**PL/SQL Control Structures, Exception Handling, and Cursors**

**1. Control Structures in PL/SQL**

Control structures determine the flow of execution based on conditions or iterations. PL/SQL supports:

* Conditional statements (IF-ELSE, CASE)
* Loops (FOR, WHILE, LOOP)

**1.1 IF-ELSE**

Used to execute a block of code conditionally.

**Syntax:**

IF condition THEN

-- statements

ELSIF condition THEN

-- statements

ELSE

-- statements

END IF;

**Example:**

DECLARE

v\_salary NUMBER := 30000;

v\_bonus NUMBER;

BEGIN

IF v\_salary > 50000 THEN

v\_bonus := 10000;

ELSIF v\_salary > 30000 THEN

v\_bonus := 5000;

ELSE

v\_bonus := 2000;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Bonus: ' || v\_bonus);

END;

**1.2 CASE Statement**

Simplifies multiple IF conditions.

**Example:**

DECLARE

v\_grade CHAR := 'B';

v\_remark VARCHAR2(30);

BEGIN

CASE v\_grade

WHEN 'A' THEN v\_remark := 'Excellent';

WHEN 'B' THEN v\_remark := 'Good';

WHEN 'C' THEN v\_remark := 'Average';

ELSE v\_remark := 'Fail';

END CASE;

DBMS\_OUTPUT.PUT\_LINE('Remark: ' || v\_remark);

END;

**1.3 Loops**

**a. FOR Loop**

Executes a block of code a fixed number of times.

FOR i IN 1..5 LOOP

DBMS\_OUTPUT.PUT\_LINE('i = ' || i);

END LOOP;

**b. WHILE Loop**

Executes as long as condition is true.

DECLARE

i NUMBER := 1;

BEGIN

WHILE i <= 5 LOOP

DBMS\_OUTPUT.PUT\_LINE('i = ' || i);

i := i + 1;

END LOOP;

END;

**c. Basic LOOP**

Repeats until explicitly exited.

DECLARE

i NUMBER := 1;

BEGIN

LOOP

EXIT WHEN i > 5;

DBMS\_OUTPUT.PUT\_LINE('i = ' || i);

i := i + 1;

END LOOP;

END;

**2. Exception Handling**

PL/SQL handles errors gracefully using exceptions.

**2.1 Predefined Exceptions**

Examples: NO\_DATA\_FOUND, ZERO\_DIVIDE

**Example:**

BEGIN

SELECT salary INTO v\_salary FROM employees WHERE emp\_id = 9999;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee not found!');

END;

**2.2 User-defined Exceptions**

You can define and raise custom exceptions.

**Example:**

DECLARE

e\_low\_salary EXCEPTION;

v\_salary NUMBER := 2000;

BEGIN

IF v\_salary < 3000 THEN

RAISE e\_low\_salary;

END IF;

EXCEPTION

WHEN e\_low\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('Salary below minimum limit.');

END;

**3. Cursors in PL/SQL**

Cursors allow row-by-row processing of SQL queries.

**3.1 Implicit Cursors**

Automatically created for DML/SELECT INTO.

**Example:**

DECLARE

v\_name employees.emp\_name%TYPE;

BEGIN

SELECT emp\_name INTO v\_name FROM employees WHERE emp\_id = 1;

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || v\_name);

END;

**3.2 Explicit Cursor**

Manually defined for queries returning multiple rows.

**Example:**

DECLARE

CURSOR emp\_cur IS SELECT emp\_name, salary FROM employees;

v\_name employees.emp\_name%TYPE;

v\_salary employees.salary%TYPE;

BEGIN

OPEN emp\_cur;

LOOP

FETCH emp\_cur INTO v\_name, v\_salary;

EXIT WHEN emp\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_name || ' - ' || v\_salary);

END LOOP;

CLOSE emp\_cur;

END;

**3.3 Cursor FOR Loop**

Simplifies looping through cursor records.

BEGIN

FOR emp\_rec IN (SELECT emp\_name, salary FROM employees) LOOP

DBMS\_OUTPUT.PUT\_LINE(emp\_rec.emp\_name || ': ' || emp\_rec.salary);

END LOOP;

END;

**3.4 Parameterized Cursor**

Pass values to filter cursor data.

DECLARE

CURSOR dept\_emp\_cur(p\_dept\_id NUMBER) IS

SELECT emp\_name FROM employees WHERE dept\_id = p\_dept\_id;

BEGIN

FOR emp IN dept\_emp\_cur(10) LOOP

DBMS\_OUTPUT.PUT\_LINE('Emp: ' || emp.emp\_name);

END LOOP;

END;

# 4. Case Study Scenarios

## Case Study 1: Grade Assignment (Control Structures)

Create a block that assigns a grade based on marks.

DECLARE

v\_student\_name VARCHAR2(50) := 'Riya';

v\_marks NUMBER := 87;

v\_grade CHAR(1);

BEGIN

IF v\_marks >= 90 THEN

v\_grade := 'A';

ELSIF v\_marks >= 80 THEN

v\_grade := 'B';

ELSIF v\_marks >= 70 THEN

v\_grade := 'C';

ELSE

v\_grade := 'F';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Student: ' || v\_student\_name || ', Marks: ' || v\_marks || ', Grade: ' || v\_grade);

END;

## Case Study 2: Bonus Calculation with Exception Handling

Raise a custom exception if bonus exceeds allowed cap.

DECLARE

v\_bonus NUMBER := 12000;

v\_max\_bonus NUMBER := 10000;

e\_bonus\_exceed EXCEPTION;

BEGIN

IF v\_bonus > v\_max\_bonus THEN

RAISE e\_bonus\_exceed;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Bonus Approved: ' || v\_bonus);

EXCEPTION

WHEN e\_bonus\_exceed THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Bonus exceeds the allowed limit.');

END;

## Case Study 3: Cursor Loop to Print Employees

DECLARE

CURSOR emp\_cur IS

SELECT emp\_id, emp\_name, salary FROM employees;

BEGIN

FOR emp IN emp\_cur LOOP

DBMS\_OUTPUT.PUT\_LINE('ID: ' || emp.emp\_id || ' - ' || emp.emp\_name || ', Salary: ' || emp.salary);

END LOOP;

END;

## Case Study 4: Parameterized Cursor for Departmental Listing

DECLARE

CURSOR emp\_by\_dept(p\_dept\_id NUMBER) IS

SELECT emp\_name FROM employees WHERE dept\_id = p\_dept\_id;

BEGIN

FOR rec IN emp\_by\_dept(10) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee in Dept 10: ' || rec.emp\_name);

END LOOP;

END;

Let me know if you'd like to add dry runs, visuals, or quiz-based activities for each!