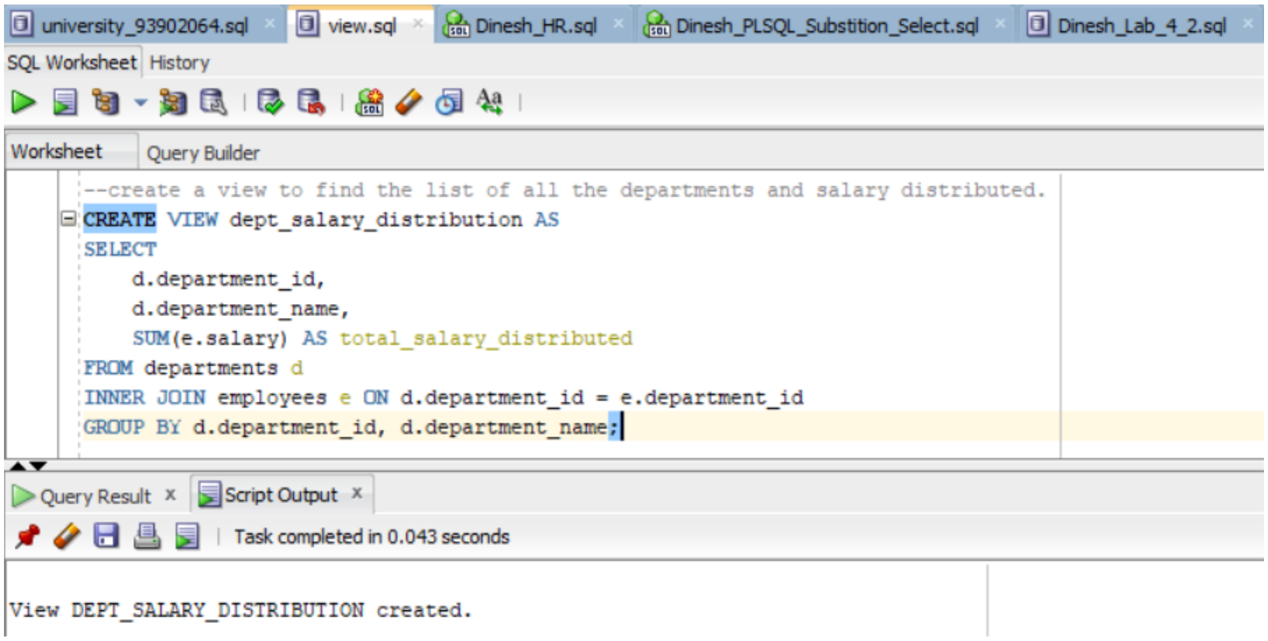
Creating a view

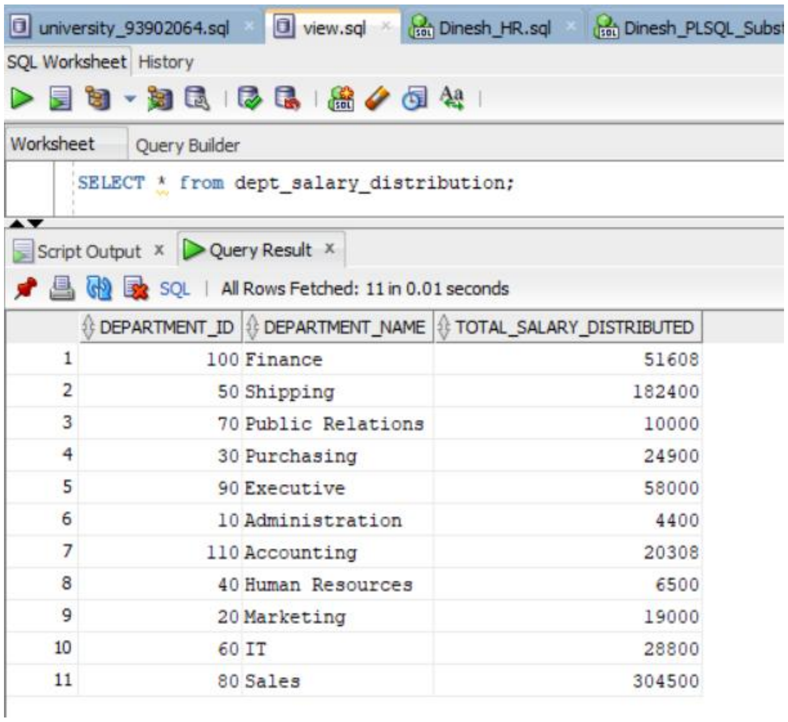


```
--create a view to find the list of all the departments and salary distributed.
CREATE VIEW dept_salary_distribution AS
SELECT
    d.department_id,
    d.department_name,
    SUM(e.salary) AS total_salary_distributed
FROM departments d
INNER JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_id, d.department_name;
```

Task completed in 0.043 seconds

```
View DEPT_SALARY_DISTRIBUTION created.
```

Querying a view

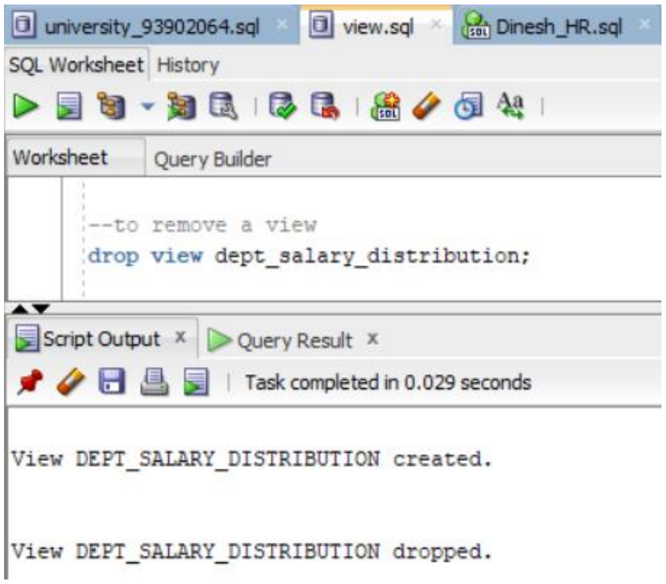


```
SELECT * from dept_salary_distribution;
```

All Rows Fetched: 11 in 0.01 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	TOTAL_SALARY_DISTRIBUTED
1	100	Finance	51608
2	50	Shipping	182400
3	70	Public Relations	10000
4	30	Purchasing	24900
5	90	Executive	58000
6	10	Administration	4400
7	110	Accounting	20308
8	40	Human Resources	6500
9	20	Marketing	19000
10	60	IT	28800
11	80	Sales	304500

Removing/Dropping a view

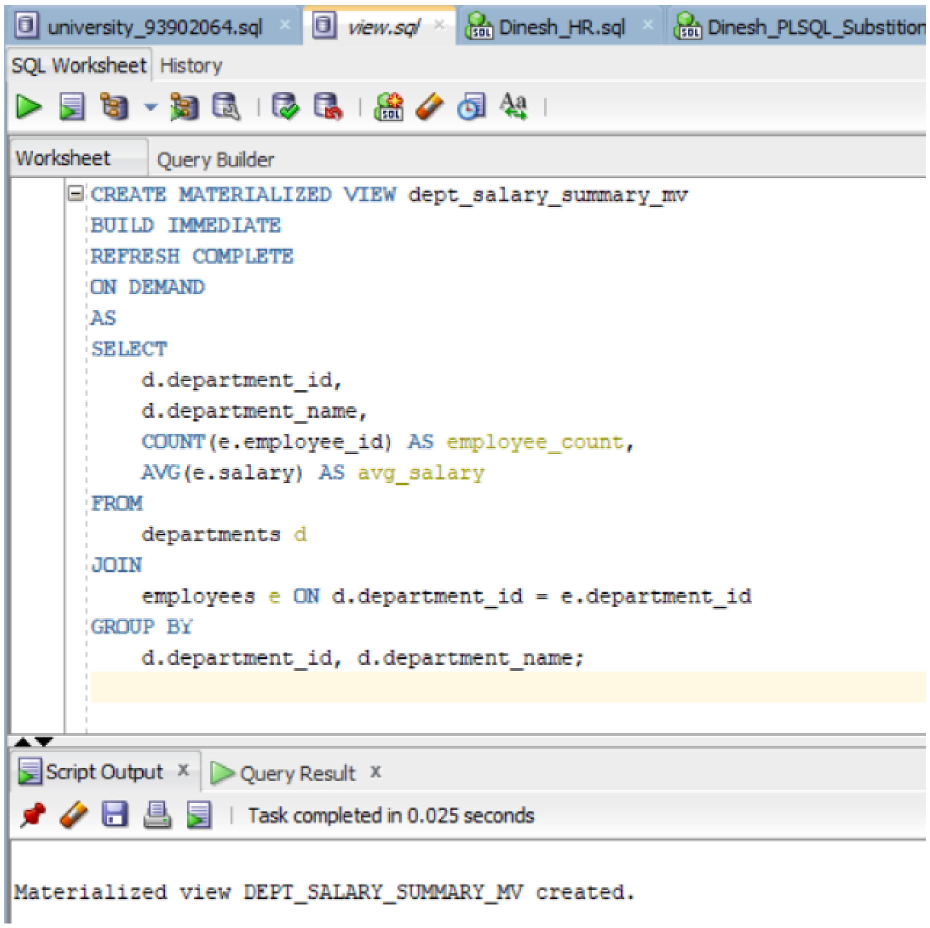


```
--to remove a view
drop view dept_salary_distribution;
```

