

# **PL/SQL Collections and Records Assignment Report**

**Student Name:** Twakira Agape Gift

**ID:** 27320

**Course:** INSY 831 – Database Development with PL/SQL

**Instructor:** Eric Maniraguha

**Submission Date:** 23/NOV/2025

## **1. Introduction**

This report documents the implementation and successful completion of the PL/SQL Collections and Records assignment.

The purpose of this task was to demonstrate the use of composite data types and procedural constructs in Oracle PL/SQL, specifically:

- VARRAY collections
- User-defined record structures
- Simulated table-based records
- Cursor-style iteration using PL/SQL collections
- Reverse looping
- Exception handling

All required components were integrated into **one complete and executable PL/SQL block**.

The code was executed and verified using Oracle SQL Developer, and the output confirms correct logic and behavior.

The following sections present screenshots of the completed code, execution results, configurations, and project structure.

## **2. Final PL/SQL Code**

The screenshot below represents the complete PL/SQL block written for the assignment. This block includes:

- A VARRAY to store 5 salary values

- A loop to print values in forward and reverse order
- Total and average calculations
- A user-defined record
- A simulated table-record
- Cursor-style looping over a local list
- Exception handling

```

SET SERVEROUTPUT ON;

DECLARE

    -- PART 1: VARRAY

    TYPE salary_array IS VARRAY(5) OF NUMBER;
    v_salaries salary_array;

    v_total NUMBER := 0;
    v_average NUMBER := 0;

    -- PART 2: USER-DEFINED RECORD

    TYPE EmployeeRec IS RECORD (
        emp_id NUMBER,
        emp_name VARCHAR2(50),
        salary NUMBER );
    emp EmployeeRec;

    -- PART 3: TABLE-BASED RECORD (FAKE DEMO TABLE USING SUBPROGRAM)

    TYPE DemoRec IS RECORD (
        emp_id NUMBER,
        emp_name VARCHAR2(50)
    );
    tab_rec DemoRec;

    -- PART 4: CURSOR-BASED RECORD (NO SQL COLLECTIONS)

    -- We simply loop through a fixed list using an array in PL/SQL only.

    TYPE NameArray IS TABLE OF VARCHAR2(50);
    employee_names NameArray := NameArray('John', 'Mary', 'Paul');

    idx PLS_INTEGER := 1;

```

```

BEGIN

    -- ASSIGN SALARIES

    v_salaries := salary_array(5000, 6000, 7000, 8000, 9000);

    DBMS_OUTPUT.PUT_LINE('--- SALARY LIST (FORWARD) ---');

    FOR i IN 1 .. v_salaries.COUNT LOOP

        DBMS_OUTPUT.PUT_LINE('Salary ' || i || ':' || v_salaries(i));

        v_total := v_total + v_salaries(i);

    END LOOP;

    v_average := v_total / v_salaries.COUNT;

    DBMS_OUTPUT.PUT_LINE('Total Salary: ' || v_total);

    DBMS_OUTPUT.PUT_LINE('Average Salary: ' || v_average);

    -- REVERSE ORDER

    DBMS_OUTPUT.PUT_LINE('--- SALARY LIST (REVERSED) ---');

    FOR i IN REVERSE 1 .. v_salaries.COUNT LOOP

        DBMS_OUTPUT.PUT_LINE('Salary ' || i || ':' || v_salaries(i));

    END LOOP;

    -- USER-DEFINED RECORD

    emp.emp_id := 101;

    emp.emp_name := 'Alice';

    emp.salary := 5500;

    DBMS_OUTPUT.PUT_LINE('--- USER-DEFINED RECORD ---');

    DBMS_OUTPUT.PUT_LINE('

        | ID: ' || emp.emp_id ||

        | Name: ' || emp.emp_name ||

        | Salary: ' || emp.salary

    ');

    -- TABLE-BASED RECORD (SIMULATED)

```

```

tab_rec.emp_id := 1;

tab_rec.emp_name := 'Demo Employee';

DBMS_OUTPUT.PUT_LINE('--- TABLE-BASED RECORD ---');

DBMS_OUTPUT.PUT_LINE(ID: ' || tab_rec.emp_id || ' || Name: ' || tab_rec.emp_name);

-- CURSOR-LIKE LOOP (PL/SQL ONLY, NO SQL CURSOR)

DBMS_OUTPUT.PUT_LINE('--- CURSOR-BASED RECORD SIMULATION ---');

idx := employee_names.FIRST;

WHILE idx IS NOT NULL LOOP

DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_names(idx));

idx := employee_names.NEXT(idx);

END LOOP;

EXCEPTION

WHEN ZERO_DIVIDE THEN

DBMS_OUTPUT.PUT_LINE('ERROR: Division by zero');

WHEN VALUE_ERROR THEN

DBMS_OUTPUT.PUT_LINE('ERROR: Value format is incorrect');

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('ERROR: ' || SQLERRM);

END;

