**SET SERVEROUTPUT ON;**

**1 )PL/SQL Program To Add Two Numbers (All PL/SQL program: using function or procedure )**

1. *Procedure*

CREATE OR REPLACE PROCEDURE adder\_proc(n1 IN NUMBER, n2 IN NUMBER, n3 OUT NUMBER) AS

BEGIN

n3 := n1 + n2;

END;

/

DECLARE

n3 NUMBER;

BEGIN

adder\_proc(11, 22, n3);

dbms\_output.put\_line('Addition is: ' || n3);

END;

/

1. *Function*

DECLARE

n3 number(2);

FUNCTION adder(n1 in number, n2 in number)

RETURN number

is

n3 number(8);

BEGIN

n3 :=n1+n2;

RETURN n3;

END;

BEGIN

n3 := adder(11,22);

dbms\_output.put\_line('Addition is: ' || n3);

END;

**2) PL/SQL Program for Prime Number**

a)*Function*

CREATE OR REPLACE FUNCTION is\_prime(num NUMBER) RETURN BOOLEAN IS

BEGIN

IF num <= 1 THEN

RETURN FALSE;

ELSE

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

IF num MOD i = 0 THEN

RETURN FALSE;

END IF;

END LOOP;

END IF;

RETURN TRUE;

END;

/

DECLARE

num NUMBER := 17;

result BOOLEAN;

BEGIN

result := is\_prime(num);

IF result THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is a prime number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is not a prime number.');

END IF;

END;

/

B) *Procedure*

CREATE OR REPLACE PROCEDURE check\_prime(num NUMBER) AS

is\_prime BOOLEAN := TRUE;

BEGIN

IF num <= 1 THEN

is\_prime := FALSE;

ELSE

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

IF num MOD i = 0 THEN

is\_prime := FALSE;

EXIT;

END IF;

END LOOP;

END IF;

IF is\_prime THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is a prime number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is not a prime number.');

END IF;

END;

/

BEGIN

check\_prime(17);

END;

/

**3)PL/SQL Program to Find Factorial of a Number**

**a)** *Function*

CREATE OR REPLACE FUNCTION factorial(num NUMBER) RETURN NUMBER IS

result NUMBER := 1;

BEGIN

IF num<= 0 THEN

RETURN NULL; -- Factorial is not defined for negative numbers

ELSE

FOR i IN 1..num LOOP

result := result \* i;

END LOOP;

END IF;

RETURN result;

END;

/

DECLARE

numNUMBER := 5;

fact NUMBER;

BEGIN

fact := factorial(num);

DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || num || ' is: ' || fact);

END;

/

b) *Procedure*

CREATE OR REPLACE PROCEDURE calculate\_factorial(num IN NUMBER, fact OUT NUMBER) AS

BEGIN

fact := 1;

IF num<= 0 THEN

fact := NULL; -- Factorial is not defined for negative numbers

ELSE

FOR i IN 1..num LOOP

fact := fact \* i;

END LOOP;

END IF;

END;

/

DECLARE

numNUMBER := 5;

fact NUMBER;

BEGIN

calculate\_factorial(num, fact);

DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || num || ' is: ' || fact);

END;

/

**4) PL/SQL Program for Reverse of a Number**

a) *Function*

CREATE OR REPLACE FUNCTION reverse\_number(num NUMBER) RETURN NUMBER IS

reversed\_numNUMBER := 0;

temp\_numNUMBER := num; -- Assign the input parameter to a local variable

BEGIN

WHILE temp\_num> 0 LOOP

reversed\_num := reversed\_num \* 10 + temp\_num MOD 10;

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

RETURN reversed\_num;

END;

/

DECLARE

numNUMBER := 12345;

reversed\_num NUMBER;

BEGIN

reversed\_num := reverse\_number(num);

DBMS\_OUTPUT.PUT\_LINE('Reverse of ' || num || ' is: ' || reversed\_num);

END;

/

*b) Procedure*

CREATE OR REPLACE PROCEDURE reverse\_number\_proc(num IN NUMBER, reversed\_num OUT NUMBER) AS

temp\_numNUMBER := num;

BEGIN

reversed\_num := 0;

WHILE temp\_num> 0 LOOP

reversed\_num := reversed\_num \* 10 + temp\_num MOD 10;

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

END;