Task completed in 0.029 seconds

```
View DEPT_SALARY_DISTRIBUTION created.


View DEPT_SALARY_DISTRIBUTION dropped.
```

Create Materialized View

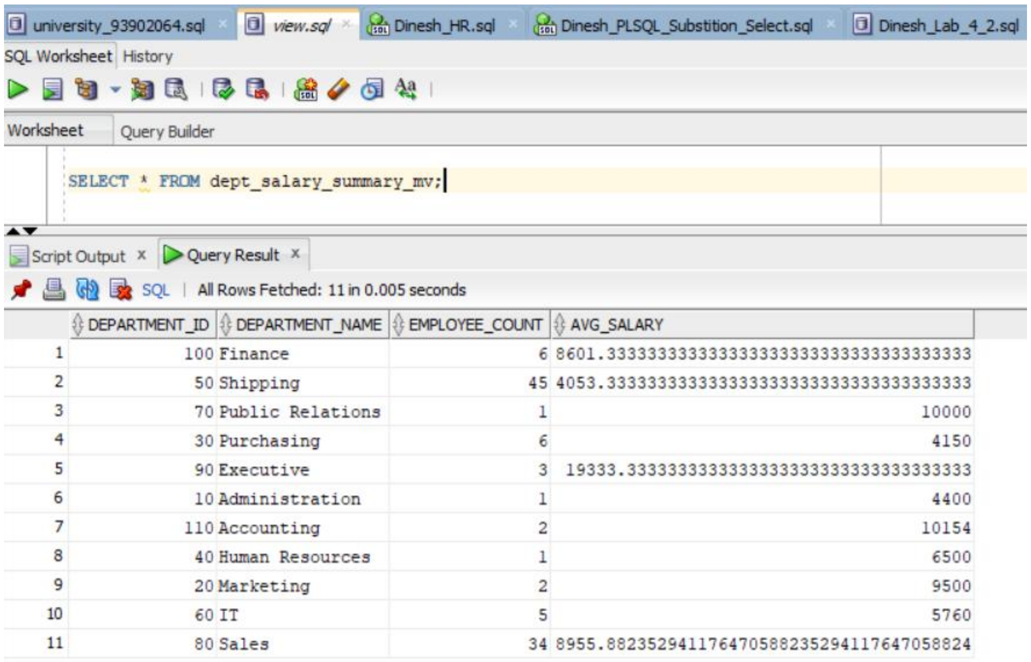


```
CREATE MATERIALIZED VIEW dept_salary_summary_mv
BUILD IMMEDIATE
REFRESH COMPLETE
ON DEMAND
AS
SELECT
    d.department_id,
    d.department_name,
    COUNT(e.employee_id) AS employee_count,
    AVG(e.salary) AS avg_salary
FROM
    departments d
JOIN
    employees e ON d.department_id = e.department_id
GROUP BY
    d.department_id, d.department_name;
```

Task completed in 0.025 seconds

```
Materialized view DEPT_SALARY_SUMMARY_MV created.
```

Querying the Materialized View

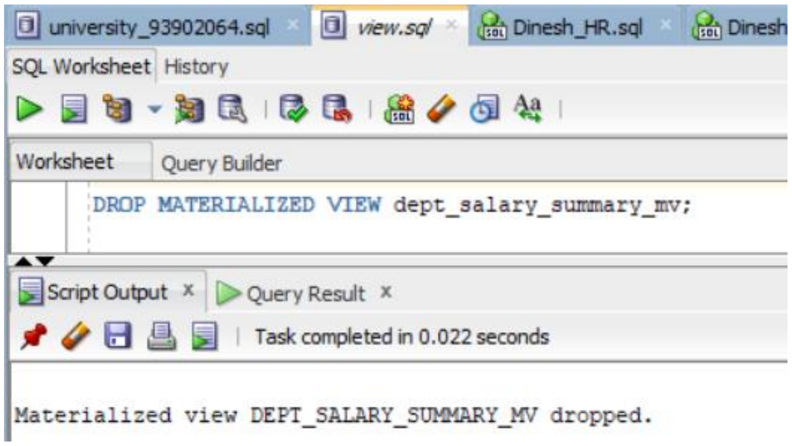


```
SELECT * FROM dept_salary_summary_mv;
```

All Rows Fetched: 11 in 0.005 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEE_COUNT	AVG_SALARY
1	100	Finance	6	8601.333333333333333333333333333333333333
2	50	Shipping	45	4053.333333333333333333333333333333333333
3	70	Public Relations	1	10000
4	30	Purchasing	6	4150
5	90	Executive	3	19333.3333333333333333333333333333333333
6	10	Administration	1	4400
7	110	Accounting	2	10154
8	40	Human Resources	1	6500
9	20	Marketing	2	9500
10	60	IT	5	5760
11	80	Sales	34	8955.882352941176470588235294117647058824

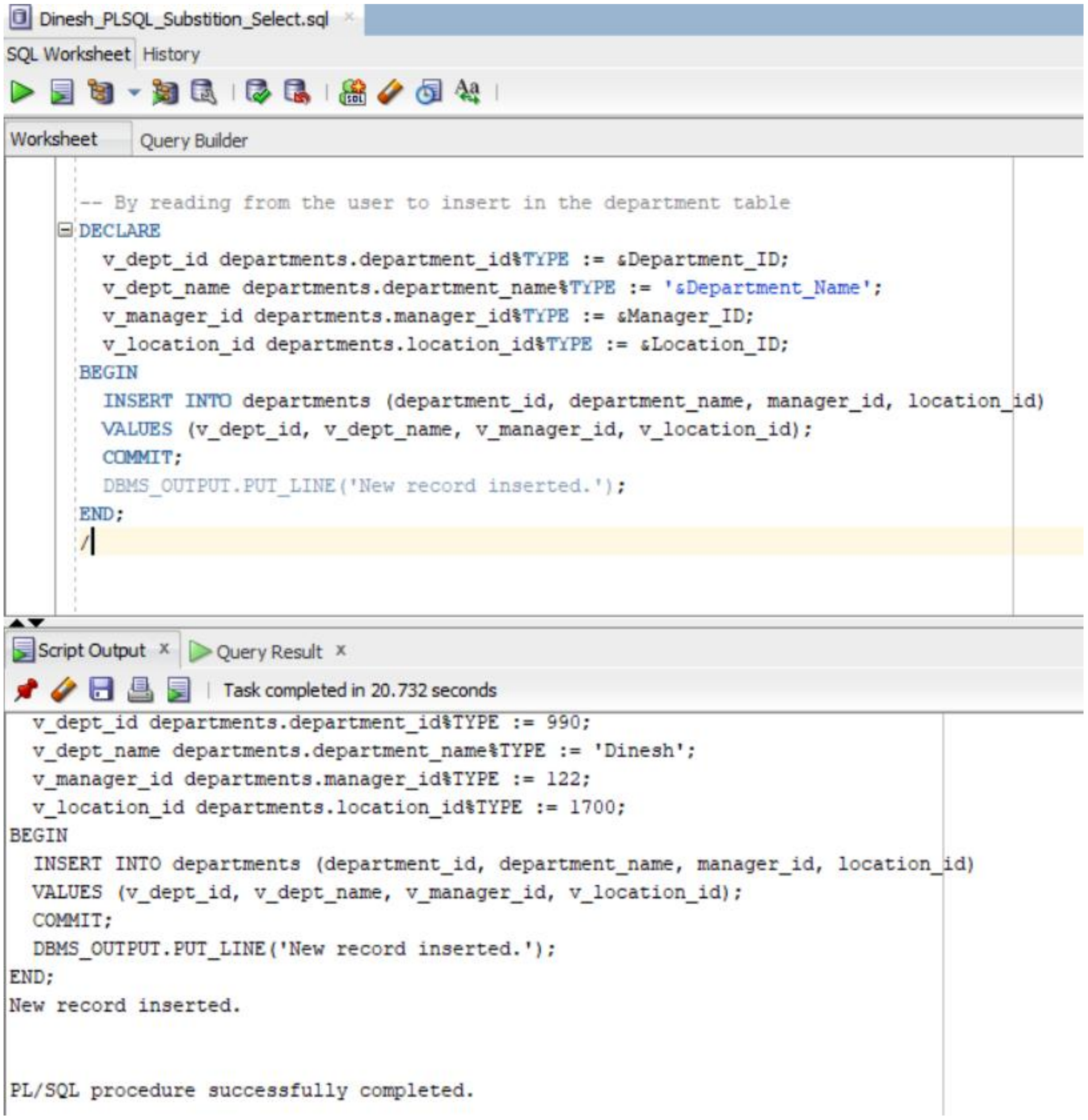
Drop Materialized view



```
DROP MATERIALIZED VIEW dept_salary_summary_mv;
```

