number: 4
Number: 3
Number: 2
Number: 1

PL/SQL procedure successfully completed.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
         v_sum NUMBER := 0;
     BEGIN
         FOR i IN 1..5 LOOP
             v_sum := v_sum + i;
         END LOOP;
         DBMS_OUTPUT.PUT_LINE('Sum of numbers from 1 to 5 is: ' || v_sum);
     END;
     /
Sum of numbers from 1 to 5 is: 15

PL/SQL procedure successfully completed.
```



```
PL/SQL procedure successfully completed.

SQL>
SQL> 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: ' || v_total_salary);
     END;
     /

PL/SQL procedure successfully completed.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
         i NUMBER := 1;
     BEGIN
         LOOP
             DBMS_OUTPUT.PUT_LINE('Number: ' || i);
             i := i + 1;
             EXIT WHEN i > 10;
         END LOOP;
     END;
     /
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
Number: 10

PL/SQL procedure successfully completed.

SQL>
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
         i NUMBER := 2;
```



```
SQL> BEGIN
         FOR i IN 1..5 LOOP
             INSERT INTO departments (department_id, department_name)
             VALUES (i + 5, 'Department_' || i);
         END LOOP;
         COMMIT;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
SQL> DECLARE
         v_salary NUMBER;
     BEGIN
         SELECT salary INTO v_salary FROM employees WHERE id = 1;
         WHILE v_salary < 10000 LOOP
             UPDATE employees SET salary = salary + 500 WHERE id = 1;
             v_salary := v_salary + 500;
         END LOOP;
         COMMIT;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
         FOR emp IN (SELECT id, name, salary FROM employees WHERE id BETWEEN 1 AND 5) LOOP
             DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.id || ', Name: ' || emp.name || ', Salary: ' || emp.salary);
         END LOOP;
     END;
     /

PL/SQL procedure successfully completed.

SQL>
SQL> 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;
     /
```