/

DECLARE

numNUMBER := 12345;

reversed\_num NUMBER;

BEGIN

reverse\_number\_proc(num, reversed\_num);

DBMS\_OUTPUT.PUT\_LINE('Reverse of ' || num || ' is: ' || reversed\_num);

END;

/

**5 )PL/SQL Program for Fibonacci Series**

1. *Function*

CREATE OR REPLACE FUNCTION fibonacci\_series(n IN NUMBER) RETURN VARCHAR2 IS

result VARCHAR2(4000) := '';

a NUMBER := 0;

b NUMBER := 1;

fib NUMBER;

BEGIN

IF n <= 0 THEN

RETURN 'Invalid input. Please enter a positive integer.';

ELSIF n = 1 THEN

RETURN '0';

ELSE

result := '0, 1';

FOR i IN 3..n LOOP

fib := a + b;

result := result || ', ' || fib;

a := b;

b := fib;

END LOOP;

END IF;

RETURN result;

END;

/

DECLARE

n NUMBER := 10; -- Specify the number of terms in the Fibonacci series

fib\_series VARCHAR2(4000);

BEGIN

fib\_series := fibonacci\_series(n);

DBMS\_OUTPUT.PUT\_LINE('Fibonacci Series: ' || fib\_series);

END;

/

*B ) Procedure*

CREATE OR REPLACE PROCEDURE fibonacci\_series\_proc(n IN NUMBER) AS

a NUMBER := 0;

b NUMBER := 1;

fib NUMBER;

BEGIN

IF n <= 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid input. Please enter a positive integer.');

ELSIF n = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('0');

ELSE

DBMS\_OUTPUT.PUT('0, 1');

FOR i IN 3..n LOOP

fib := a + b;

DBMS\_OUTPUT.PUT(', ' || fib);

a := b;

b := fib;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

END IF;

END;

/

DECLARE

n NUMBER := 10; -- Specify the number of terms in the Fibonacci series

BEGIN

fibonacci\_series\_proc(n);

END;

/

**6) PL/SQL Program to Check Number is Odd or Even**

*a) Function*

CREATE OR REPLACE FUNCTION check\_odd\_even\_func(num IN NUMBER) RETURN VARCHAR2 IS

result VARCHAR2(20);

BEGIN

IF MOD(num, 2) = 0 THEN

result := 'Even';

ELSE

result := 'Odd';

END IF;

RETURN result;

END;

/

DECLARE

numNUMBER := 8; -- Specify the number to be checked

odd\_even VARCHAR2(20);

BEGIN

odd\_even := check\_odd\_even\_func(num);

DBMS\_OUTPUT.PUT\_LINE('The number ' || num || ' is ' || odd\_even);

END;

/

1. *Procedure*

CREATE OR REPLACE PROCEDURE check\_odd\_even\_proc(num IN NUMBER) AS

BEGIN

IF MOD(num, 2) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('The number ' || num || ' is Even');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The number ' || num || ' is Odd');

END IF;

END;

/

DECLARE

numNUMBER := 7; -- Specify the number to be checked

BEGIN

check\_odd\_even\_proc(num);

END;

/

**7 )PL/SQL Program to Reverse a String**

*A ) Function*

CREATE OR REPLACE FUNCTION reverse\_string\_func(str IN VARCHAR2) RETURN VARCHAR2 IS

reversed\_str VARCHAR2(4000) := '';

BEGIN

IF str IS NULL THEN

RETURN NULL;

END IF;

FOR i IN REVERSE 1..LENGTH(str) LOOP

reversed\_str := reversed\_str || SUBSTR(str, i, 1);

END LOOP;

RETURN reversed\_str;

END;

/

DECLARE

input\_str VARCHAR2(100) := 'Hello World'; -- Specify the string to be reversed

reversed\_str VARCHAR2(4000);

BEGIN

reversed\_str := reverse\_string\_func(input\_str);

DBMS\_OUTPUT.PUT\_LINE('Original String: ' || input\_str);

DBMS\_OUTPUT.PUT\_LINE('Reversed String: ' || reversed\_str);

END;

/

*B) Procedure*

CREATE OR REPLACE PROCEDURE reverse\_string\_proc(str IN OUT VARCHAR2) AS

reversed\_str VARCHAR2(4000) := '';

BEGIN

IF str IS NULL THEN

RETURN;

END IF;

