

Database Management System

PI SQL SOLUTIONS

NAME : PRATHESHA J

ROLL NO : 241001172

1. Display Hello Message

```
SET SERVEROUTPUT ON; DECLARE
    a VARCHAR2(20);
BEGIN
    a := 'Hello';
    DBMS_OUTPUT.PUT_LINE(a);
END;
/
```

Expected Output:

Hello

PL/SQL procedure successfully completed.

2. Input Value from User and Display It

```
SET SERVEROUTPUT ON; DECLARE
    a VARCHAR2(20);
BEGIN
    a := '&a';
```

```
    DBMS_OUTPUT.PUT_LINE(a);
END;
/
```

When prompted, enter: 5 Expected Output:

```
Enter value for a: 5 old
4:      a := '&a';
new   4:      a := '5';
5
```

PL/SQL procedure successfully completed.

3. Greatest of Two Numbers

```
SET SERVEROUTPUT ON;
DECLARE      a
NUMBER(7);      b
NUMBER(7); BEGIN
a := &a;      b := &b;
IF (a > b) THEN
    DBMS_OUTPUT.PUT_LINE('The greater of the two is: '
|| a);
ELSE
    DBMS_OUTPUT.PUT_LINE('The greater of the two is: '
|| b);
END IF;
END;
/
```

When prompted, enter: a=5, b=9 Expected Output:

```
Enter value for a: 5 Enter
value for b: 9 old      5:      a
:= &a; new      5:      a := 5;
old      6:      b := &b; new
6:      b := 9; The greater of
the two is: 9
```

PL/SQL procedure successfully completed.

4. Greatest of Three Numbers

```
SET SERVEROUTPUT ON;
DECLARE      a
NUMBER(7);      b
NUMBER(7);      c
NUMBER(7); BEGIN
a := &a;      b := &b;
c := &c;
      IF (a > b AND a > c) THEN
          DBMS_OUTPUT.PUT_LINE('The greatest of the three is:
' || a);
      ELSIF (b > c) THEN
          DBMS_OUTPUT.PUT_LINE('The greatest of the three is:
' || b);
      ELSE
          DBMS_OUTPUT.PUT_LINE('The greatest of the three is:
' || c);
      END IF;
END;
/
```

When prompted, enter: a=5, b=7, c=1 Expected Output:

```
Enter value for a: 5
```

```
Enter value for b: 7 Enter value
for c: 1 old    6:      a := &a;
new    6:      a := 5; old    7:
b := &b; new    7:      b := 7;
old    8:      c := &c; new    8:
c := 1; The greatest of the
three is: 7
```

PL/SQL procedure successfully completed.

5. Print Numbers 1 to 5 Using Simple Loop

```
SET SERVEROUTPUT ON;
DECLARE      a NUMBER
:= 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE(a);
    a := a + 1;
    EXIT WHEN a > 5;
  END LOOP;
END;
/
```

Expected Output:

```
1
2
3
4
5
```

PL/SQL procedure successfully completed.

6. Print Numbers 1 to 4 Using WHILE Loop

```
SET SERVEROUTPUT ON;
DECLARE      a NUMBER
:= 1;
BEGIN
    WHILE (a < 5) LOOP
DBMS_OUTPUT.PUT_LINE(a);          a
:= a + 1;
    END LOOP;
END;
/
```

Expected Output:

```
1
2
3
4
```

PL/SQL procedure successfully completed.

7. Print Numbers 1 to 5 Using FOR Loop

```
SET SERVEROUTPUT ON;
DECLARE      a NUMBER
:= 1;
BEGIN
    FOR a IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE(a);
    END LOOP;
END;
/
```

Expected Output:

```
1  
2  
3  
4  
5
```

PL/SQL procedure successfully completed.

8. Print Numbers 1 to 5 in Reverse Using FOR Loop

```
SET SERVEROUTPUT ON;  
DECLARE      a NUMBER  
:= 1;  
BEGIN  
    FOR a IN REVERSE 1..5 LOOP  
        DBMS_OUTPUT.PUT_LINE(a);  
    END LOOP;  
END;  
/
```

Expected Output:

```
5  
4  
3  
2  
1
```

PL/SQL procedure successfully completed.

9. Calculate Area of Circle

```
SET SERVEROUTPUT ON; DECLARE
    pi CONSTANT NUMBER(4,2) := 3.14;
    area NUMBER(20);      radius
    NUMBER(20); BEGIN
        radius := &radius;
        area := pi * POWER(radius, 2);
        DBMS_OUTPUT.PUT_LINE('The area of circle is: ' || area);
    END;
/
```

When prompted, enter: radius=2 Expected Output:

```
Enter value for radius: 2 old
6:      radius := &radius; new
6:      radius := 2;
```

```
The area of circle is: 12.56
```

PL/SQL procedure successfully completed.

10. Bank Account Operations

Create table and insert data:

```
CREATE TABLE saccount (accno NUMBER(5), name VARCHAR2(20),
bal NUMBER(10));
INSERT INTO saccount VALUES (1, 'mala', 20000);
INSERT INTO saccount VALUES (2, 'kala', 30000);
COMMIT;
```

PL/SQL block for debit operation:

```
SET SERVEROUTPUT ON;
DECLARE
  a_bal NUMBER(7);      a_no
  VARCHAR2(20);        debit
  NUMBER(7) := 2000;    min_amt
  NUMBER(7) := 500; BEGIN
  a_no := '&a_no';

  SELECT bal INTO a_bal FROM saccount WHERE accno = a_no;
  a_bal := a_bal - debit;

  IF (a_bal > min_amt) THEN
    UPDATE saccount SET bal = bal - debit WHERE accno =
  a_no;
    DBMS_OUTPUT.PUT_LINE('Transaction successful');

  ELSE
    DBMS_OUTPUT.PUT_LINE('Insufficient balance after
transaction');

  END IF;

END;
/
```

When prompted, enter: a_no=1 Expected Output:

```
Enter value for a_no: 1 old
8:      a_no := '&a_no'; new
8:      a_no := '1';
Transaction successful
```

PL/SQL procedure successfully completed.

