**PL/SQL Lab Report  
LAB – 9**

**Student Name : AHNAF SHAHRIAR PIAS**

**Student ID : 220042146**

**Task 1.1: Display "Hello World"**

**Code:**

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

/

**Output:**

Hello World

**Explanation:**

This task demonstrated the basic structure of a PL/SQL block and the use of DBMS\_OUTPUT.PUT\_LINE to print messages.

**Task 1.2: Declare and Display Variables**

**Code:**

DECLARE

my\_name VARCHAR2(50) := 'Ahnaf';

my\_id NUMBER := 101;

friend\_name VARCHAR2(50) := 'Pias';

friend\_id NUMBER := 102;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('My Name :' || my\_name || ', ID: ' || my\_id);

DBMS\_OUTPUT.PUT\_LINE('Friend Name: ' || friend\_name || ', ID: ' || friend\_id);

END;

/

**Output:**

My Name : Ahnaf, ID: 101

Friend Name: Pias, ID: 102

**Explanation:**

This task introduced variable declaration and string concatenation using the || operator.

**Task 1.3: Perform Arithmetic Operations**

**Code:**

DECLARE

num1 NUMBER := 10;

num2 NUMBER := 5;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Addition : ' || (num1 + num2));

DBMS\_OUTPUT.PUT\_LINE('Subtraction : ' || (num1 - num2));

DBMS\_OUTPUT.PUT\_LINE('Multiplication: ' || (num1 \* num2));

DBMS\_OUTPUT.PUT\_LINE('Division : ' || (num1 / num2));

END;

/

**Output:**

Addition : 15

Subtraction : 5

Multiplication: 50

Division : 2

**Explanation:**

The program demonstrated mathematical operations using PL/SQL variables.

**Task 1.4: Determine Grade Based on Percentage**

**Code:**

DECLARE

mark\_percentage NUMBER := 85;

grade VARCHAR2(10);

BEGIN

IF mark\_percentage >= 90 THEN

grade := 'A';

ELSIF mark\_percentage >= 80 THEN

grade := 'B';

ELSIF mark\_percentage >= 70 THEN

grade := 'C';

ELSIF mark\_percentage >= 60 THEN

grade := 'D';

ELSE

grade := 'F';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Grade: ' || grade);

END;

/

**Output:**

Grade: B

**Explanation:**

This task illustrated the use of IF-ELSE conditions for decision-making.

**Task 1.5: Print 68 Student IDs Using a While Loop**

**Code:**

DECLARE

student\_id NUMBER := 101;

counter NUMBER := 1;

BEGIN

WHILE counter <= 68 LOOP

DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || student\_id);

student\_id := student\_id + 1;

counter := counter + 1;

END LOOP;

END;

/

**Output:**

Student ID: 101

Student ID: 102

...

Student ID: 168

**Explanation:**

This task demonstrated the use of loops to execute repetitive tasks efficiently.

**Task 2: Determine Decade Start Year**

**Code:**

DECLARE

current\_year NUMBER := 2024;

decade\_start NUMBER;

BEGIN

IF MOD(current\_year, 10) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Yes');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No');

END IF;

decade\_start := current\_year - MOD(current\_year, 10);

DBMS\_OUTPUT.PUT\_LINE('The ' || decade\_start || 's');

END;

/

**Output:**

No

The 2020s

**Explanation:**

This task introduced modular arithmetic and string concatenation.

**Task 3: Sum of Prime Numbers Until 20**

**Code:**

DECLARE

num NUMBER := 2;

sum\_primes NUMBER := 0;

is\_prime BOOLEAN;

i NUMBER;

BEGIN

WHILE sum\_primes + num <= 20 LOOP

is\_prime := TRUE;

FOR i IN 2..TRUNC(SQRT(num)) LOOP

IF MOD(num, i) = 0 THEN

is\_prime := FALSE;

EXIT;

END IF;

END LOOP;

IF is\_prime THEN

DBMS\_OUTPUT.PUT\_LINE(num);

sum\_primes := sum\_primes + num;

END IF;

num := num + 1;

END LOOP;

END;

/

**Output:**

2

3

5

7

**Explanation:**

This task involved loop control structures and prime number identification using the square root method.