CREATE OR REPLACE PROCEDURE update\_pol\_allocation (

p\_pol\_grade IN VARCHAR2,

p\_regular\_allocation IN NUMBER,

p\_additional\_allocation IN NUMBER

)

IS

v\_existing\_regular\_alloc NUMBER;

v\_existing\_additional\_alloc NUMBER;

v\_total\_allocation NUMBER;

v\_remaining\_pol NUMBER;

BEGIN

-- Retrieve the current allocations and remaining POL for the given POL\_Grade

SELECT Regular\_Allocation, Additional\_Allocation, Remaining\_Pol

INTO v\_existing\_regular\_alloc, v\_existing\_additional\_alloc, v\_remaining\_pol

FROM POL

WHERE POL\_Grade = p\_pol\_grade

FOR UPDATE;

-- Update the regular and additional allocations by adding new amounts

v\_existing\_regular\_alloc := v\_existing\_regular\_alloc + p\_regular\_allocation;

v\_existing\_additional\_alloc := v\_existing\_additional\_alloc + p\_additional\_allocation;

-- Calculate the new total allocation

v\_total\_allocation := v\_existing\_regular\_alloc + v\_existing\_additional\_alloc;

-- Update Remaining\_Pol by adding the new total allocation

v\_remaining\_pol := v\_remaining\_pol + (p\_regular\_allocation + p\_additional\_allocation);

-- Update the POL table with the new values

UPDATE POL

SET Regular\_Allocation = v\_existing\_regular\_alloc,

Additional\_Allocation = v\_existing\_additional\_alloc,

Total\_Allocation = v\_total\_allocation,

Remaining\_Pol = v\_remaining\_pol

WHERE POL\_Grade = p\_pol\_grade;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No record found for POL\_Grade: ' || p\_pol\_grade);

WHEN OTHERS THEN

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

END;

create or replace PROCEDURE update\_vdra\_km (

p\_bano IN VDRA.BANo%TYPE,

p\_vdra\_date IN VDRA.VDRA\_Date%TYPE,

p\_route IN VDRA.Route%TYPE,

p\_km\_reading IN VDRA.KM\_Reading%TYPE

)

IS

-- Declare local variables

v\_previous\_km VDRA.KM\_Reading%TYPE := 0;

v\_kpl Vehicle.KPL%TYPE := 0;

v\_pol\_used NUMBER := 0;

v\_tank\_state VDRA.Tank\_State%TYPE := NULL;

BEGIN

-- Step 1: Get the max KM\_Reading for the vehicle (BANo)

BEGIN

SELECT MAX(KM\_Reading)

INTO v\_previous\_km

FROM VDRA

WHERE BANo = p\_bano;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- If no previous KM\_Reading found, set to default

v\_previous\_km := 0;

END;

-- Step 2: Get the KPL for the vehicle from the Vehicle table

BEGIN

SELECT KPL

INTO v\_kpl

FROM Vehicle

WHERE BANo = p\_bano;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- If no KPL found, set to default

v\_kpl := 0;

END;

-- Step 3: Get the last non-null Tank\_State for the vehicle (BANo) with max VDRA\_Date

BEGIN

SELECT Tank\_State

INTO v\_tank\_state

FROM (

SELECT Tank\_State

FROM VDRA

WHERE BANo = p\_bano

AND Tank\_State IS NOT NULL

ORDER BY VDRA\_Date DESC

)

WHERE ROWNUM = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- If no previous Tank\_State found, set to NULL

v\_tank\_state := NULL;

END;

-- Step 4: Calculate the POL\_Used value

IF v\_kpl != 0 THEN

v\_pol\_used := (p\_km\_reading - v\_previous\_km) / v\_kpl; -- Calculate POL used

ELSE

v\_pol\_used := 0; -- Avoid division by zero

END IF;

-- Step 5: Deduct POL\_Used from Tank\_State, ensuring it's not negative

IF v\_tank\_state IS NOT NULL THEN

v\_tank\_state := v\_tank\_state - v\_pol\_used;

IF v\_tank\_state < 0 THEN

v\_tank\_state := 0; -- Ensure Tank\_State is never negative

END IF;

END IF;

-- Step 6: Update the Vehicle table's KM\_Reading

UPDATE Vehicle

SET KM\_Reading = p\_km\_reading

WHERE BANo = p\_bano;

-- Step 7: Insert or update the VDRA record with the new values

MERGE INTO VDRA v

USING (SELECT p\_bano AS BANo, p\_vdra\_date AS VDRA\_Date FROM dual) src

ON (v.BANo = src.BANo AND v.VDRA\_Date = src.VDRA\_Date)

WHEN MATCHED THEN

UPDATE SET v.KM\_Reading = p\_km\_reading, v.POL\_Used = v\_pol\_used, v.Tank\_State = v\_tank\_state

WHEN NOT MATCHED THEN

INSERT (BANo, VDRA\_Date, Route, KM\_Reading, POL\_Used, Tank\_State)

VALUES (p\_bano, p\_vdra\_date, p\_route, p\_km\_reading, v\_pol\_used, v\_tank\_state);

END update\_vdra\_km;