Practical-10

Aim : PL/SQL Block:

1. Write a PL/SQL Block to Add 2 Numbers
2. Write a PL/SQL Block to find Area of Rectangle, Triangle and Square.
3. Write a PL/SQL Block to find Maximum of 3 numbers
4. Write a PL/SQL Block to print sum of N Numbers using For Loop.
5. Write a PL/SQL Block to generate Fibonacci series of N numbers

Note : Make sure to adjust the table and column names based on your database schema.

1)

```
DECLARE
  num1 NUMBER := 10;
  num2 NUMBER := 20;
  result NUMBER;
BEGIN
  result := num1 + num2;
  DBMS_OUTPUT.PUT_LINE('The sum of ' || num1 || ' and ' || num2 || ' is ' || result);
END;
/
```

In the above PL/SQL block, we declare two variables **num1** and **num2** and assign them the values 10 and 20, respectively. Then, we declare a variable **result** to store the sum of the two numbers.

Inside the **BEGIN** and **END** block, we perform the addition operation **num1 + num2** and store the result in the **result** variable. Finally, we use the **DBMS_OUTPUT.PUT_LINE** function to display the sum in the console output.

To execute the PL/SQL block, you can run it in a PL/SQL editor or in an Oracle SQL Developer tool that supports PL/SQL execution.

2)

```
DECLARE
  length NUMBER := 5;
  width NUMBER := 3;
  base NUMBER := 4;
  height NUMBER := 6;
  side NUMBER := 2;
  rectangle_area NUMBER;
  triangle_area NUMBER;
```

```
  square_area NUMBER;
BEGIN
  rectangle_area := length * width;
  triangle_area := 0.5 * base * height;
  square_area := side * side;

  DBMS_OUTPUT.PUT_LINE('Area of Rectangle: ' || rectangle_area);
  DBMS_OUTPUT.PUT_LINE('Area of Triangle: ' || triangle_area);
  DBMS_OUTPUT.PUT_LINE('Area of Square: ' || square_area);
END;
/
```

In the above PL/SQL block, we declare the necessary variables for the length, width, base, height, and side of the shapes. We then calculate the area for each shape using the corresponding formulas.

Inside the **BEGIN** and **END** block, we perform the calculations for the area of the rectangle, triangle, and square. We store the results in the variables **rectangle_area**, **triangle_area**, and **square_area**, respectively.

Finally, we use the **DBMS_OUTPUT.PUT_LINE** function to display the calculated areas in the console output.

To execute the PL/SQL block, you can run it in a PL/SQL editor or in an Oracle SQL Developer tool that supports PL/SQL execution.

3)

```
DECLARE
  num1 NUMBER := 10;
  num2 NUMBER := 20;
  num3 NUMBER := 15;
  max_num NUMBER;
BEGIN
  IF num1 >= num2 AND num1 >= num3 THEN
    max_num := num1;
  ELSIF num2 >= num1 AND num2 >= num3 THEN
    max_num := num2;
  ELSE
    max_num := num3;
  END IF;
```

```
  DBMS_OUTPUT.PUT_LINE('The maximum number among ' || num1 || ', ' || num2 || ', and ' ||
num3 || ' is ' || max_num);
END;
/
```

In the above PL/SQL block, we declare three variables `num1`, `num2`, and `num3` with the values 10, 20, and 15, respectively. We also declare a variable `max_num` to store the maximum number.

Inside the `BEGIN` and `END` block, we use an `IF-ELSE` statement to compare the values of the three numbers and determine the maximum among them. We assign the maximum value to the `max_num` variable accordingly.

Finally, we use the `DBMS_OUTPUT.PUT_LINE` function to display the maximum number in the console output.

To execute the PL/SQL block, you can run it in a PL/SQL editor or in an Oracle SQL Developer tool that supports PL/SQL execution.

4)

```
DECLARE
  n NUMBER := 5; -- Change the value of n to the desired number of elements
  sum NUMBER := 0;
BEGIN
  FOR i IN 1..n LOOP
    sum := sum + i;
  END LOOP;

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

In the above PL/SQL block, we declare the variable `n` to represent the number of elements for which we want to calculate the sum. You can modify the value of `n` to the desired number of elements.

Inside the `BEGIN` and `END` block, we use a FOR loop with the loop index variable `i` ranging from 1 to `n`. For each iteration of the loop, we add the value of `i` to the `sum` variable.

Finally, we use the `DBMS_OUTPUT.PUT_LINE` function to display the calculated sum in the console output.

To execute the PL/SQL block, you can run it in a PL/SQL editor or in an Oracle SQL Developer tool that supports PL/SQL execution.

5)

```
DECLARE
  n NUMBER := 10; -- Change the value of n to the desired number of Fibonacci numbers
  fib_number NUMBER;
  prev1 NUMBER := 0;
  prev2 NUMBER := 1;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
  DBMS_OUTPUT.PUT_LINE(prev1); -- Display the first Fibonacci number
  DBMS_OUTPUT.PUT_LINE(prev2); -- Display the second Fibonacci number

  FOR i IN 3..n LOOP
    fib_number := prev1 + prev2;
    DBMS_OUTPUT.PUT_LINE(fib_number); -- Display the current Fibonacci number

    prev1 := prev2;
    prev2 := fib_number;
  END LOOP;
END;
/
```

In the above PL/SQL block, we declare the variable `n` to represent the number of Fibonacci numbers we want to generate. You can modify the value of `n` to the desired number of Fibonacci numbers.

Inside the `BEGIN` and `END` block, we use a FOR loop with the loop index variable `i` ranging from 3 to `n`. We initialize `prev1` and `prev2` as the first and second Fibonacci numbers, respectively.

For each iteration of the loop, we calculate the current Fibonacci number by adding `prev1` and `prev2`. We then display the current Fibonacci number using the `DBMS_OUTPUT.PUT_LINE` function.

After calculating the current Fibonacci number, we update `prev1` and `prev2` to prepare for the next iteration of the loop.

To execute the PL/SQL block, you can run it in a PL/SQL editor or in an Oracle SQL Developer tool that supports PL/SQL execution.