**PL/SQL Overview**

PL/SQL (Procedural Language/SQL) is Oracle’s extension to SQL that allows procedural programming within the database. It supports variables, loops, conditions, functions, and error handling.

**Key Features**

* Portable across Oracle-supported platforms
* Efficient: Reduces network traffic by bundling SQL
* Error Handling: Built-in exception management
* Tool Support: Compatible with IDEs like Oracle SQL Developer
* Structured Programming: Supports modular code blocks

**🔹 Basic Block Structure**

**BEGIN**

**dbms\_output.put\_line('Welcome to PL/SQL');**

**END;**

**/**

**🔹 Conditional Logic: IF Statement**

**DECLARE**

**a1 NUMBER := 10;**

**b NUMBER := 100;**

**c1 NUMBER;**

**BEGIN**

**IF a1 > b THEN**

**c1 := a1;**

**ELSE**

**c1 := b;**

**END IF;**

**dbms\_output.put\_line('Maximum number in 10 and 100: ' || c1);**

**END;**

**/**

**🔹 Arithmetic Operations**

**DECLARE**

**n1 NUMBER := &n1;**

**n2 NUMBER := &n2;**

**sum NUMBER;**

**sub NUMBER;**

**div NUMBER;**

**mult NUMBER;**

**BEGIN**

**sum := n1 + n2;**

**sub := n1 - n2;**

**div := n1 / n2;**

**mult := n1 \* n2;**

**dbms\_output.put\_line('Addition: ' || sum);**

**dbms\_output.put\_line('Subtraction: ' || sub);**

**dbms\_output.put\_line('Multiplication: ' || mult);**

**dbms\_output.put\_line('Division: ' || div);**

**END;**

**/**

**🔹 Data Types in PL/SQL**

| **Type** | **Description** |
| --- | --- |
| **Scalar** | **NUMBER, CHAR, VARCHAR2, BOOLEAN** |
| **LOB** | **CLOB, BLOB, NCLOB, BFILE** |
| **Reference** | **REF CURSOR, REFS** |
| **Composite** | **Arrays, Records, Tables** |

