Exercise 7

Simple on PL/SQL program to add two numbers

AIM:

To write a PL/SQL program that declares two numeric variables, adds them, and displays the result.

PROGRAM:

```
DECLARE
  a NUMBER := 10;
  b NUMBER := 20;
  sum NUMBER;
BEGIN
  sum := a + b;
  DBMS_OUTPUT.PUT_LINE('Sum = ' || sum);
END;
/
```

OUTPUT :

```
Sum = 30
```

RESULT:

The program was successfully executed.

Exercise 8

Simple on PL/SQL program to check even and odd number

AIM:

To write a PL/SQL program to check whether a given number is even or odd and display the result.

PROGRAM:

```
DECLARE
  num NUMBER := 15;
BEGIN
  IF num MOD 2 = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Even number');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Odd number');
  END IF;
END;
/
```

OUTPUT :

```
Odd number
```

RESULT:

The program was successfully executed.

Exercise 9

PL/SQL program to demonstrate LOOP

AIM:

To write a PL/SQL program that demonstrates the use of a LOOP statement.

PROGRAM:

```
DECLARE
  i NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('i = ' || i);
    i := i + 1;
    EXIT WHEN i > 5;
  END LOOP;
END;
/
```

OUTPUT :

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

RESULT:

The program was successfully executed..

Exercise 10

PL/SQL program to demonstrate WHILE LOOP

AIM:

To write a PL/SQL program that demonstrates the use of a WHILE loop.

PROGRAM:

```
DECLARE
  i NUMBER := 1;
BEGIN
  WHILE i <= 5 LOOP
    DBMS_OUTPUT.PUT_LINE('i = ' || i);
    i := i + 1;
  END LOOP;
END;
/
```

OUTPUT :

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

RESULT:

The program was successfully executed.

Exercise 11

PL/SQL program to demonstrate FOR LOOP

AIM:

To write a PL/SQL program that demonstrates the use of a FOR loop.

PROGRAM:

```
BEGIN
  FOR i IN 1..5 LOOP
    DBMS_OUTPUT.PUT_LINE('i = ' || i);
  END LOOP;
END;
/
```

OUTPUT :

```
i = 1
i = 2
i = 3
i = 4
i = 5
```

RESULT:

The program was successfully executed.

Exercise 12

PL/SQL program to demonstrate CASE

AIM:

To write a PL/SQL program that uses the CASE statement to display a message based on the value of a student's grade.

PROGRAM:

```
DECLARE
  grade CHAR := 'B';
BEGIN
  CASE grade
    WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
    WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    ELSE DBMS_OUTPUT.PUT_LINE('Needs Improvement');
  END CASE;
END;
/
```

OUTPUT :

```
Very Good
```

RESULT:

The program was successfully executed.

Exercise 13

PL/SQL program to demonstrate Searched CASE

AIM:

To write a PL/SQL program using a searched CASE statement to determine and display the grade based on the marks obtained.

PROGRAM:

```
DECLARE
  marks NUMBER := 78;
BEGIN
  CASE
    WHEN marks >= 90 THEN DBMS_OUTPUT.PUT_LINE('Grade: A');
    WHEN marks >= 80 THEN DBMS_OUTPUT.PUT_LINE('Grade: B');
    WHEN marks >= 70 THEN DBMS_OUTPUT.PUT_LINE('Grade: C');
    ELSE DBMS_OUTPUT.PUT_LINE('Grade: D');
  END CASE;
END;
/
```

OUTPUT :

```
Grade: C
```

RESULT:

The program was successfully executed.

Exercise 14

PL/SQL program to demonstrate Cursor

AIM:

To write a PL/SQL program that demonstrates the use of an explicit cursor to retrieve and display employee names and salaries from the employee's table.

PROGRAM:

Employees table creation :

```
CREATE TABLE employees (
  employee_id NUMBER PRIMARY KEY,
  first_name VARCHAR2(50),
  salary NUMBER
);
```

Inserting values to created table :

```
INSERT INTO employees (employee_id, first_name, salary) VALUES (101, 'John', 5000);
INSERT INTO employees (employee_id, first_name, salary) VALUES (102, 'Alice', 4500);
INSERT INTO employees (employee_id, first_name, salary) VALUES (103, 'Robert', 6000);
INSERT INTO employees (employee_id, first_name, salary) VALUES (104, 'Nina', 5200);
INSERT INTO employees (employee_id, first_name, salary) VALUES (105, 'David', 4800);
INSERT INTO employees (employee_id, first_name, salary) VALUES (106, 'Sara', 5500); --
extra row to test ROWNUM <= 5
COMMIT;
```

PL/SQL Cursor Program

```
DECLARE
  -- Define a cursor to fetch employee names
  CURSOR emp_cursor IS
    SELECT first_name, salary FROM employees WHERE ROWNUM <= 5;
```

```
  v_name employees.first_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('Name: ' || v_name || ', Salary: ' || v_salary);
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

OUTPUT :

```
Name: Alice, Salary: 4500
Name: Robert, Salary: 6000
Name: Nina, Salary: 5200
Name: David, Salary: 4800
```

RESULT:

The program was successfully executed.

Exercise 15

PL/SQL program to demonstrate Exception Handling

AIM:

To write a PL/SQL program that performs division of two numbers and handles the division by zero exception using exception handling blocks.

PROGRAM:

```
DECLARE
  v_num1 NUMBER := 10;
  v_num2 NUMBER := 0;  -- will cause division by zero error
  v_result NUMBER;
BEGIN
  v_result := v_num1 / v_num2;
  DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);

EXCEPTION
  WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred.');
END;
/
```

OUTPUT :

```
Error: Division by zero is not allowed.
```

RESULT:

The program was successfully executed.

Exercise 16

PL/SQL program to demonstrate User-defined Exception

AIM:

To write a PL/SQL program that raises and handles a user-defined exception when the salary is below a minimum acceptable level.

PROGRAM:

```
DECLARE
  v_salary NUMBER := 3000;
  e_low_salary EXCEPTION;  -- user-defined exception
BEGIN
  IF v_salary < 5000 THEN
    RAISE e_low_salary;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Salary is acceptable.');
EXCEPTION
  WHEN e_low_salary THEN
    DBMS_OUTPUT.PUT_LINE('Error: Salary is below minimum acceptable level.');
END;
/
```

OUTPUT :

```
Error: Salary is below minimum acceptable level.
```

RESULT:

The program was successfully executed.