Name : Haris **Roll # 23P-0573**

1. Create a PL/SQL block that computes and prints the bonus amount for a given Employee based on the employee's salary. Accept the employee number as user input with a SQL*Plus substitution Variable. a. If the employee's salary is less than 1,000, set the bonus amount for the Employee to 10% of the salary. b. If the employee's salary is between 1,000 and 1,500, set the bonus amount for the employee to 15% of the salary. c. If the employee's salary exceeds 1,500, set the bonus amount for the employee to 20% of the salary. d. If the employee's salary is NULL, set the bonus amount for the employee to 0.

SET SERVEROUTPUT ON;

DECLARE e_empno employees.employee_id%TYPE; e_empsal employees.salary%TYPE; e_bonus NUMBER(10,2); BEGIN -- Accept employee number from user input e_empno := &empno;

```
-- Fetch the employee's salary
SELECT salary INTO e_empsal
FROM employees
WHERE employee_id = e_empno;

-- Calculate the bonus based on salary conditions
IF e_empsal < 1000 THEN
    e_bonus := e_empsal * 0.10; -- 10% bonus
ELSIF e_empsal >= 1000 AND e_empsal <= 1500 THEN
    e_bonus := e_empsal * 0.15; -- 15% bonus
ELSIF e_empsal > 1500 THEN
    e_bonus := e_empsal * 0.20; -- 20% bonus
ELSE
    e_bonus := 0; -- If salary is NULL
```

```
END IF;

-- Display the bonus
DBMS_OUTPUT.PUT_LINE('Bonus for employee ' || e_empno || ' is: ' ||
e_bonus);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Employee with ID ' || e_empno || ' not found.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    e_empno employees.employee_id%TYPE;
    e_empsal employees.salary%TYPE;
    e_bonus NUMBER(10,2);
BEGIN
    -- Accept employee number from user input
    e_empno := &empno;

    -- Fetch the employee's salary
    SELECT salary INTO e_empsal
    FROM employees
    WHERE employee_id = e_empno;
```

Script Output

Task completed in 4.672 seconds

```
XCEPTION
   WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Employee with ID ' || e_empno || ' not found.');
   WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
ND;

L/SQL procedure successfully completed.
```

2.

2 : Write a pl/sql block in sql that ask a user for employee id than it checks its commission if commission is null than it updates salary of that employee by adding commission into salary.

SET SERVEROUTPUT ON;

DECLARE v_emp_id employees.employee_id%TYPE; v_salary employees.salary%TYPE; v_commission employees.commission_pct%TYPE; v_new_salary NUMBER(10,2); BEGIN v_emp_id := &emp_id;

```
SELECT salary, NVL(commission_pct, 0)
INTO v_salary, v_commission
FROM employees
WHERE employee_id = v_emp_id;

IF v_commission = 0 THEN
    v_new_salary := v_salary;
    DBMS_OUTPUT.PUT_LINE('Commission is already NULL or 0. No update
needed.');
ELSE
    v_new_salary := v_salary + (v_salary * v_commission);

    UPDATE employees
    SET salary = v_new_salary
    WHERE employee_id = v_emp_id;

    DBMS_OUTPUT.PUT_LINE('Updated salary for employee ' || v_emp_id ||
' is: ' || v_new_salary);
END IF;
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_emp_id employees.employee_id%TYPE;
    v_salary employees.salary%TYPE;
    v_commission employees.commission_pct%TYPE;
    v_new_salary NUMBER(10,2);
BEGIN
    v_emp_id := &emp_id;

    SELECT salary, NVL(commission_pct, 0)
    INTO v_salary, v_commission
    FROM employees
    WHERE employee_id = v_emp_id;
```

Script Output

Task completed in 13.474 seconds

```
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee with ID ' || v_emp_id || ' not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;

PL/SQL procedure successfully completed.
```

3: Write a PL/SQL block to obtain the department name of the employee who works for

deptno 30

SET SERVEROUTPUT ON;

DECLARE v_dept_name departments.department_name%TYPE; BEGIN SELECT department_name INTO v_dept_name FROM departments WHERE department_id = 30;

```
DBMS_OUTPUT.PUT_LINE('Department Name: ' || v_dept_name);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No department found for department ID 30.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_dept_name departments.department_name%TYPE;
BEGIN
    SELECT department_name
    INTO v_dept_name
    FROM departments
    WHERE department_id = 30;

    DBMS_OUTPUT.PUT_LINE('Department Name: ' || v_dept_name);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

Script Output

Task completed in 0.118 seconds

```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;

PL/SQL procedure successfully completed.