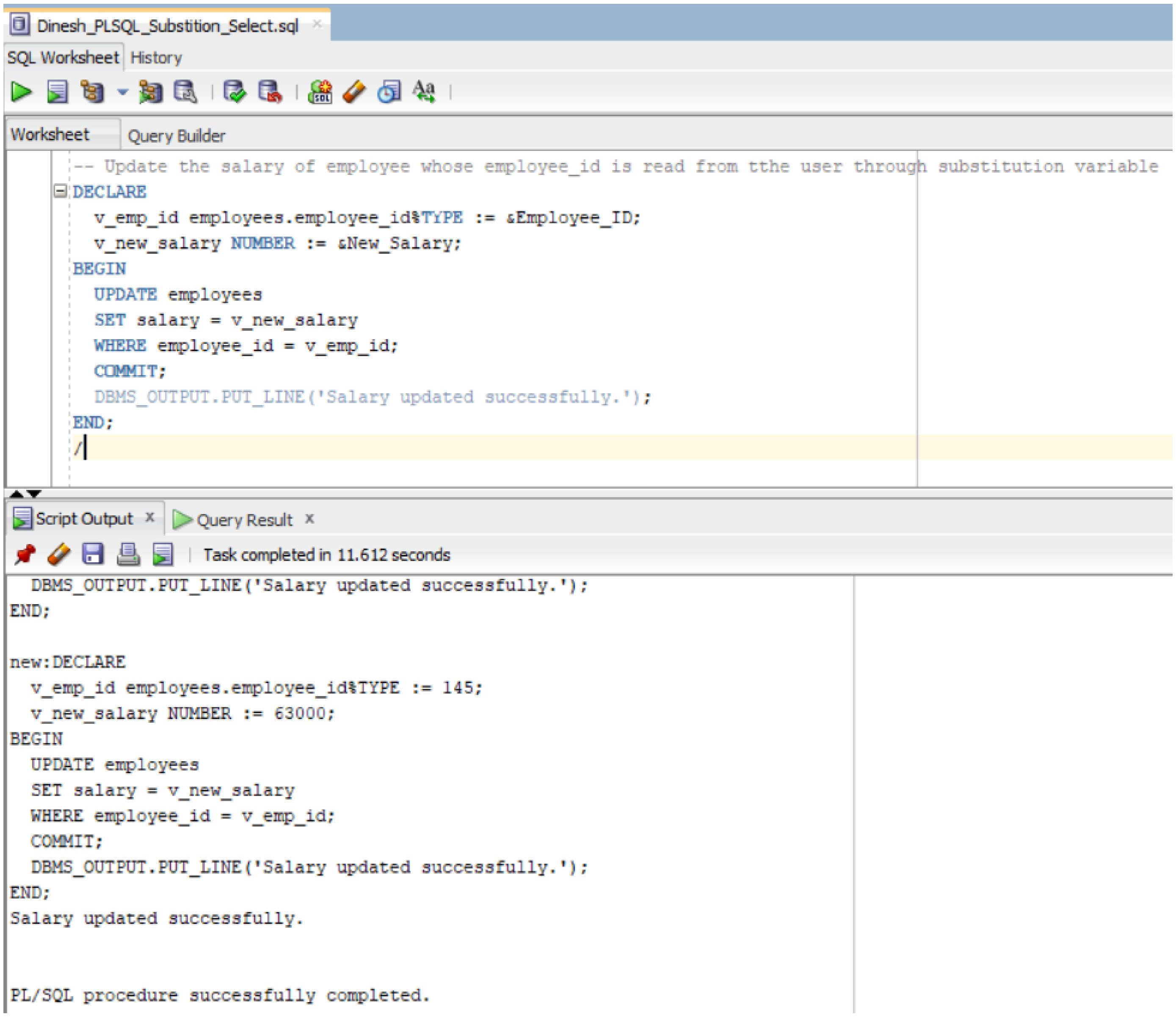
```
Materialized view DEPT_SALARY_SUMMARY_MV dropped.
```

Lab 4.1



```
-- By reading from the user to insert in the department table
DECLARE
  v_dept_id departments.department_id%TYPE := &Department_ID;
  v_dept_name departments.department_name%TYPE := '&Department_Name';
  v_manager_id departments.manager_id%TYPE := &Manager_ID;
  v_location_id departments.location_id%TYPE := &Location_ID;
BEGIN
  INSERT INTO departments (department_id, department_name, manager_id, location_id)
  VALUES (v_dept_id, v_dept_name, v_manager_id, v_location_id);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('New record inserted.');
END;
/
```

```
  v_dept_id departments.department_id%TYPE := 990;
  v_dept_name departments.department_name%TYPE := 'Dinesh';
  v_manager_id departments.manager_id%TYPE := 122;
  v_location_id departments.location_id%TYPE := 1700;
BEGIN
  INSERT INTO departments (department_id, department_name, manager_id, location_id)
  VALUES (v_dept_id, v_dept_name, v_manager_id, v_location_id);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('New record inserted.');
END;
New record inserted.


PL/SQL procedure successfully completed.
```

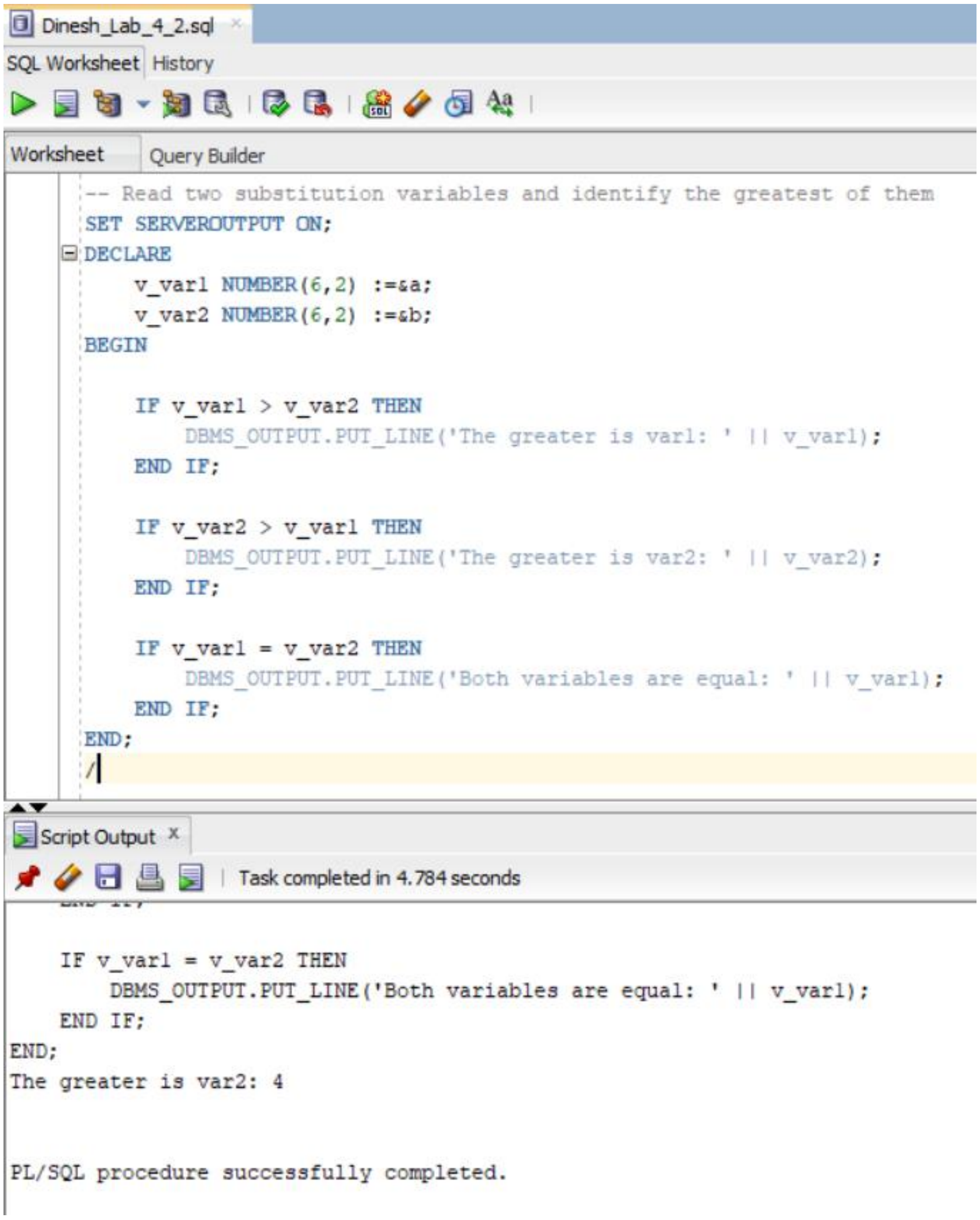

Dinesh_PLSQL_Substition_Select.sql

```
-- Update the salary of employee whose employee_id is read from tthe user through substitution variable
DECLARE
  v_emp_id employees.employee_id%TYPE := &Employee_ID;
  v_new_salary NUMBER := &New_Salary;
BEGIN
  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = v_emp_id;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');
END;
/
```

