PLSQL

**What is PLSQL:**  
 PL/SQL (Procedural Language/Structured Query Language) is Oracle’s extension of SQL for writing procedural code.It allows you to write blocks of code with variables, loops, and error handling.

**Syntax**:

DECLARE

-- Declare variables here

BEGIN

-- Write your logic here

EXCEPTION

-- Handle errors here

END;

**STORED PROCEDURES:**

* A reusable block of code stored in the database.
* Can accept parameters and perform complex operations.

**Syntax:**

CREATE OR REPLACE PROCEDURE procedure\_name (parameter1 IN datatype, parameter2 OUT datatype) IS

BEGIN

-- Logic

END;

>DESC USER\_SOURCE ----------------🡪 TO VIEW TEXT

>DESC USER\_PROCEDURES----------🡪TO VIEW LIST OF PROCEDURES

>USER\_OBJECTS------------🡪TO CHECK PROCEDURES EXIST OR NOT

>SHOW ERRORS PROCEDURES INSERT\_EMP;-------🡪LIST ERRORS

**PL/SQL Study Notes: Day 1**

**Topics Covered:** PL/SQL Basics, Stored Procedures, and Practice Problems

**1. PL/SQL Block Structure**

PL/SQL code is organized into **blocks** with declarative, executable, and exception-handling sections.

**Syntax:**

DECLARE

-- Declare variables, cursors, etc. here

BEGIN

-- Main logic (SQL/PLSQL statements)

EXCEPTION

-- Error handling (optional)

END;

**Example:**

DECLARE

message VARCHAR2(50) := 'Hello, World!';

BEGIN

DBMS\_OUTPUT.PUT\_LINE(message);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

**2. Variables and Data Types**

* **Declaration:**

variable\_name DATATYPE [NOT NULL] := value;

Example:

DECLARE

total NUMBER := 0;

name VARCHAR2(100) := 'Alice';

* **Common Data Types:**
  + NUMBER: Numeric values.
  + VARCHAR2: Variable-length strings.
  + DATE: Dates and times.

**3. Loops**

FOR counter IN start..end LOOP

-- Statements

END LOOP;

**Example:** Sum of numbers from 1 to 10:

DECLARE

total NUMBER := 0;

BEGIN

FOR i IN 1..10 LOOP

total := total + i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Sum: ' || total); -- Output: Sum: 55

END;

**4. Stored Procedures**

Procedures are reusable blocks of code stored in the database.

**Syntax:**

CREATE OR REPLACE PROCEDURE procedure\_name (

parameter1 IN DATATYPE,

parameter2 OUT DATATYPE

) IS

-- Local variables

BEGIN

-- Logic

COMMIT; -- Save changes (optional)

END;

**Example:** Delete records older than a specific date:

CREATE OR REPLACE PROCEDURE delete\_old\_records (

cutoff\_date IN DATE

) IS

BEGIN

DELETE FROM employees WHERE hire\_date < cutoff\_date;

COMMIT; -- Finalize the transaction

END;

**Calling a Procedure:**

BEGIN

delete\_old\_records(TO\_DATE('2022-01-01', 'YYYY-MM-DD'));

END;

**5. Key Functions**

* **DBMS\_OUTPUT.PUT\_LINE:** Prints output to the console.
  + Enable it first:
  + SET SERVEROUTPUT ON;
* **COMMIT:** Saves changes permanently (used in DML operations).

**6. Practice Problems & Solutions**

**Problem 1:** Find the second highest salary in the employees table.  
**Solution:**

DECLARE

second\_highest NUMBER;

BEGIN

SELECT MAX(salary) INTO second\_highest

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees);

DBMS\_OUTPUT.PUT\_LINE('Second Highest Salary: ' || second\_highest);

END;

**Problem 2:** Update an employee’s salary by a percentage using a procedure.  
**Solution:**

CREATE OR REPLACE PROCEDURE update\_salary (

emp\_id IN NUMBER,

percentage IN NUMBER

) IS

BEGIN

UPDATE employees

SET salary = salary \* (1 + percentage/100)

WHERE id = emp\_id;

COMMIT;

END;

**7. Troubleshooting Tips**

1. **Enable Output:** Always run SET SERVEROUTPUT ON; before executing PL/SQL blocks.
2. **Check Variable Initialization:** Ensure variables are initialized (e.g., total NUMBER := 0;).
3. **Test with Sample Data:** Use a test table to validate your code.

**Key Takeaways**

1. PL/SQL blocks combine SQL with procedural logic (loops, conditions).
2. Use **stored procedures** for reusable tasks like data cleanup or calculations.
3. Always **commit transactions** (COMMIT) for DML operations (INSERT/UPDATE/DELETE).
4. Debug using DBMS\_OUTPUT.PUT\_LINE to print intermediate values.