Department Name: Purchasing


PL/SQL procedure successfully completed.
```

Write a PL /SQL block to find the nature of job of the employee whose deptno is 20(to be passed as an argument)

SET SERVEROUTPUT ON;

DECLARE v_deptno NUMBER := &deptno; v_job employees.job_id%TYPE; BEGIN SELECT job_id INTO v_job FROM employees WHERE department_id = v_deptno FETCH FIRST 1 ROWS ONLY;

```
DBMS_OUTPUT.PUT_LINE('Job for department ' || v_deptno || ' is: ' ||
v_job);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No employee found in department ' || v_deptno || '.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_deptno NUMBER := &deptno;
    v_job employees.job_id%TYPE;
BEGIN
    SELECT job_id
    INTO v_job
    FROM employees
    WHERE department_id = v_deptno
    FETCH FIRST 1 ROWS ONLY;

    DBMS_OUTPUT.PUT_LINE('Job for department ' || v_deptno || ' is: ' || v_job);
```

```
Script Output
Task completed in 0.021 seconds
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
D;

/SQL procedure successfully completed.

partment Name: Purchasing


/SQL procedure successfully completed.
```

Write a PL/SQL block to update the salary of the employee with a 10% increase whose empno is to be passed as an argument for the procedure

SET SERVEROUTPUT ON;

DECLARE v_empno NUMBER := &empno; v_salary employees.salary%TYPE; v_new_salary NUMBER(10,2); BEGIN SELECT salary INTO v_salary FROM employees WHERE employee_id = v_empno;

```
v_new_salary := v_salary * 1.10;

UPDATE employees
SET salary = v_new_salary
WHERE employee_id = v_empno;

DBMS_OUTPUT.PUT_LINE('Updated salary for employee ' || v_empno || '
is: ' || v_new_salary);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || v_empno || '.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_empno NUMBER := &empno;
    v_salary employees.salary%TYPE;
    v_new_salary NUMBER(10,2);
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = v_empno;

    v_new_salary := v_salary * 1.10;
```

Script Output

Task completed in 3.985 seconds

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || v_empno || '.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
ND;
o employee found with ID 23.


L/SQL procedure successfully completed.
```

Write a procedure to add an amount of Rs.1000 for the employees whose salaries is greater than 5000 and who belongs to the deptno passed as an argument.

CREATE OR REPLACE PROCEDURE Increase_Salary(p_deptno NUMBER) IS BEGIN UPDATE employees SET salary = salary + 1000 WHERE department_id = p_deptno AND salary > 5000;

```
DBMS_OUTPUT.PUT_LINE('Salaries updated for employees in department '
|| p_deptno);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No employees found in department ' || p_deptno || ' with salary greater than 5000.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END Increase_Salary; /

```
--23P-0573
CREATE OR REPLACE PROCEDURE Increase_Salary(p_deptno NUMBER) IS
BEGIN
    UPDATE employees
    SET salary = salary + 1000
    WHERE department_id = p_deptno AND salary > 5000;

    DBMS_OUTPUT.PUT_LINE('Salaries updated for employees in department ' || p_deptno);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employees found in department ' || p_deptno || ' with salary greater than 5000.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END Increase_Salary;
/
```

Script Output

Task completed in 0.166 seconds

```
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
No employee found with ID 23.


PL/SQL procedure successfully completed.