Task completed in 11.612 seconds

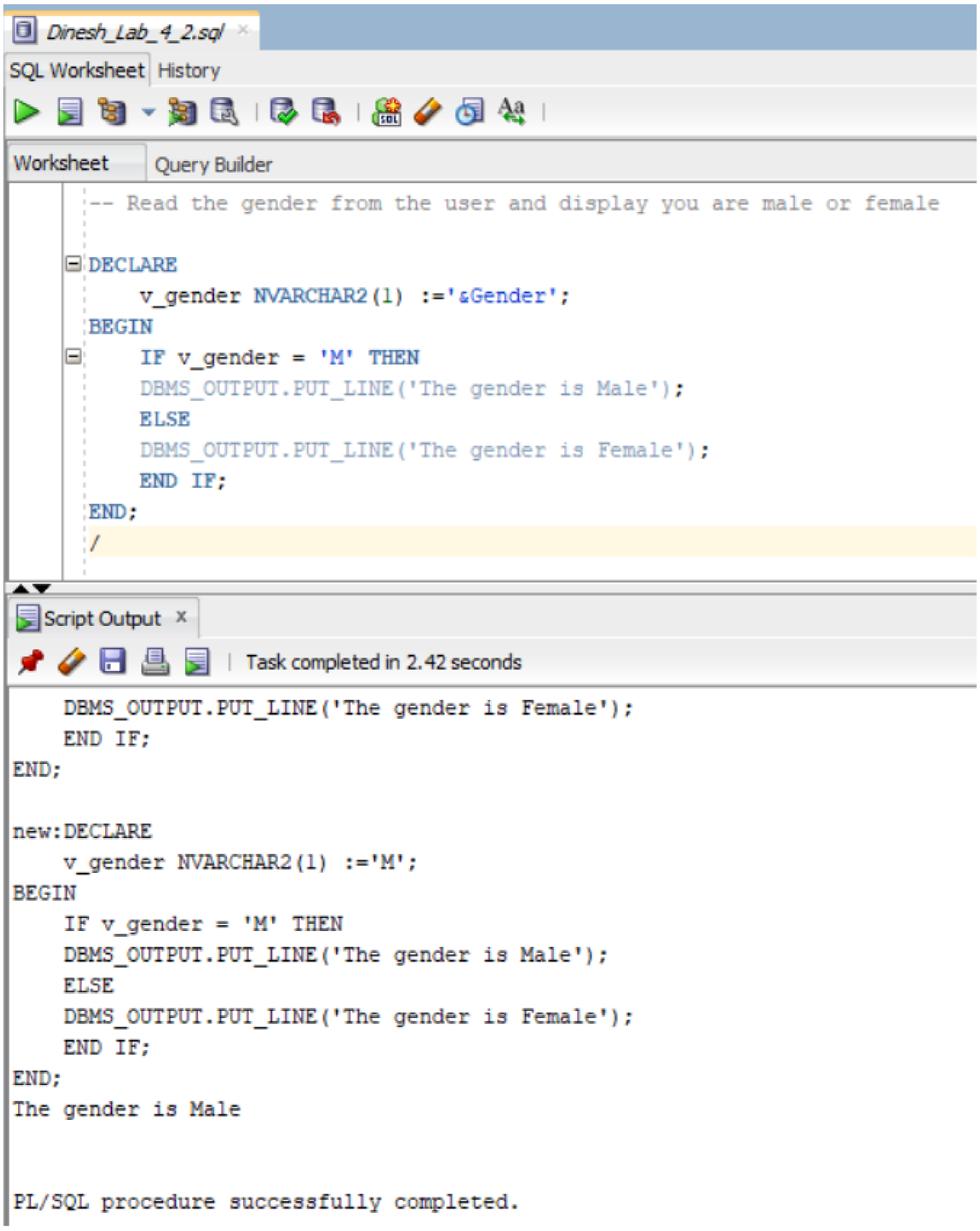
```
  DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');
END;

new:DECLARE
  v_emp_id employees.employee_id%TYPE := 145;
  v_new_salary NUMBER := 63000;
BEGIN
  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = v_emp_id;
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Salary updated successfully.');
END;
Salary updated successfully.


PL/SQL procedure successfully completed.
```

Lab 4.2



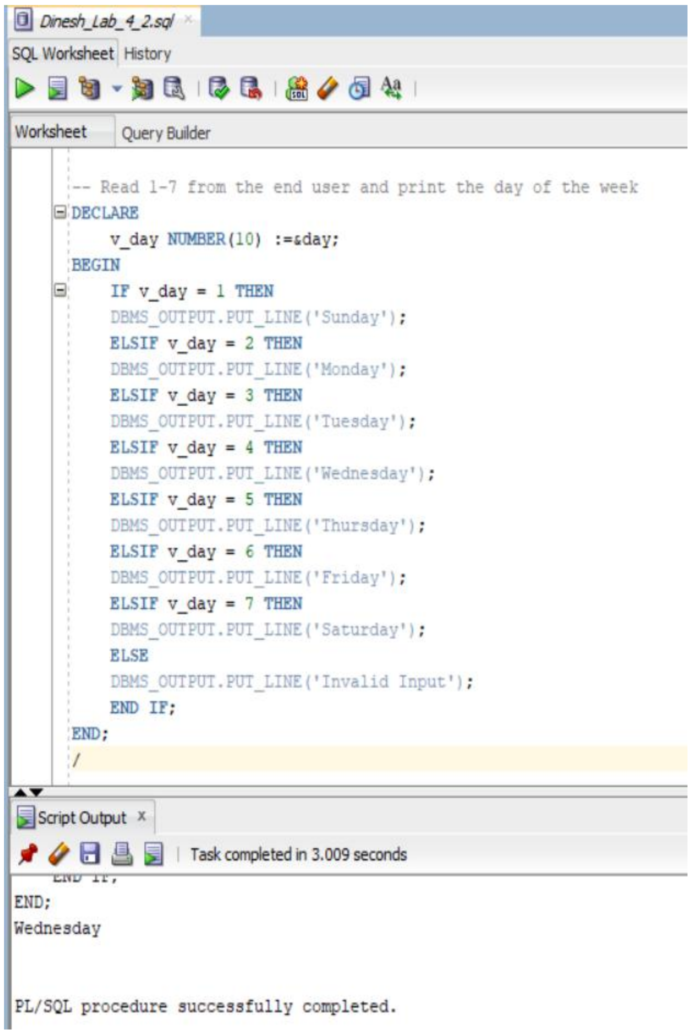
```
-- Read two substitution variables and identify the greatest of them
SET SERVEROUTPUT ON;
DECLARE
    v_var1 NUMBER(6,2) :=&a;
    v_var2 NUMBER(6,2) :=&b;
BEGIN

    IF v_var1 > v_var2 THEN
        DBMS_OUTPUT.PUT_LINE('The greater is var1: ' || v_var1);
    END IF;

    IF v_var2 > v_var1 THEN
        DBMS_OUTPUT.PUT_LINE('The greater is var2: ' || v_var2);
    END IF;

    IF v_var1 = v_var2 THEN
        DBMS_OUTPUT.PUT_LINE('Both variables are equal: ' || v_var1);
    END IF;
END;
/
```

```
    IF v_var1 = v_var2 THEN
        DBMS_OUTPUT.PUT_LINE('Both variables are equal: ' || v_var1);
    END IF;
END;
The greater is var2: 4


PL/SQL procedure successfully completed.
```



```
-- Read the gender from the user and display you are male or female

DECLARE
    v_gender NVARCHAR2(1) :='&Gender';
BEGIN
    IF v_gender = 'M' THEN
    DBMS_OUTPUT.PUT_LINE('The gender is Male');
    ELSE
    DBMS_OUTPUT.PUT_LINE('The gender is Female');
    END IF;
END;
/
```

