**DATABASE LEVEL-3**

**1.** **Write a PL/SQL block to create a package that contains a function and a procedure. The function should take in an array of integers and return the sum of the array. The procedure should take in an array of integers and sort the array in descending order.**

CREATE OR REPLACE PACKAGE array\_operations\_pkg AS

-- Function to calculate the sum of an array of integers

FUNCTION calculate\_sum(arr IN sys.odcinumberlist) RETURN NUMBER;

-- Procedure to sort an array of integers in descending order

PROCEDURE sort\_array\_desc(arr IN OUT sys.odcinumberlist);

END array\_operations\_pkg;

CREATE OR REPLACE PACKAGE BODY array\_operations\_pkg AS

-- Function to calculate the sum of an array of integers

FUNCTION calculate\_sum(arr IN sys.odcinumberlist) RETURN NUMBER IS

total\_sum NUMBER := 0;

BEGIN

FOR i IN 1..arr.COUNT LOOP

total\_sum := total\_sum + arr(i);

END LOOP;

RETURN total\_sum;

END calculate\_sum;

-- Procedure to sort an array of integers in descending order

PROCEDURE sort\_array\_desc(arr IN OUT sys.odcinumberlist) IS

temp NUMBER;

BEGIN

FOR i IN 1..arr.COUNT-1 LOOP

FOR j IN i+1..arr.COUNT LOOP

IF arr(j) > arr(i) THEN

temp := arr(i);

arr(i) := arr(j);

arr(j) := temp;

END IF;

END LOOP;

END LOOP;

END sort\_array\_desc;

END array\_operations\_pkg;

**2. Write a PL/SQL block to create a nested table type, insert data into the nested table, and then query the nested table to retrieve specific data.**

DECLARE

-- Define the nested table type

TYPE employee\_nt IS TABLE OF VARCHAR2(100);

-- Declare a variable of the nested table type

employees employee\_nt := employee\_nt();

BEGIN

-- Insert data into the nested table

employees.EXTEND(3);

employees(1) := 'John Smith';

employees(2) := 'Jane Doe';

employees(3) := 'Michael Johnson';

-- Query the nested table to retrieve specific data

FOR i IN 1..employees.COUNT LOOP

IF employees(i) LIKE '%Smith' THEN

DBMS\_OUTPUT.PUT\_LINE('Employee with "Smith" in the name: ' || employees(i));

END IF;

END LOOP;

END;

**3. Write a PL/SQL block to create a stored procedure that takes in two dates and returns a list of employees who were hired between those dates, along with their hire dates and job title.**

CREATE OR REPLACE PROCEDURE GetEmployeesByHireDate(

p\_start\_date IN DATE,

p\_end\_date IN DATE

) AS

BEGIN

FOR emp IN (

SELECT e.EmpID, e.DateofJoining, e.EmpPosition

FROM EmployeePosition e

JOIN EmployeeInfo j ON e.EmpID = j.EmpID

WHERE e.DateofJoining BETWEEN p\_start\_date AND p\_end\_date

)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp.EmpID);

DBMS\_OUTPUT.PUT\_LINE('Hire Date: ' || TO\_CHAR(emp.DateofJoining, 'DD-MON-YYYY'));

DBMS\_OUTPUT.PUT\_LINE('Job Title: ' || emp.EmpPosition);

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

END LOOP;

END;

/

--to view the output

SET SERVEROUTPUT ON;

BEGIN

GetEmployeesByHireDate(TO\_DATE('01-JAN-2023', 'DD-MON-YYYY'), TO\_DATE('31-DEC-2023', 'DD-MON-YYYY'));

END;

/

**4. Write a PL/SQL block to create a function that takes in a string as input and returns the string in reverse order.**

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

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

BEGIN

IF input\_string IS NOT NULL THEN

FOR i IN REVERSE 1 .. LENGTH(input\_string) LOOP

reversed\_string := reversed\_string || SUBSTR(input\_string, i, 1);

END LOOP;

END IF;

RETURN reversed\_string;

END;

/

DECLARE

input\_string VARCHAR2(100) := 'Hello, World!';

reversed\_string VARCHAR2(100);

BEGIN

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

DBMS\_OUTPUT.PUT\_LINE('Original string: ' || input\_string);

DBMS\_OUTPUT.PUT\_LINE('Reversed string: ' || reversed\_string);

END;

/

**5. Write a PL/SQL block to create a trigger that automatically updates a table with the current date and time whenever a row is inserted.**

CREATE OR REPLACE TRIGGER update\_date\_trigger

BEFORE INSERT ON EmployeePosition

FOR EACH ROW

BEGIN

:NEW.DateOfjoining := SYSDATE;

END;