Procedure INCREASE_SALARY compiled
```

Create views for following purposes: -

a. Display each designation and number of employees with that particular Designation.

CREATE OR REPLACE VIEW Designation_Count AS SELECT job_id, COUNT(*) AS num_employees FROM employees GROUP BY job_id;

```
--23P-0573
CREATE OR REPLACE VIEW Designation_Count AS
SELECT job_id, COUNT(*) AS num_employees
FROM employees
GROUP BY job_id;
```

Script Output

Task completed in 0.043 seconds

```
'iew DESIGNATION_COUNT created.
```

b. The organization wants to display only the details like empno, empname , deptno , deptname of all the employee except king.

CREATE OR REPLACE VIEW Employee_Details_Excluding_King AS SELECT e.employee_id, e.first_name || ' ' || e.last_name AS emp_name, e.department_id, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE UPPER(e.last_name) <> 'KING';

```
--23P-0573
CREATE OR REPLACE VIEW Employee_Details_Excluding_King AS
SELECT e.employee_id, e.first_name || ' ' || e.last_name AS emp_name,
       e.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE UPPER(e.last_name) <> 'KING';
```

Script Output

Task completed in 0.029 seconds

```
View DESIGNATION_COUNT created.
```

c. The organization wants to display only the details empno, empname, deptno, deptname of the employees.

CREATE OR REPLACE VIEW Employee_Basic_Details AS SELECT e.employee_id, e.first_name || ' ' || e.last_name AS emp_name, e.department_id, d.department_name FROM employees e JOIN departments d ON e.department_id = d.department_id;

```
--23P-0573
CREATE OR REPLACE VIEW Employee_Basic_Details AS
SELECT e.employee_id, e.first_name || ' ' || e.last_name AS emp_name,
       e.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

Script Output

Task completed in 0.026 seconds

```
iew EMPLOYEE_BASIC_DETAILS created.
```

Write a PL/SQL code that takes two inputs from user, add them and store the sum in new variable and show the output.

SET SERVEROUTPUT ON;

DECLARE v_num1 NUMBER := &num1; v_num2 NUMBER := &num2; v_sum NUMBER; BEGIN v_sum := v_num1 + v_num2; DBMS_OUTPUT.PUT_LINE('The sum of ' || v_num1 || ' and ' || v_num2 || ' is: ' || v_sum); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_num1 NUMBER := &num1;
    v_num2 NUMBER := &num2;
    v_sum NUMBER;
BEGIN
    v_sum := v_num1 + v_num2;
    DBMS_OUTPUT.PUT_LINE('The sum of ' || v_num1 || ' and ' || v_num2 || ' is: ' || v_sum);
END;
/
```

Script Output

Task completed in 4.125 seconds

```
   v_sum NUMBER;
EGIN
   v_sum := v_num1 + v_num2;
   DBMS_OUTPUT.PUT_LINE('The sum of ' || v_num1 || ' and ' || v_num2 || ' is: ' || v_sum);
ND;
he sum of 23 and 34 is: 57


L/SQL procedure successfully completed.
```

10. Write a PL/SQL code that takes two inputs, lower boundary and upper boundary, then print the sum of all the numbers between the boundaries INCLUSIVE.

SET SERVEROUTPUT ON;

DECLARE v_lower NUMBER := &lower_bound; v_upper NUMBER := &upper_bound; v_sum NUMBER := 0; v_counter NUMBER; BEGIN IF v_lower > v_upper THEN DBMS_OUTPUT.PUT_LINE('Lower boundary should be less than or equal to upper boundary.'); ELSE FOR v_counter IN v_lower..v_upper LOOP v_sum := v_sum + v_counter; END LOOP;

```
   DBMS_OUTPUT.PUT_LINE('The sum of numbers from ' || v_lower || ' to
' || v_upper || ' is: ' || v_sum);
END IF;
```

END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_lower NUMBER := &lower_bound;
    v_upper NUMBER := &upper_bound;
    v_sum NUMBER := 0;
    v_counter NUMBER;
BEGIN
    IF v_lower > v_upper THEN
        DBMS_OUTPUT.PUT_LINE('Lower boundary should be less than or equal to upper boundary.');
    ELSE
        FOR v_counter IN v_lower..v_upper LOOP
            v_sum := v_sum + v_counter;
        END LOOP;
```

Script Output ×

Task completed in 3.75 seconds

```
        END LOOP;

        DBMS_OUTPUT.PUT_LINE('The sum of numbers from ' || v_lower || ' to ' || v_upper || ' is: ' || v_sum);
    END IF;
ND;
he sum of numbers from 43 to 45 is: 132


L/SQL procedure successfully completed.
```

Write a PL/SQL code to retrieve the employee name, hiredate, and the department name in which he works, whose number is input by the user.