Verify the update:

```
SELECT * FROM saccount;
```

Expected Output:

ACCNO	NAME	BAL
1	mala	18000
2	kala	30000

11. Route Fare Calculation

Create table and insert data:

```
CREATE TABLE sroutes (rno NUMBER(5), origin VARCHAR2(20),
destination VARCHAR2(20), fare
NUMBER(10), distance NUMBER(10));
INSERT INTO sroutes VALUES (2, 'chennai', 'dindugal', 400,
230);
INSERT INTO sroutes VALUES (3, 'chennai', 'madurai', 250,
300);
INSERT INTO sroutes VALUES (6, 'thanjavur', 'palani', 350,
370);
COMMIT;
```

PL/SQL block for fare calculation:

```
SET SERVEROUTPUT ON; DECLARE
    route sroutes.rno%TYPE;
    fares sroutes.fare%TYPE;
    dist sroutes.distance%TYPE;
BEGIN
    route := &route;
    SELECT fare, distance INTO fares, dist FROM sroutes
```

```

WHERE rno = route;

IF (dist < 250) THEN
    UPDATE sroutes SET fare = 300 WHERE rno = route;
    DBMS_OUTPUT.PUT_LINE('Fare updated to 300');
ELSIF dist BETWEEN 250 AND 370 THEN
    UPDATE sroutes SET fare = 400 WHERE rno = route;
    DBMS_OUTPUT.PUT_LINE('Fare updated to 400');
ELSIF (dist > 400) THEN
    DBMS_OUTPUT.PUT_LINE('Sorry, distance too long');
END IF;
COMMIT;

END;
/

```

When prompted, enter: route=3 Expected Output:

```

Enter value for route: 3 old
6:      route := &route; new
6:      route := 3; Fare
updated to 400

```

PL/SQL procedure successfully completed.

Verify the update:

```
SELECT * FROM sroutes;
```

Expected Output:

RNO	ORIGIN	DESTINATION
FARE	DISTANCE	

```

-----
-- 
2    chennai           dindugal      400
230
3    chennai           madurai      400
300
6    thanjavur        palani
350      370

```

12. Calculate and Store Circle Areas

Create table:

```
CREATE TABLE scalculate (radius NUMBER(3), area
NUMBER(5,2));
```

PL/SQL block to calculate areas:

```

SET SERVEROUTPUT ON;
DECLARE
    pi CONSTANT NUMBER(4,2) := 3.14;
    area NUMBER(5,2);      radius
    NUMBER(3); BEGIN      radius := 3;
        WHILE (radius <= 7) LOOP
            area := pi * POWER(radius, 2);
            INSERT INTO scalculate VALUES (radius, area);
            radius := radius + 1;
        END LOOP;
        COMMIT;
    END;
/

```

Expected Output: PL/SQL procedure successfully completed.

Verify the data:

```
SELECT * FROM scalculate;
```

Expected Output:

RADIUS	AREA
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86

13. Calculate Factorial

```
SET SERVEROUTPUT ON;
DECLARE      f
NUMBER(4) := 1;      i
NUMBER(4); BEGIN      i
:= &i;
      WHILE (i >= 1) LOOP
f := f * i;          i :=
i - 1;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('The factorial is: ' || f);
END;
/
```

When prompted, enter: i=5 Expected Output:

```
Enter value for i: 5 old
5:      i := &i; new   5:
i := 5;
The factorial is: 120

PL/SQL procedure successfully completed.
```

Additional PL/SQL Control Structure Examples:

CASE Statement Example:

```
SET SERVEROUTPUT ON;
DECLARE      grade
CHAR(1);      result
VARCHAR2(20);
BEGIN
grade := '&grade';

CASE grade
    WHEN 'A' THEN result := 'Excellent';
    WHEN 'B' THEN result := 'Very Good';
    WHEN 'C' THEN result := 'Good';
    WHEN 'D' THEN result := 'Fair';
    WHEN 'F' THEN result := 'Poor';
    ELSE result := 'No such grade';
END CASE;

DBMS_OUTPUT.PUT_LINE('Result: ' || result);
END;
/
```

Nested IF Example:

```
SET SERVEROUTPUT ON;
DECLARE      marks
NUMBER; BEGIN
    marks := &marks;

    IF marks >= 90 THEN
        DBMS_OUTPUT.PUT_LINE('Grade: A');
    ELSE
        IF marks >= 75 THEN
            DBMS_OUTPUT.PUT_LINE('Grade: B');
        ELSE
            IF marks >= 50 THEN
                DBMS_OUTPUT.PUT_LINE('Grade: C');
            ELSE
                DBMS_OUTPUT.PUT_LINE('Grade: F');
            END IF;
        END IF;
    END IF;
END;
/
```

Key PL/SQL Concepts Demonstrated:

1. **Variable Declaration** - Using DECLARE section
2. **Constants** - Using CONSTANT keyword
3. **User Input** - Using & substitution variables
4. **Conditional Statements** - IF-THEN-ELSE, ELSIF
5. **Loop Structures** - Simple LOOP, WHILE LOOP, FOR LOOP
6. **DML Operations** - INSERT, UPDATE, DELETE within PL/SQL
7. **Exception Handling** - Implicit handling in some examples

8. Database Interaction - SELECT INTO, DML with COMMIT