FOR i IN REVERSE 1..LENGTH(str) LOOP

reversed\_str := reversed\_str || SUBSTR(str, i, 1);

END LOOP;

str := reversed\_str;

END;

/

DECLARE

input\_str VARCHAR2(100) := 'Hello World'; -- Specify the string to be reversed

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Original String: ' || input\_str);

reverse\_string\_proc(input\_str);

DBMS\_OUTPUT.PUT\_LINE('Reversed String: ' || input\_str);

END;

/

**8). Pl/SQL Program for Palindrome Number**

A )*Function*

CREATE OR REPLACE FUNCTION check\_palindrome\_func(num IN NUMBER) RETURN BOOLEAN IS

reversed\_numNUMBER := 0;

original\_numNUMBER := num;

temp\_numNUMBER := num;

BEGIN

WHILE temp\_num> 0 LOOP

reversed\_num := reversed\_num \* 10 + MOD(temp\_num, 10);

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

IF original\_num = reversed\_num THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

END;

/

DECLARE

numNUMBER := 12321; -- Specify the number to be checked

is\_palindrome BOOLEAN;

BEGIN

is\_palindrome := check\_palindrome\_func(num);

IF is\_palindrome THEN

DBMS\_OUTPUT.PUT\_LINE('The number ' || num || ' is a palindrome.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The number ' || num || ' is not a palindrome.');

END IF;

END;

/

*B ) procedure*

CREATE OR REPLACE PROCEDURE check\_palindrome\_proc(num IN NUMBER) AS

reversed\_numNUMBER := 0;

original\_numNUMBER := num;

temp\_numNUMBER := num;

BEGIN

WHILE temp\_num> 0 LOOP

reversed\_num := reversed\_num \* 10 + MOD(temp\_num, 10);

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

IF original\_num = reversed\_num THEN

DBMS\_OUTPUT.PUT\_LINE('The number ' || original\_num || ' is a palindrome.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The number ' || original\_num || ' is not a palindrome.');

END IF;

END;

/

DECLARE

numNUMBER := 12321; -- Specify the number to be checked

BEGIN

check\_palindrome\_proc(num);

END;

/

**9 )PL/SQL Program to Swap two Numbers**

1. Function

CREATE OR REPLACE FUNCTION swap\_numbers\_func(num1 IN NUMBER, num2 IN NUMBER) RETURN VARCHAR2 AS

temp NUMBER;

swapped\_num1 NUMBER := num2;

swapped\_num2 NUMBER := num1;

BEGIN

RETURN 'Numbers swapped successfully: num1 = ' || swapped\_num1 || ', num2 = ' || swapped\_num2;

END;

/

DECLARE

a NUMBER := 10;

b NUMBER := 20;

result VARCHAR2(100);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before swapping: a = ' || a || ', b = ' || b);

result := swap\_numbers\_func(a, b);

DBMS\_OUTPUT.PUT\_LINE(result);

END;

/

B ) Procedure

CREATE OR REPLACE PROCEDURE swap\_numbers\_proc(num1 IN OUT NUMBER, num2 IN OUT NUMBER) AS

temp NUMBER;

BEGIN

temp := num1;

num1 := num2;

num2 := temp;

END;

/

DECLARE

a NUMBER := 10;

b NUMBER := 20;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Before swapping: a = ' || a || ', b = ' || b);

swap\_numbers\_proc(a, b);

DBMS\_OUTPUT.PUT\_LINE('After swapping: a = ' || a || ', b = ' || b);

END;

/

**10 )PL/SQL Program for Armstrong Number**

*A ) Function*

DECLARE

numNUMBER := 153;

result BOOLEAN;

FUNCTION is\_armstrong\_number(num NUMBER) RETURN BOOLEAN IS

total\_digits NUMBER;

temp\_num NUMBER;

digit NUMBER;

sum\_of\_powersNUMBER := 0;

BEGIN

total\_digits := LENGTH(num);

temp\_num := num;

WHILE temp\_num> 0 LOOP

digit := MOD(temp\_num, 10);

sum\_of\_powers := sum\_of\_powers + POWER(digit, total\_digits);

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

RETURN (sum\_of\_powers = num);

END;

BEGIN

result := is\_armstrong\_number(num);

IF result THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is an Armstrong number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is not an Armstrong number.');

END IF;

END;

