PL/SQL : Introduction and Control Structures

Introduction to PL/SQL

•**PL/SQL (Procedural Language/SQL)** is Oracle's procedural extension of SQL. (You specify not only what to do like SQL but also How to do)

•It allows you to write full programs to control logic, process data, and handle transactions within Oracle databases.

•Combines SQL for data manipulation and a procedural language for flow control.

Efficient Data Processing: PL/SQL can send entire blocks of SQL statements to the database at once.

Error Handling: PL/SQL provides robust error handling using EXCEPTION blocks.

Security: PL/SQL code can be stored in the database, and access can be restricted.

Modularity: You can write procedures, functions, and packages to promote reusable code.

Basic Structure of a PL/SQL Block

A PL/SQL block has three sections:

1.Declarative Section: Where variables, constants etc are declared.

2.Executable Section: Where the actual logic and SQL queries are written.

3.Exception Handling Section: Where errors or exceptions are managed

```
DECLARE
   -- Declaration of variables and constants
BEGIN
   -- Main logic and SQL queries
EXCEPTION
   -- Exception handling
END;
/  -- used for running the query
```

Variables and Data Types

Common datatypes:

Number: For Numeric Values.
Varchar2: For Character Strings.
Date: For Date And Time Values.
Boolean: For True/False Values.

```
DECLARE
   v_salary NUMBER(8,2);    -- Number variable for salary
   v_name VARCHAR2(50);     -- String variable for employee name
BEGIN
   -- Code goes here
END;
```

Control Structures in PL/SQL

Control structures in PL/SQL provide ways to control the flow of execution based on conditions or loops.

Conditional Control: IF Statements

IF-THEN: Executes statements if a condition is true.

```
IF condition THEN
   -- Statements
END IF;
```

IF-THEN-ELSE: Adds an alternate path if the condition is false.

```
IF condition THEN
   -- Statements if true
ELSE
   -- Statements if false
END IF;
```

IF-THEN-ELSIF-ELSE: Multiple conditions.

```
IF condition1 THEN
   -- Statements if condition1 is true
ELSIF condition2 THEN
   -- Statements if condition2 is true
ELSE
   -- Statements if all conditions are false
END IF;
```

Sequential Control: GOTO Statement

Transfers control unconditionally to another part of the program.

```
<<label_name>>
BEGIN
   -- Statements
   GOTO label_name;
END;
```

Iterative Control: Loops

FOR Loop: Used to iterate over a fixed number of times

```
FOR i IN 1..10 LOOP
   DBMS_OUTPUT.PUT_LINE('Value: ' || i);
END LOOP;
```

WHILE Loop: Loops while a condition is true.

```
WHILE condition LOOP
   -- Statements
END LOOP;
```

Simple Loop: Loops indefinitely until EXIT is called.

```
LOOP
   -- Statements
   EXIT WHEN condition;
END LOOP;
```

1. Add Two Numbers: Write a PL/SQL block to add two numbers and display the result.

2. Find the Square of a Number: Write a PL/SQL block to find the square of a number and display it.

3. Calculate the Factorial of a Number: Write a PL/SQL block to calculate the factorial of a number.

4. Check if a Number is Even or Odd: Write a PL/SQL block to check if a number is even or odd.

5. Concatenate Two Strings: Write a PL/SQL block to concatenate two strings and display the result.

6. Calculate the Area of a Rectangle: Write a PL/SQL block to calculate the area of a rectangle.

7. Check if a Number is Positive, Negative, or Zero: Write a PL/SQL block to check if a number is positive, negative, or zero.

```
 1  declare
  2  dd number:=2;
  3  ee number:=4;
  4  cc number;
  5  begin
  6  cc:=dd+ee;
  7  dbms_output.put_line(cc);
  8* end;
```

```
ACCEPT a PROMPT 'Enter value for a: '
ACCEPT b PROMPT 'Enter value for b: '
SET SERVEROUTPUT ON;

DECLARE
   v_a NUMBER := &a;  -- Substitution variable used here
   v_b NUMBER := &b;  -- Substitution variable used here
   v_c NUMBER;
BEGIN
   v_c := v_a * v_b;
   DBMS_OUTPUT.PUT_LINE('The result is ');
END;
/
```

1. Display the First 10 Natural Numbers: Write a PL/SQL block to display the first 10 natural numbers.

2. Calculate the Sum of Numbers in a Range: Write a PL/SQL block to calculate the sum of numbers from 1 to 100.

3. Find the Largest of Three Numbers: Write a PL/SQL block to find the largest of three numbers.

4. Generate a Multiplication Table for a Number: Write a PL/SQL block to generate a multiplication table for the number 7.

5. Find the Sum of Even Numbers in a Range: Write a PL/SQL block to find the sum of even numbers from 1 to 20.

6. Display Fibonacci Series up to a Certain Number: Write a PL/SQL block to display the Fibonacci series up to 100.

SQL Statements in PL/SQL

You can perform all SQL operations inside a PL/SQL block, including INSERT, UPDATE, DELETE, and SELECT

```
BEGIN
   INSERT INTO employees (employee_id, first_name,
last_name, salary)
   VALUES (1, 'John', 'Doe', 5000);
END;
/
```

Selecting data into a variable

```
DECLARE
   v_salary NUMBER;
BEGIN
   SELECT salary INTO v_salary FROM employees WHERE
employee_id = 101;
   DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
END;
/
```

Display All Columns of All Employees

```
BEGIN
    FOR emp_rec IN (SELECT * FROM employees) LOOP
        DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.employee_id ||
                             ', Name: ' || emp_rec.first_name || ' `
                             || emp_rec.last_name ||
                             ', Salary: ' || emp_rec.salary ||
                             ', Hire Date: ' || emp_rec.hire_date);
    END LOOP;
END;
/
```

Display Employees with a Specific Salary Range

```
BEGIN
    FOR emp_rec IN (SELECT first_name, last_name, salary
FROM employees WHERE salary BETWEEN 4000 AND 6000) LOOP

    DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.first_name ||
    ' ' || emp_rec.last_name || ', Salary: ' ||
    emp_rec.salary);

    END LOOP;
END;
/
```

Display Employees Ordered by Salary

```
BEGIN
    FOR emp_rec IN (SELECT first_name, last_name, salary
FROM employees ORDER BY salary DESC) LOOP
        DBMS_OUTPUT.PUT_LINE('Name: ' ||
emp_rec.first_name || ' ' || emp_rec.last_name ||
                              ', Salary: ' ||
emp_rec.salary);
    END LOOP;
END;
/
```

Display Employee with Highest Salary

```
DECLARE
    v_max_salary NUMBER;
BEGIN
    SELECT MAX(salary) INTO v_max_salary FROM employees;

    FOR emp_rec IN (SELECT first_name, last_name FROM
    employees WHERE salary = v_max_salary) LOOP

    DBMS_OUTPUT.PUT_LINE('Highest Salary: ' || v_max_salary
    || ', Employee: ' || emp_rec.first_name || ' ' ||
    emp_rec.last_name);
    END LOOP;
END;
/
```

Display Employees with Names Starting with 'J'

```
BEGIN
    FOR emp_rec IN (SELECT first_name, last_name FROM
    employees WHERE first_name LIKE 'J%') LOOP

    DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.first_name
    || ' ' || emp_rec.last_name);
    END LOOP;
END;
/
```

Display Total Number of Employees

```
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM employees;
    DBMS_OUTPUT.PUT_LINE('Total number of employees: ' 
    || v_count);
END;
/
```