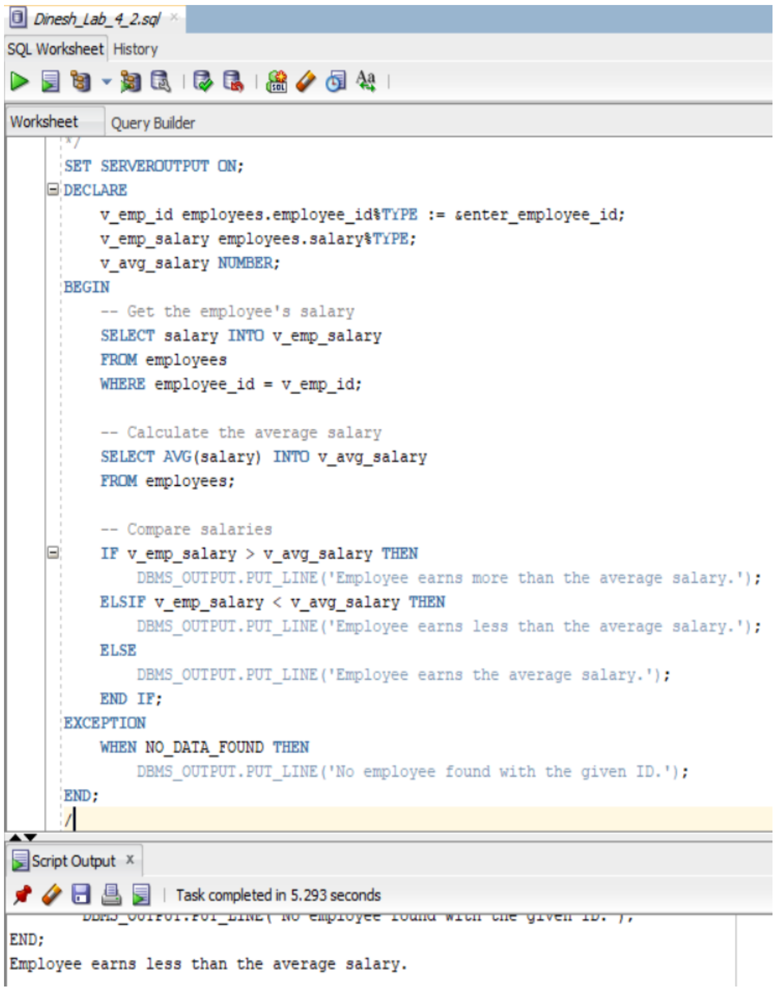
```
    DBMS_OUTPUT.PUT_LINE('The gender is Female');
    END IF;
END;

new:DECLARE
    v_gender NVARCHAR2(1) :='M';
BEGIN
    IF v_gender = 'M' THEN
    DBMS_OUTPUT.PUT_LINE('The gender is Male');
    ELSE
    DBMS_OUTPUT.PUT_LINE('The gender is Female');
    END IF;
END;
The gender is Male


PL/SQL procedure successfully completed.
```



```
-- Read 1-7 from the end user and print the day of the week
DECLARE
    v_day NUMBER(10) :=&day;
BEGIN
    IF v_day = 1 THEN
    DBMS_OUTPUT.PUT_LINE('Sunday');
    ELSIF v_day = 2 THEN
    DBMS_OUTPUT.PUT_LINE('Monday');
    ELSIF v_day = 3 THEN
    DBMS_OUTPUT.PUT_LINE('Tuesday');
    ELSIF v_day = 4 THEN
    DBMS_OUTPUT.PUT_LINE('Wednesday');
    ELSIF v_day = 5 THEN
    DBMS_OUTPUT.PUT_LINE('Thursday');
    ELSIF v_day = 6 THEN
    DBMS_OUTPUT.PUT_LINE('Friday');
    ELSIF v_day = 7 THEN
    DBMS_OUTPUT.PUT_LINE('Saturday');
    ELSE
    DBMS_OUTPUT.PUT_LINE('Invalid Input');
    END IF;
END;
/
```

Task completed in 3.009 seconds

```
    END IF;
END;
Wednesday


PL/SQL procedure successfully completed.
```

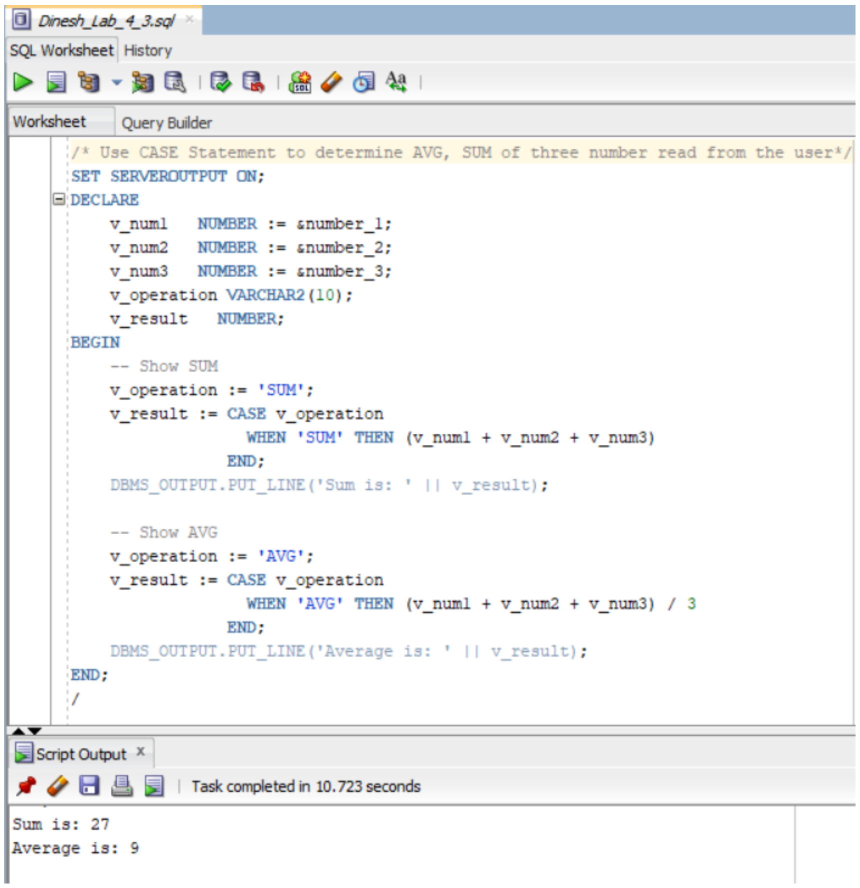


```
*/
SET SERVEROUTPUT ON;
DECLARE
    v_emp_id employees.employee_id%TYPE := &enter_employee_id;
    v_emp_salary employees.salary%TYPE;
    v_avg_salary NUMBER;
BEGIN
    -- Get the employee's salary
    SELECT salary INTO v_emp_salary
    FROM employees
    WHERE employee_id = v_emp_id;

    -- Calculate the average salary
    SELECT AVG(salary) INTO v_avg_salary
    FROM employees;

    -- Compare salaries
    IF v_emp_salary > v_avg_salary THEN
        DBMS_OUTPUT.PUT_LINE('Employee earns more than the average salary.');
    ELSIF v_emp_salary < v_avg_salary THEN
        DBMS_OUTPUT.PUT_LINE('Employee earns less than the average salary.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Employee earns the average salary.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with the given ID.');
END;
/
```

Task completed in 5.293 seconds

```
    DBMS_OUTPUT.PUT_LINE('No employee found with the given ID.');
END;
Employee earns less than the average salary.
```