SET SERVEROUTPUT ON;

DECLARE v_emp_id employees.employee_id%TYPE := &emp_id; v_emp_name employees.first_name%TYPE; v_hire_date employees.hire_date%TYPE; v_dept_name departments.department_name%TYPE; BEGIN SELECT e.first_name || ' ' || e.last_name, e.hire_date, d.department_name INTO v_emp_name, v_hire_date, v_dept_name FROM employees e JOIN departments d ON e.department_id = d.department_id WHERE e.employee_id = v_emp_id;

```
DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
DBMS_OUTPUT.PUT_LINE('Hire Date: ' || v_hire_date);
DBMS_OUTPUT.PUT_LINE('Department Name: ' || v_dept_name);
```

EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || v_emp_id || '.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_emp_id employees.employee_id%TYPE := &emp_id;
    v_emp_name employees.first_name%TYPE;
    v_hire_date employees.hire_date%TYPE;
    v_dept_name departments.department_name%TYPE;
BEGIN
    SELECT e.first_name || ' ' || e.last_name, e.hire_date, d.department_name
    INTO v_emp_name, v_hire_date, v_dept_name
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    WHERE e.employee_id = v_emp_id;
```

Script Output

Task completed in 4.69 seconds

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || v_emp_id || '.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
No employee found with ID 9001.


PL/SQL procedure successfully completed.
```

Write a PL/SQL code to check whether the given number is palindrome or not.

SET SERVEROUTPUT ON;

DECLARE v_num NUMBER := # v_reverse NUMBER := 0; v_temp NUMBER; v_digit NUMBER; BEGIN v_temp := v_num;

```
WHILE v_temp > 0 LOOP
    v_digit := MOD(v_temp, 10);
    v_reverse := (v_reverse * 10) + v_digit;
    v_temp := TRUNC(v_temp / 10);
END LOOP;

IF v_num = v_reverse THEN
    DBMS_OUTPUT.PUT_LINE(v_num || ' is a palindrome.');
ELSE
    DBMS_OUTPUT.PUT_LINE(v_num || ' is not a palindrome.');
```

```
END IF;


END; /
```

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_num NUMBER := &num;
    v_reverse NUMBER := 0;
    v_temp NUMBER;
    v_digit NUMBER;
BEGIN
    v_temp := v_num;

    WHILE v_temp > 0 LOOP
        v_digit := MOD(v_temp, 10);
        v_reverse := (v_reverse * 10) + v_digit;
        v_temp := TRUNC(v_temp / 10);
```

Script Output

Task completed in 3.054 seconds

```
        DBMS_OUTPUT.PUT_LINE(v_num || ' is a palindrome.');
    ELSE
        DBMS_OUTPUT.PUT_LINE(v_num || ' is not a palindrome.');
    END IF;
END;
56 is not a palindrome.


PL/SQL procedure successfully completed.
```

13. Write a PL/SQL code that takes all the required inputs from the user for the Employee table and then insert it into the Employee and Department table in the database.

SET SERVEROUTPUT ON;

DECLARE v_emp_id employees.employee_id%TYPE := &emp_id; v_first_name employees.first_name%TYPE := '&first_name'; v_last_name employees.last_name%TYPE := '&last_name'; v_salary employees.salary%TYPE := &salary; v_hire_date

employees.hire_date%TYPE := TO_DATE('&hire_date', 'YYYY-MM-DD'); v_dept_id departments.department_id%TYPE := &dept_id; v_dept_name departments.department_name%TYPE := '&dept_name'; BEGIN INSERT INTO departments (department_id, department_name) VALUES (v_dept_id, v_dept_name) ON CONFLICT (department_id) DO NOTHING; -- Avoid duplicate department insertions

```
INSERT INTO employees (employee_id, first_name, last_name, salary,
hire_date, department_id)
VALUES (v_emp_id, v_first_name, v_last_name, v_salary, v_hire_date,
v_dept_id);

