**🔹 Coding Challenge (Wk 2 Day 2)**

**Problem Statement 1: Employee Salary Increment Processor**

**Scenario:**

Your company performs annual salary hikes. You need to write a PL/SQL block to:

* Accept an employee’s name and current salary
* Apply an increment based on the following rules:
  + Salary < 30,000 → 20% hike
  + 30,000 ≤ Salary < 50,000 → 15% hike
  + Salary ≥ 50,000 → 10% hike
* Display the old salary, new salary, and employee name.

**Solution:**

DECLARE

v\_emp\_name VARCHAR2(50) := 'Sneha';

v\_salary NUMBER := 28000;

v\_new\_salary NUMBER;

BEGIN

IF v\_salary < 30000 THEN

v\_new\_salary := v\_salary \* 1.20;

ELSIF v\_salary < 50000 THEN

v\_new\_salary := v\_salary \* 1.15;

ELSE

v\_new\_salary := v\_salary \* 1.10;

END IF;

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

DBMS\_OUTPUT.PUT\_LINE('Old Salary: ' || v\_salary);

DBMS\_OUTPUT.PUT\_LINE('New Salary: ' || ROUND(v\_new\_salary, 2));

END;

**Expected Output:**

Employee: Sneha

Old Salary: 28000

New Salary: 33600

**Problem Statement 2: Inventory Price Tracker Using Collections**

**Scenario:**

You are managing inventory prices using three collection types:

* **Associative Array** for item names and current prices.
* **VARRAY** for storing fixed category discount limits.
* **Nested Table** for storing actual discount values applied.

**Solution:**

DECLARE

-- Associative Array: Item Prices

TYPE item\_price\_map IS TABLE OF NUMBER INDEX BY VARCHAR2(30);

item\_prices item\_price\_map;

-- VARRAY: Discount Caps per Category

TYPE discount\_cap\_array IS VARRAY(3) OF NUMBER;

discount\_caps discount\_cap\_array := discount\_cap\_array(20, 15, 10); -- Percent

-- Nested Table: Applied Discounts

TYPE discount\_table\_type IS TABLE OF NUMBER;

applied\_discounts discount\_table\_type := discount\_table\_type(12.5, 7.5, 15);

BEGIN

-- Associative Array values

item\_prices('Rice') := 1200;

item\_prices('Toothpaste') := 90;

item\_prices('Laptop') := 45000;

-- Print item prices

DBMS\_OUTPUT.PUT\_LINE('Item: Rice → Price: ' || item\_prices('Rice'));

DBMS\_OUTPUT.PUT\_LINE('Item: Toothpaste → Price: ' || item\_prices('Toothpaste'));

DBMS\_OUTPUT.PUT\_LINE('Item: Laptop → Price: ' || item\_prices('Laptop'));

-- Print Discount Caps

FOR i IN 1..discount\_caps.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Discount Cap %: ' || discount\_caps(i));

END LOOP;

-- Print Applied Discounts

FOR i IN 1..applied\_discounts.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Applied Discount %: ' || applied\_discounts(i));

END LOOP;

END;

**Expected Output:**

Item: Rice → Price: 1200

Item: Toothpaste → Price: 90

Item: Laptop → Price: 45000

Discount Cap %: 20

Discount Cap %: 15

Discount Cap %: 10

Applied Discount %: 12.5

Applied Discount %: 7.5

Applied Discount %: 15

**What Learners Practice:**

| **Concept** | **Covered In** |
| --- | --- |
| Scalar variables and conditions | Problem 1 |
| Anchored datatypes (optional) | Modify for %TYPE in salary block |
| Associative Arrays | Item prices |
| VARRAYs | Category discount caps |
| Nested Tables | Discount history or records |
| Looping & DBMS\_OUTPUT | Both scenarios |

Solution of learner file :

## ****Problem Statement 1: Student Grading System****

plsql

CopyEdit

DECLARE

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

v\_marks NUMBER := 87;

v\_grade CHAR(1);

BEGIN

-- Conditional grading logic

IF v\_marks >= 90 THEN

v\_grade := 'A';

ELSIF v\_marks BETWEEN 80 AND 89 THEN

v\_grade := 'B';

ELSIF v\_marks BETWEEN 70 AND 79 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;

✅ **Output:**

yaml

CopyEdit

Student: Riya, Marks: 87, Grade: B

## ****Problem Statement 2: Department Budget Tracker****

DECLARE

-- Associative Array: Department Budgets

TYPE dept\_budget\_type IS TABLE OF NUMBER INDEX BY VARCHAR2(20);

dept\_budgets dept\_budget\_type;

-- VARRAY: Fixed Department Limits

TYPE budget\_limit\_array IS VARRAY(3) OF NUMBER;

budget\_limits budget\_limit\_array := budget\_limit\_array(100000, 75000, 90000);

-- Nested Table: Actual Expense Entries

TYPE expense\_table\_type IS TABLE OF NUMBER;

expenses expense\_table\_type := expense\_table\_type(48000, 62000, 50000);

-- For loop counter

i PLS\_INTEGER;

BEGIN

-- Assign values to associative array

dept\_budgets('IT') := 50000;

dept\_budgets('HR') := 40000;

dept\_budgets('Finance') := 60000;

-- Output from associative array

DBMS\_OUTPUT.PUT\_LINE('Department: IT → Budget: ' || dept\_budgets('IT'));

DBMS\_OUTPUT.PUT\_LINE('Department: HR → Budget: ' || dept\_budgets('HR'));

DBMS\_OUTPUT.PUT\_LINE('Department: Finance → Budget: ' || dept\_budgets('Finance'));

-- Output from VARRAY

FOR i IN 1 .. budget\_limits.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('VARRAY Limit: ' || budget\_limits(i));

END LOOP;

-- Output from Nested Table

FOR i IN 1 .. expenses.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Expense Entry: ' || expenses(i));

END LOOP;

END;

✅ **Expected Output:**

yaml

CopyEdit

Department: IT → Budget: 50000

Department: HR → Budget: 40000

Department: Finance → Budget: 60000

VARRAY Limit: 100000

VARRAY Limit: 75000

VARRAY Limit: 90000

Expense Entry: 48000

Expense Entry: 62000

Expense Entry: 50000