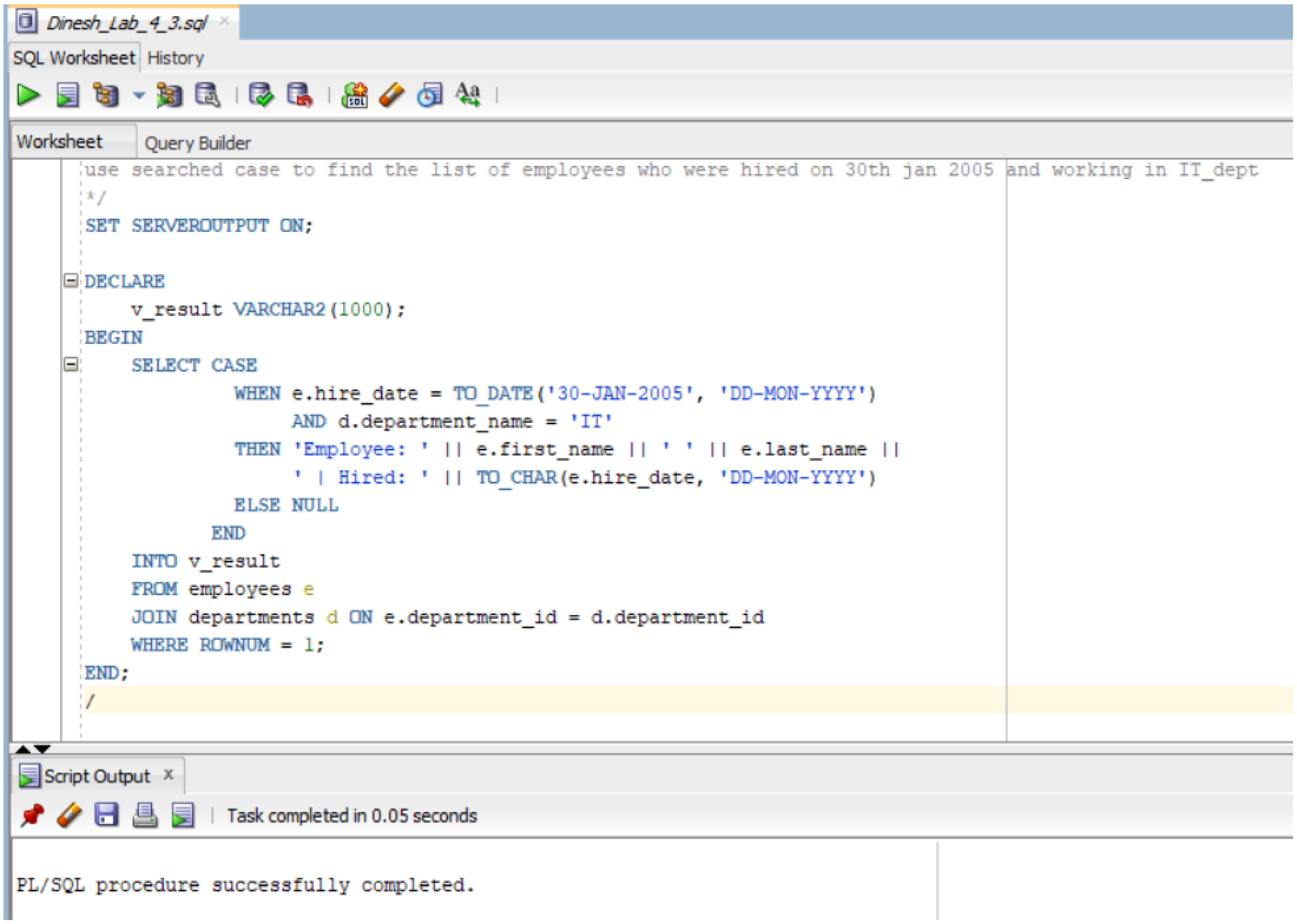
Lab 4.3



```
/* Use CASE Statement to determine AVG, SUM of three number read from the user*/
SET SERVEROUTPUT ON;
DECLARE
    v_num1    NUMBER := &number_1;
    v_num2    NUMBER := &number_2;
    v_num3    NUMBER := &number_3;
    v_operation VARCHAR2(10);
    v_result    NUMBER;
BEGIN
    -- Show SUM
    v_operation := 'SUM';
    v_result := CASE v_operation
                  WHEN 'SUM' THEN (v_num1 + v_num2 + v_num3)
                END;
    DBMS_OUTPUT.PUT_LINE('Sum is: ' || v_result);

    -- Show AVG
    v_operation := 'AVG';
    v_result := CASE v_operation
                  WHEN 'AVG' THEN (v_num1 + v_num2 + v_num3) / 3
                END;
    DBMS_OUTPUT.PUT_LINE('Average is: ' || v_result);
END;
/
```

Task completed in 10.723 seconds

```
Sum is: 27
Average is: 9
```


Dinesh_Lab_4_3.sql

SQL Worksheet History

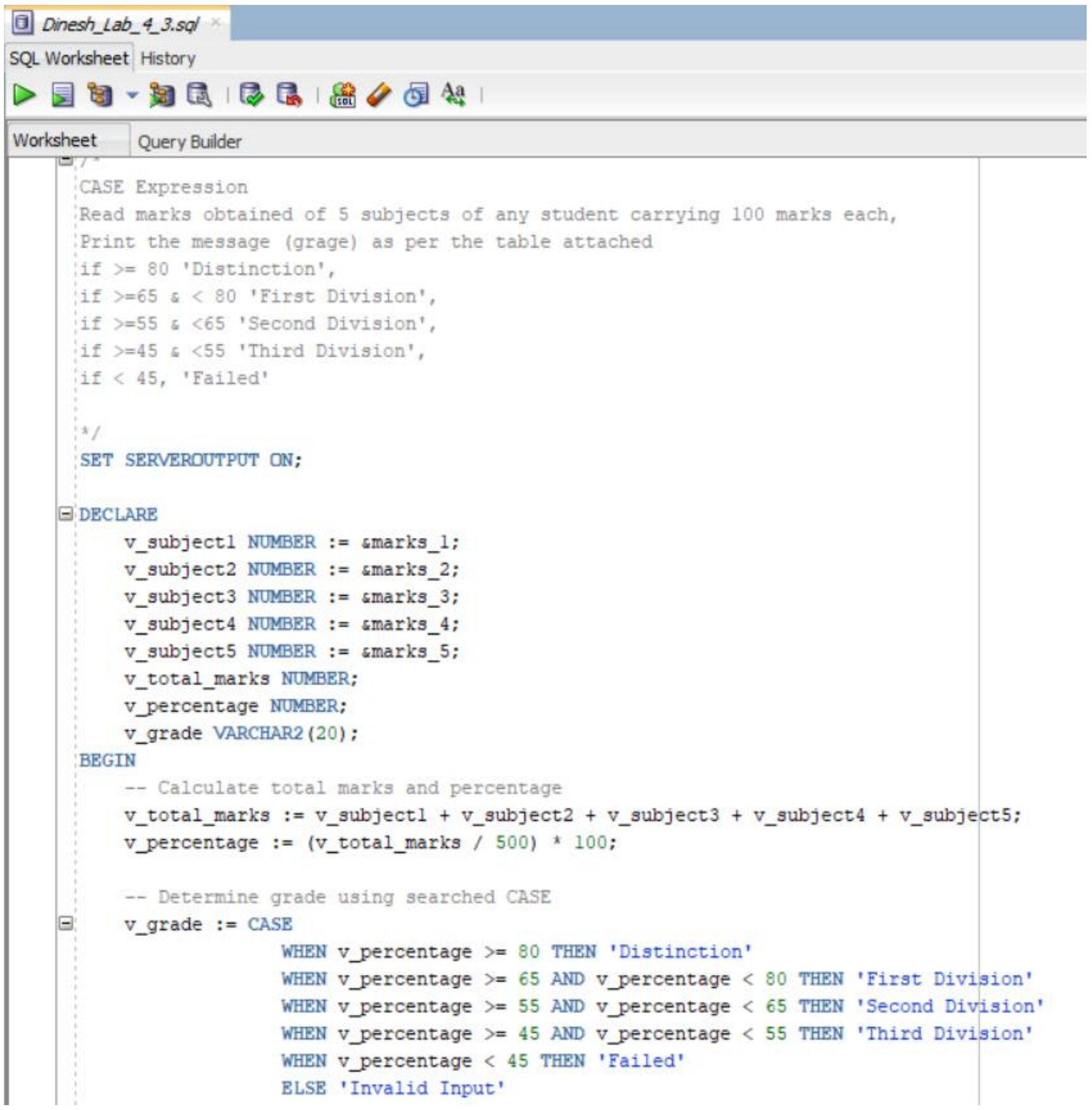
Worksheet Query Builder

```
use searched case to find the list of employees who were hired on 30th jan 2005 and working in IT_dept
*/
SET SERVEROUTPUT ON;

DECLARE
    v_result VARCHAR2(1000);
BEGIN
    SELECT CASE
                WHEN e.hire_date = TO_DATE('30-JAN-2005', 'DD-MON-YYYY')
                     AND d.department_name = 'IT'
                THEN 'Employee: ' || e.first_name || ' ' || e.last_name ||
                     ' | Hired: ' || TO_CHAR(e.hire_date, 'DD-MON-YYYY')
                ELSE NULL
              END
    INTO v_result
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    WHERE ROWNUM = 1;
END;
/
```