DBMS_OUTPUT.PUT_LINE('Employee and Department details inserted
successfully.');
```

EXCEPTION WHEN DUP_VAL_ON_INDEX THEN DBMS_OUTPUT.PUT_LINE('Error: Duplicate Employee ID or Department ID.'); WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM); END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_emp_id employees.employee_id%TYPE := &emp_id;
    v_first_name employees.first_name%TYPE := '&first_name';
    v_last_name employees.last_name%TYPE := '&last_name';
    v_salary employees.salary%TYPE := &salary;
    v_hire_date employees.hire_date%TYPE := TO_DATE('&hire_date', 'YYYY-MM-DD');
    v_dept_id departments.department_id%TYPE := &dept_id;
    v_dept_name departments.department_name%TYPE := '&dept_name';
BEGIN
    INSERT INTO departments (department_id, department_name)
    VALUES (v_dept_id, v_dept_name)
    ON CONFLICT (department_id) DO NOTHING; -- Avoid duplicate department insertions
```

Script Output

Task completed in 28.001 seconds

```
   v_dept_id departments.department_id%TYPE := &dept_id;
   v_dept_name departments.department_name%TYPE := '&dept_name';
EGIN
   INSERT INTO departments (department_id, department_name)
   VALUES (v_dept_id, v_dept_name)
   ON CONFLICT (department_id) DO NOTHING; -- Avoid duplicate department insertions
```

Write a PL/SQL code to find the first employee who has a salary over $2500 and is higher in the chain of command than employee 90. Note: For chain, use of LOOP is Necessary.

SET SERVEROUTPUT ON;

DECLARE v_emp_id employees.employee_id%TYPE := 90; v_mgr_id employees.manager_id%TYPE; v_emp_name employees.first_name%TYPE; v_salary employees.salary%TYPE; found BOOLEAN := FALSE; BEGIN LOOP SELECT manager_id INTO v_mgr_id FROM employees WHERE employee_id = v_emp_id;

```
   EXIT WHEN v_mgr_id IS NULL;

    BEGIN
        SELECT employee_id, first_name, salary
        INTO v_emp_id, v_emp_name, v_salary
        FROM employees
        WHERE employee_id = v_mgr_id AND salary > 2500;

        found := TRUE;
        EXIT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            v_emp_id := v_mgr_id;
    END;
END LOOP;

IF found THEN
    DBMS_OUTPUT.PUT_LINE('First employee with salary over $2500 higher
in the chain: ' || v_emp_name || ' (ID: ' || v_emp_id || ')');
ELSE
    DBMS_OUTPUT.PUT_LINE('No such employee found.');
END IF;
```

END; /

```
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_emp_id employees.employee_id%TYPE := 90;
    v_mgr_id employees.manager_id%TYPE;
    v_emp_name employees.first_name%TYPE;
    v_salary employees.salary%TYPE;
    found BOOLEAN := FALSE;
BEGIN
    LOOP
        SELECT manager_id INTO v_mgr_id FROM employees WHERE employee_id = v_emp_id;

        EXIT WHEN v_mgr_id IS NULL;
```

Script Output

Task completed in 0.037 seconds

```
        WHERE employee_id = v_mgr_id AND salary > 2500;

        found := TRUE;
        EXIT;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            v_emp_id := v_mgr_id;
    END;
END LOOP;
```

15. Write a PL/SQL code to print the sum of first 100 numbers.

SET SERVEROUTPUT ON;

DECLARE v_sum NUMBER := 0; v_counter NUMBER; BEGIN FOR v_counter IN 1..100 LOOP v_sum := v_sum + v_counter; END LOOP;

```
DBMS_OUTPUT.PUT_LINE('The sum of the first 100 numbers is: ' ||
v_sum);
```

END; /

```oracle_plsql
--23P-0573
SET SERVEROUTPUT ON;

DECLARE
    v_sum NUMBER := 0;
    v_counter NUMBER;
BEGIN
    FOR v_counter IN 1..100 LOOP
        v_sum := v_sum + v_counter;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('The sum of the first 100 numbers is: ' || v_sum);
END;
/
```

Script Output

Task completed in 0.026 seconds

```
The sum of the first 100 numbers is: 5050


PL/SQL procedure successfully completed.
```