**🔹 Constant Declaration**

**DECLARE**

**school\_name CONSTANT VARCHAR2(20) := 'DYP';**

**BEGIN**

**dbms\_output.put\_line('I study in ' || school\_name);**

**END;**

**/**

**🔹 Even or Odd Check**

**DECLARE**

**n1 NUMBER := &n1;**

**BEGIN**

**IF MOD(n1, 2) = 0 THEN**

**dbms\_output.put\_line(n1 || ' is even');**

**ELSE**

**dbms\_output.put\_line(n1 || ' is odd');**

**END IF;**

**END;**

**/**

**🔹 CASE Statement**

**DECLARE**

**n1 INT := &n1;**

**n2 INT;**

**BEGIN**

**n2 := MOD(n1, 2);**

**CASE n2**

**WHEN 0 THEN dbms\_output.put\_line(n1 || ' is even');**

**ELSE dbms\_output.put\_line(n1 || ' is odd');**

**END CASE;**

**END;**

**/**

**🔹 CASE with Operation**

**DECLARE**

**n1 INT := &n1;**

**n2 INT := &n2;**

**operation VARCHAR2(100) := '&operation';**

**BEGIN**

**CASE operation**

**WHEN 'sum' THEN dbms\_output.put\_line('Sum is: ' || (n1 + n2));**

**WHEN 'sub' THEN dbms\_output.put\_line('Subtraction is: ' || (n1 - n2));**

**ELSE dbms\_output.put\_line('No operation selected');**

**END CASE;**

**END;**

**/**

**🔹 Looping Examples**

**Print 1 to 10:**

**DECLARE**

**i INT := 1;**

**BEGIN**

**LOOP**

**EXIT WHEN i > 10;**

**dbms\_output.put\_line(i);**

**i := i + 1;**

**END LOOP;**

**END;**

**/**

**Table of 2:**

**DECLARE**

**i INT := 1;**

**num INT := 2;**

**BEGIN**

**LOOP**

**EXIT WHEN i > 10;**

**dbms\_output.put\_line(num \* i);**

**i := i + 1;**

**END LOOP;**

**END;**

**/**

**While Loop (Odd Numbers):**

**DECLARE**

**num INT := 1;**

**BEGIN**

**WHILE num <= 10 LOOP**

**dbms\_output.put\_line(num);**

**num := num + 2;**

**END LOOP;**

**END;**

**/**

**For Loop:**

**DECLARE**

**i NUMBER(2);**

**BEGIN**

**FOR i IN 1..10 LOOP**

**dbms\_output.put\_line(i);**

**END LOOP;**

**END;**

**/**

**Reverse For Loop:**

**DECLARE**

**i NUMBER(2);**

**BEGIN**

**FOR i IN REVERSE 1..10 LOOP**

**dbms\_output.put\_line(i);**

**END LOOP;**

**END;**

**/**

**🔹 Functions**

**Find Maximum:**

**DECLARE**

**a1 NUMBER := 10;**

**b1 NUMBER := 100;**

**c1 NUMBER;**

**FUNCTION findMax(x IN NUMBER, y IN NUMBER) RETURN NUMBER IS**

**z NUMBER;**

**BEGIN**

**IF x > y THEN**

**z := x;**

**ELSE**

**z := y;**

**END IF;**

**RETURN z;**

**END;**

**BEGIN**

**c1 := findMax(a1, b1);**

**dbms\_output.put\_line('Maximum number in 10 and 100 is : ' || c1);**

**END;**

**/**

**🔹 Procedures vs Functions**

* **Function returns a value using RETURN**
* **Procedure performs actions but does not return a value directly**

**🔹 Procedure Example**

**Create Table:**

**CREATE TABLE user5 (**

**id NUMBER(10) PRIMARY KEY,**

**name2 VARCHAR2(100)**

**);**

**Create Procedure:**

**CREATE OR REPLACE PROCEDURE insertuser3 (**

**id IN NUMBER,**

**name IN VARCHAR2**

**) IS**

**BEGIN**

**INSERT INTO user5 VALUES (id, name);**

**END;**

**/**

**Call Procedure:**

**BEGIN**

**insertuser3(101, 'RAHUL');**

**dbms\_output.put\_line('Record inserted successfully');**

**END;**

**/**

**🔹 Cursor Example**

**Create Table:**

**CREATE TABLE student (**

**rollno NUMBER(10) PRIMARY KEY,**

**sname VARCHAR2(100),**

**course VARCHAR2(100)**

**);**

**Insert Data:**

**BEGIN**

**INSERT INTO student VALUES (102, 'Amit', 'Python');**

**-- Add other records similarly**

**DBMS\_OUTPUT.PUT\_LINE('10 records inserted successfully');**

**END;**

**/**

**Cursor Block:**

**DECLARE**

**CURSOR showRec(sno student.rollno%TYPE) IS**

**SELECT sname, course FROM student WHERE rollno = sno;**

**a student.sname%TYPE;**

**b student.course%TYPE;**

**d NUMBER := :rollno;**

**BEGIN**

**OPEN showRec(d);**

**IF showRec%ISOPEN = FALSE THEN**

**dbms\_output.put\_line('Cannot open cursor');**

**ELSE**

**LOOP**

**FETCH showRec INTO a, b;**

**EXIT WHEN showRec%NOTFOUND;**

**dbms\_output.put\_line(a || ' ' || b);**

**END LOOP;**

**END IF;**

**CLOSE showRec;**

**END;**

**/**

**🔹 Trigger Example**

**CREATE OR REPLACE TRIGGER display\_salary\_changes**

**BEFORE DELETE OR INSERT OR UPDATE ON customer**

**FOR EACH ROW**

**WHEN (NEW.id > 0)**

**DECLARE**

**sal\_diff NUMBER;**

**BEGIN**

**sal\_diff := :NEW.salary - :OLD.salary;**

**dbms\_output.put\_line('Old salary : ' || :OLD.salary);**

**dbms\_output.put\_line('New salary : ' || :NEW.salary);**

**dbms\_output.put\_line('Salary Difference: ' || sal\_diff);**

**END;**

**/**

**🔹 Package Example**

**Package Spec:**

**CREATE OR REPLACE PACKAGE c\_package AS**

**PROCEDURE addCustomer(c\_id customer.id%TYPE,**

**c\_name customer.name%TYPE,**

**c\_age customer.age%TYPE,**

**c\_addr customer.address%TYPE,**

**c\_sal customer.salary%TYPE);**

**PROCEDURE delCustomer(c\_id customer.id%TYPE);**

**PROCEDURE listCustomer;**

**END c\_package;**

**/**

**Package Body:**

**CREATE OR REPLACE PACKAGE BODY c\_package AS**

**PROCEDURE addCustomer(...) IS**

**BEGIN**

**INSERT INTO customer (...) VALUES (...);**

**END addCustomer;**

**PROCEDURE delCustomer(...) IS**

**BEGIN**

**DELETE FROM customer WHERE id = c\_id;**

**END delCustomer;**

**END c\_package;**

**/**

**Using the Package:**

**DECLARE**

**code customer.id%TYPE := 8;**

**BEGIN**

**c\_package.addCustomer(17, 'Rajnish', 25, 'Chennai', 3500);**

**c\_package.delCustomer(code);**

**END;**

**/**