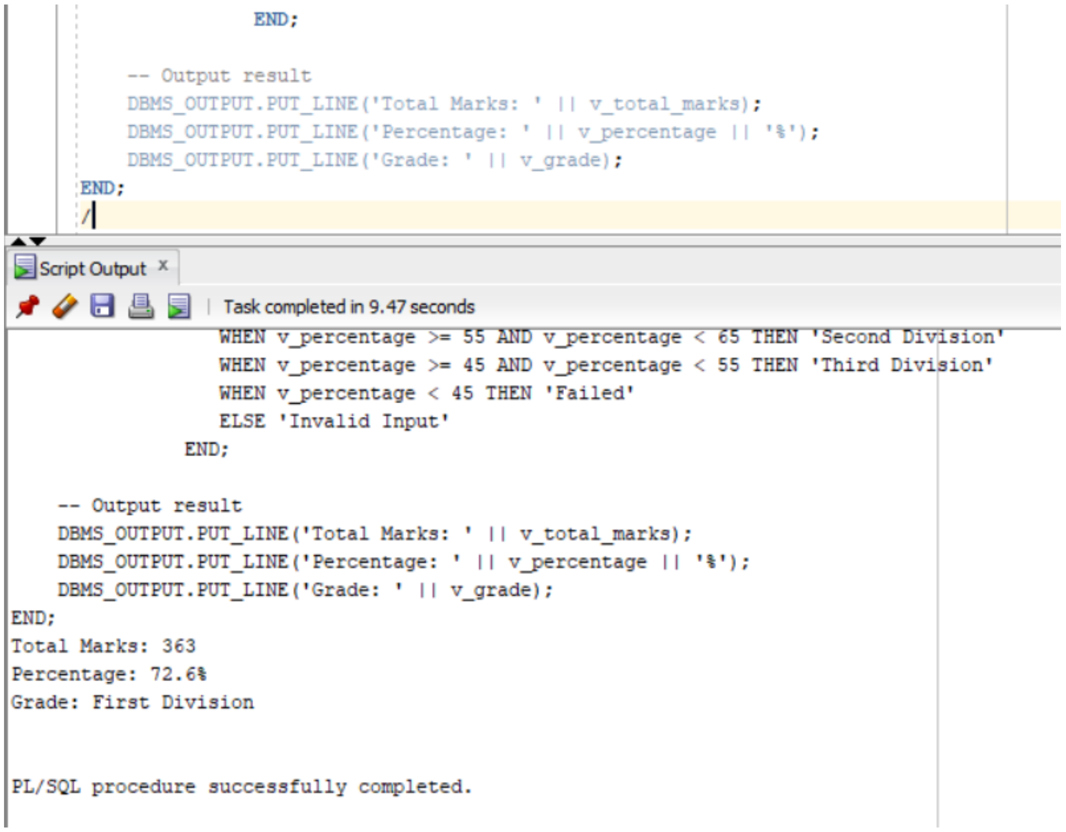
Script Output

Task completed in 0.05 seconds

```
PL/SQL procedure successfully completed.
```

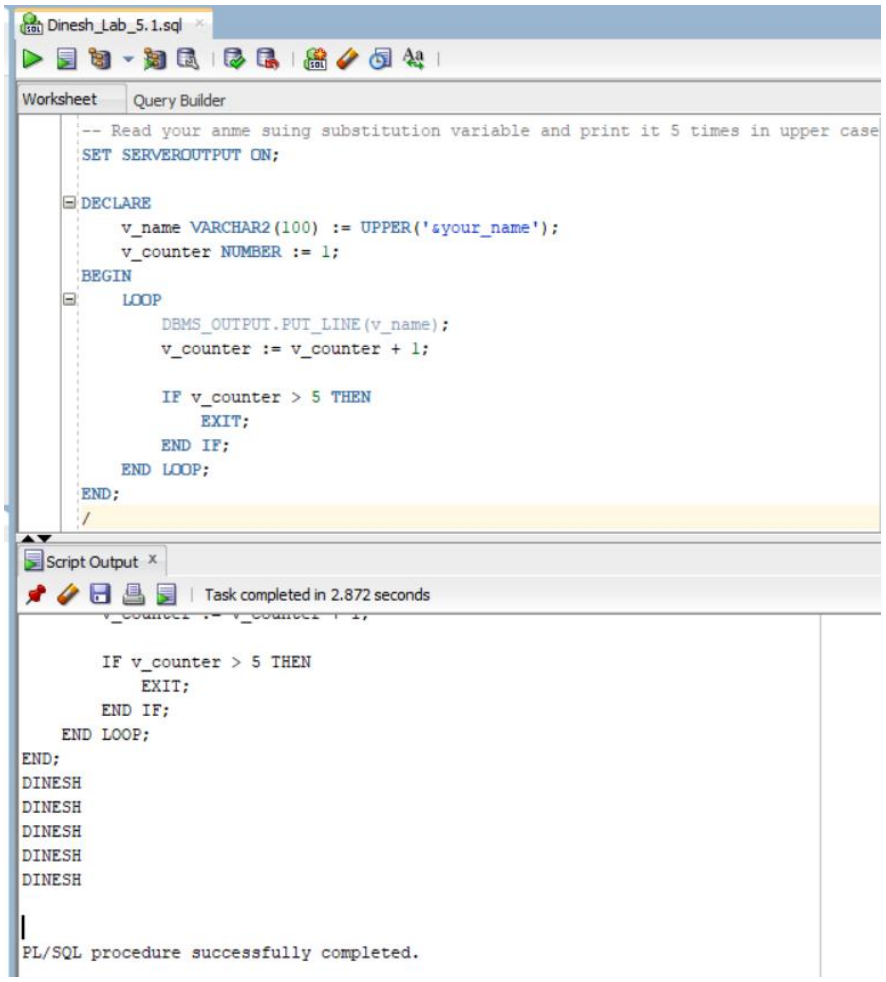


```
/*
CASE Expression
Read marks obtained of 5 subjects of any student carrying 100 marks each,
Print the message (grage) as per the table attached
if >= 80 'Distinction',
if >=65 & < 80 'First Division',
if >=55 & <65 'Second Division',
if >=45 & <55 'Third Division',
if < 45, 'Failed'

*/
SET SERVEROUTPUT ON;

DECLARE
    v_subject1 NUMBER := &marks_1;
    v_subject2 NUMBER := &marks_2;
    v_subject3 NUMBER := &marks_3;
    v_subject4 NUMBER := &marks_4;
    v_subject5 NUMBER := &marks_5;
    v_total_marks NUMBER;
    v_percentage NUMBER;
    v_grade VARCHAR2(20);
BEGIN
    -- Calculate total marks and percentage
    v_total_marks := v_subject1 + v_subject2 + v_subject3 + v_subject4 + v_subject5;
    v_percentage := (v_total_marks / 500) * 100;

    -- Determine grade using searched CASE
    v_grade := CASE
                    WHEN v_percentage >= 80 THEN 'Distinction'
                    WHEN v_percentage >= 65 AND v_percentage < 80 THEN 'First Division'
                    WHEN v_percentage >= 55 AND v_percentage < 65 THEN 'Second Division'
                    WHEN v_percentage >= 45 AND v_percentage < 55 THEN 'Third Division'
                    WHEN v_percentage < 45 THEN 'Failed'
                    ELSE 'Invalid Input'
```