*B ) procedure*

DECLARE

numNUMBER := 153;

result BOOLEAN;

PROCEDURE check\_armstrong\_number\_proc(num NUMBER) AS

total\_digits NUMBER;

temp\_num NUMBER;

digit NUMBER;

sum\_of\_powersNUMBER := 0;

BEGIN

total\_digits := LENGTH(num);

temp\_num := num;

WHILE temp\_num> 0 LOOP

digit := MOD(temp\_num, 10);

sum\_of\_powers := sum\_of\_powers + POWER(digit, total\_digits);

temp\_num := FLOOR(temp\_num / 10);

END LOOP;

IF sum\_of\_powers = num THEN

DBMS\_OUTPUT.PUT\_LINE(num || ' is an Armstrong number.');

ELSE

DBMS\_OUTPUT.PUT\_LINE(num || ' is not an Armstrong number.');

END IF;

END;

BEGIN

check\_armstrong\_number\_proc(num);

END;

/

**11) PL/SQL Program to Find Greatest of Three Numbers**

**A )***Function*

CREATE OR REPLACE FUNCTION find\_greatest(num1 NUMBER, num2 NUMBER, num3 NUMBER) RETURN NUMBER AS

greatest NUMBER;

BEGIN

IF num1 >= num2 AND num1 >= num3 THEN

greatest := num1;

ELSIF num2 >= num1 AND num2 >= num3 THEN

greatest := num2;

ELSE

greatest := num3;

END IF;

RETURN greatest;

END;

/

DECLARE

num1 NUMBER := 10;

num2 NUMBER := 20;

num3 NUMBER := 15;

greatest\_num NUMBER;

BEGIN

greatest\_num := find\_greatest(num1, num2, num3);

DBMS\_OUTPUT.PUT\_LINE('The greatest number is: ' || greatest\_num);

END;

/

B )*Procedure*

CREATE OR REPLACE PROCEDURE find\_greatest\_number(

num1 IN NUMBER,

num2 IN NUMBER,

num3 IN NUMBER,

greatest\_num OUT NUMBER

)

IS

BEGIN

IF num1 >= num2 AND num1 >= num3 THEN

greatest\_num := num1;

ELSIF num2 >= num1 AND num2 >= num3 THEN

greatest\_num := num2;

ELSE

greatest\_num := num3;

END IF;

END;

/

DECLARE

num1 NUMBER := 10;

num2 NUMBER := 20;

num3 NUMBER := 15;

greatest NUMBER;

BEGIN

find\_greatest\_number(num1, num2, num3, greatest);

DBMS\_OUTPUT.PUT\_LINE('The greatest number is: ' || greatest);

END;

/

**12 )PL/SQL procedure to check whether the current day is a weekend or weekday**

*A ) Function*

CREATE OR REPLACE FUNCTION check\_weekend

RETURN VARCHAR2

IS

current\_day VARCHAR2(20);

is\_weekend BOOLEAN;

BEGIN

-- Get the current day

current\_day := TO\_CHAR(SYSDATE, 'Day');

-- Check if the current day is a weekend

IF current\_day = 'Saturday' OR current\_day = 'Sunday' THEN

is\_weekend := TRUE;

ELSE

is\_weekend := FALSE;

END IF;

-- Return the result

IF is\_weekend THEN

RETURN 'Weekend';

ELSE

RETURN 'Weekday';

END IF;

END;

/

-- Example usage

DECLARE

day\_type VARCHAR2(20);

BEGIN

day\_type := check\_weekend();

DBMS\_OUTPUT.PUT\_LINE('Today is ' || day\_type);

END;

/

*B ) Procedure*

CREATE OR REPLACE PROCEDURE check\_weekend\_weekday AS

current\_day VARCHAR2(20);

is\_weekend BOOLEAN;

BEGIN

-- Get the current day

current\_day := TO\_CHAR(SYSDATE, 'Day');

-- Check if the current day is a weekend

IF current\_day = 'Saturday' OR current\_day = 'Sunday' THEN

is\_weekend := TRUE;

ELSE

is\_weekend := FALSE;

END IF;

-- Display the result

IF is\_weekend THEN

DBMS\_OUTPUT.PUT\_LINE('Today is a Weekend');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Today is a Weekday');

END IF;

END;

/

-- Example usage

BEGIN

check\_weekend\_weekday();

END;

/