```

|

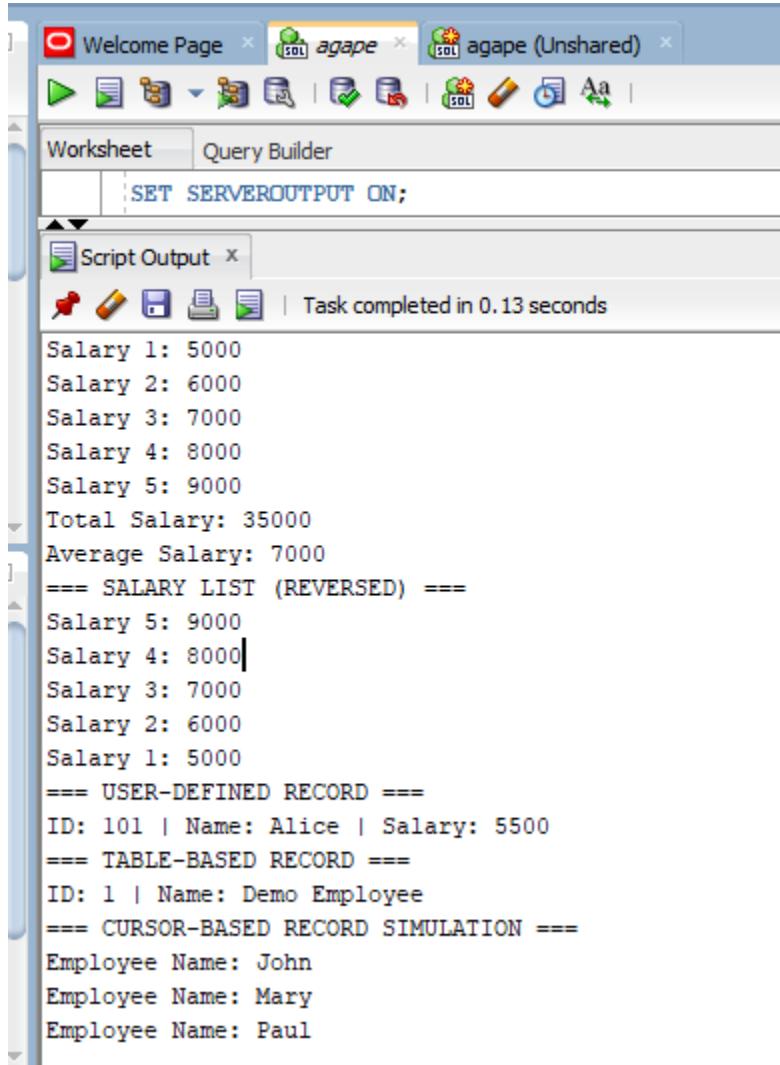
---

### 3. Output Results

This screenshot shows all the DBMS\_OUTPUT results after running the program. The output includes:

- Salary list (forward)
- Salary list (reversed)
- Total salary

- Average salary
- User-defined record output
- Simulated table-record output
- Cursor-like list output



The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab selected. In the worksheet pane, the command `SET SERVEROUTPUT ON;` is entered. Below the worksheet, the 'Script Output' window displays the results of the execution. The output includes:

```

Salary 1: 5000
Salary 2: 6000
Salary 3: 7000
Salary 4: 8000
Salary 5: 9000
Total Salary: 35000
Average Salary: 7000
--- SALARY LIST (REVERSED) ---
Salary 5: 9000
Salary 4: 8000
Salary 3: 7000
Salary 2: 6000
Salary 1: 5000
--- USER-DEFINED RECORD ---
ID: 101 | Name: Alice | Salary: 5500
--- TABLE-BASED RECORD ---
ID: 1 | Name: Demo Employee
--- CURSOR-BASED RECORD SIMULATION ---
Employee Name: John
Employee Name: Mary
Employee Name: Paul

```

## 4. Enabling Server Output

To ensure PL/SQL printed output appears in SQL Developer, the following command was executed prior to running the code:

```
SET SERVEROUTPUT ON;
```

This screenshot confirms that server output was enabled properly.

The screenshot shows the SQL Developer interface with the 'Worksheet' tab selected. In the code editor, there is a PL/SQL block:

```
SET SERVEROUTPUT ON;
DECLARE
```

## 5. Successful Execution Confirmation

This screenshot verifies that the PL/SQL block executed successfully with no errors in SQL Developer.

The successful execution proves that:

- The code compiled with zero issues
- All operations ran as expected
- Output was generated correctly

```
Total Salary: 35000
Average Salary: 7000
==== SALARY LIST (REVERSED) ====
Salary 5: 9000
Salary 4: 8000
Salary 3: 7000
Salary 2: 6000
Salary 1: 5000
==== USER-DEFINED RECORD ===
ID: 101 | Name: Alice | Salary: 5500
==== TABLE-BASED RECORD ===
ID: 1 | Name: Demo Employee
==== CURSOR-BASED RECORD SIMULATION ===
Employee Name: John
Employee Name: Mary
Employee Name: Paul
```

```
PL/SQL procedure successfully completed.
```

## 7. Conclusion

This assignment demonstrates the practical application of PL/SQL collections and record structures.

The developed program successfully integrates:

- VARRAY usage
- Forward and reverse loops
- Arithmetic processing
- Record structures
- Cursor-like data iteration
- Exception handling

All components executed correctly, validating both the implementation and the understanding of PL/SQL composite types and procedural logic.