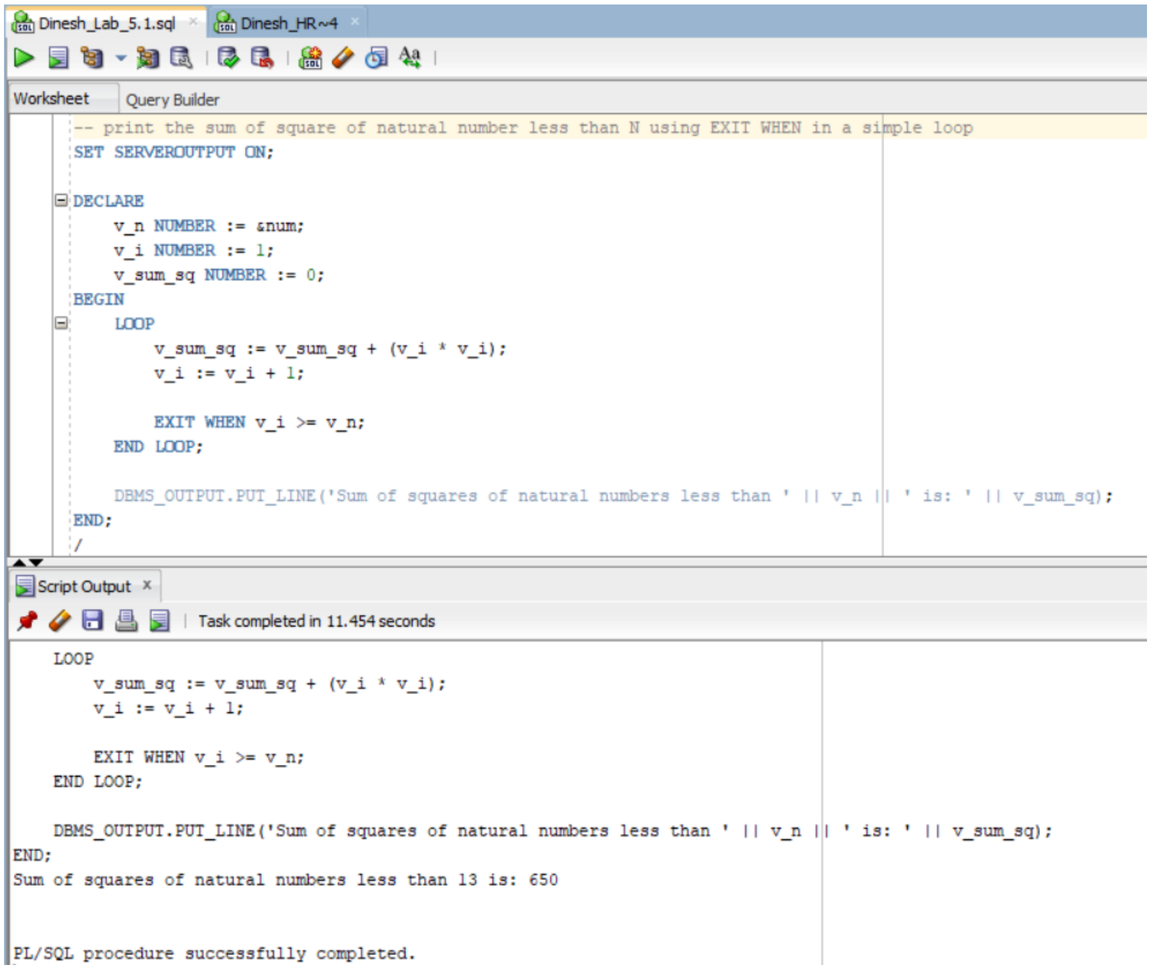
```
                END;

    -- Output result
    DBMS_OUTPUT.PUT_LINE('Total Marks: ' || v_total_marks);
    DBMS_OUTPUT.PUT_LINE('Percentage: ' || v_percentage || '%');
    DBMS_OUTPUT.PUT_LINE('Grade: ' || v_grade);
END;
/
```

```
                WHEN v_percentage >= 55 AND v_percentage < 65 THEN 'Second Division'
                WHEN v_percentage >= 45 AND v_percentage < 55 THEN 'Third Division'
                WHEN v_percentage < 45 THEN 'Failed'
                ELSE 'Invalid Input'
             END;

    -- Output result
    DBMS_OUTPUT.PUT_LINE('Total Marks: ' || v_total_marks);
    DBMS_OUTPUT.PUT_LINE('Percentage: ' || v_percentage || '%');
    DBMS_OUTPUT.PUT_LINE('Grade: ' || v_grade);
END;
Total Marks: 363
Percentage: 72.6%
Grade: First Division


PL/SQL procedure successfully completed.
```



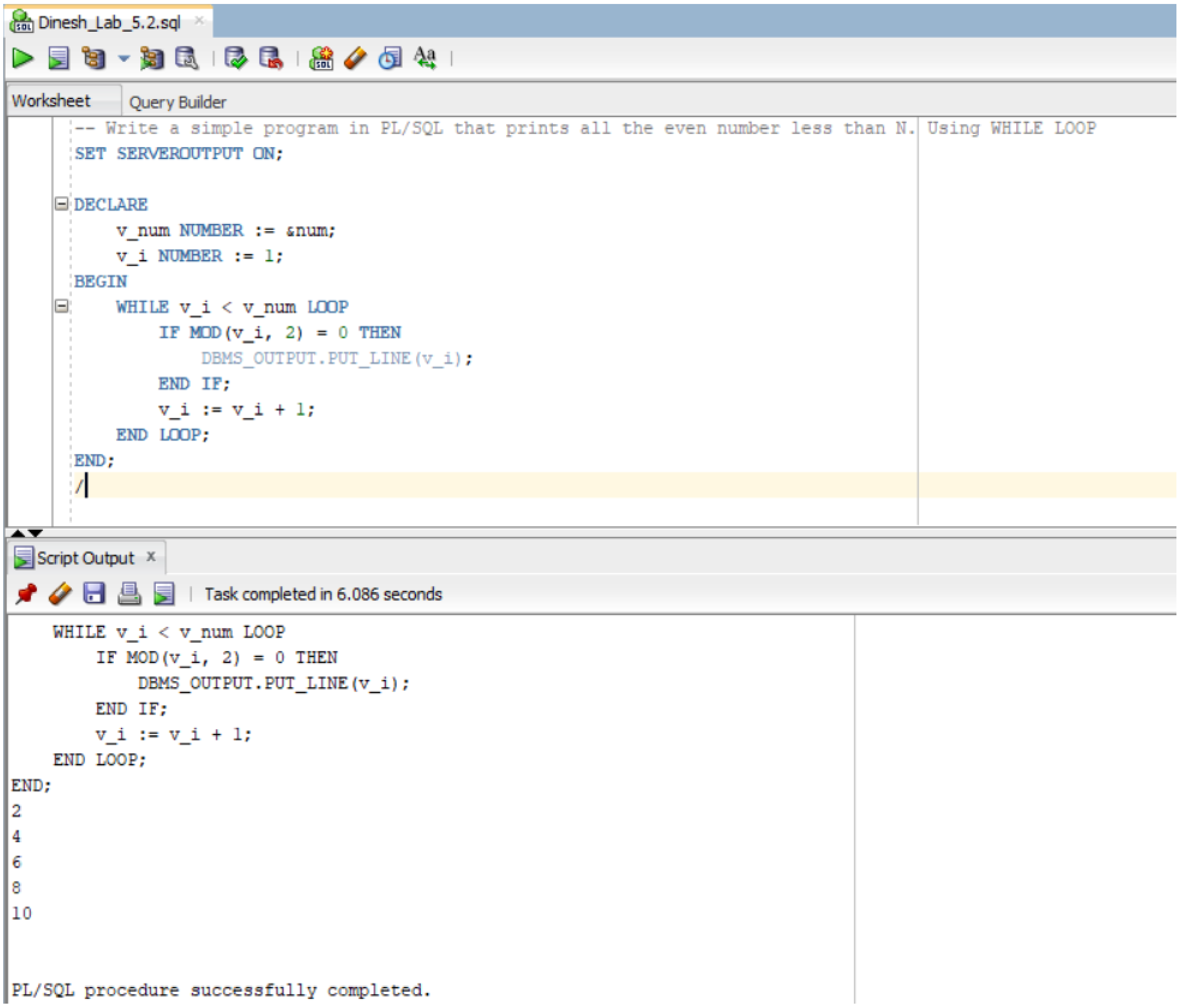
```
-- Read your anme suing substitution variable and print it 5 times in upper case
SET SERVEROUTPUT ON;

DECLARE
    v_name VARCHAR2(100) := UPPER('&your_name');
    v_counter NUMBER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_name);
        v_counter := v_counter + 1;

        IF v_counter > 5 THEN
            EXIT;
        END IF;
    END LOOP;
END;
/
```

```

        IF v_counter > 5 THEN
            EXIT;
        END IF;
    END LOOP;
END;
DINESH
DINESH
DINESH
DINESH
DINESH


PL/SQL procedure successfully completed.
```



```
-- print the sum of square of natural number less than N using EXIT WHEN in a simple loop
SET SERVEROUTPUT ON;

DECLARE
    v_n NUMBER := &num;
    v_i NUMBER := 1;
    v_sum_sq NUMBER := 0;
BEGIN
    LOOP
        v_sum_sq := v_sum_sq + (v_i * v_i);
        v_i := v_i + 1;

        EXIT WHEN v_i >= v_n;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Sum of squares of natural numbers less than ' || v_n || ' is: ' || v_sum_sq);
END;
/
```

```
    LOOP
        v_sum_sq := v_sum_sq + (v_i * v_i);
        v_i := v_i + 1;

        EXIT WHEN v_i >= v_n;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Sum of squares of natural numbers less than ' || v_n || ' is: ' || v_sum_sq);
END;
Sum of squares of natural numbers less than 13 is: 650


PL/SQL procedure successfully completed.
```



```
-- Write a simple program in PL/SQL that prints all the even number less than N. Using WHILE LOOP
SET SERVEROUTPUT ON;

DECLARE
    v_num NUMBER := &num;
    v_i NUMBER := 1;
BEGIN
    WHILE v_i < v_num LOOP
        IF MOD(v_i, 2) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(v_i);
        END IF;
        v_i := v_i + 1;
    END LOOP;
END;
/
```

```
    WHILE v_i < v_num LOOP
        IF MOD(v_i, 2) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(v_i);
        END IF;
        v_i := v_i + 1;
    END LOOP;
END;
2
4
6
8
10


PL/SQL procedure successfully completed.
```