-- Trigger to generate POL\_ID for POL table

CREATE OR REPLACE TRIGGER trg\_generate\_pol\_id

BEFORE INSERT ON POL

FOR EACH ROW

BEGIN

:NEW.POL\_ID := 'P' || TO\_CHAR(pol\_seq.NEXTVAL, 'FM0000');

END;

/

-- Trigger to generate IssueID for POL\_ISSUE table

CREATE OR REPLACE TRIGGER trg\_generate\_pol\_issue\_id

BEFORE INSERT ON POL\_ISSUE

FOR EACH ROW

BEGIN

:NEW.IssueID := 'Issue' || TO\_CHAR(pol\_issue\_seq.NEXTVAL, 'FM0000');

END;

/

-- Trigger to generate RouteID for ROUTE table

CREATE OR REPLACE TRIGGER trg\_generate\_route\_id

BEFORE INSERT ON ROUTE

FOR EACH ROW

BEGIN

:NEW.RouteID := 'R' || TO\_CHAR(route\_seq.NEXTVAL, 'FM0000');

END;

/

-- Trigger to generate DriverVehicleRouteID for DRIVER\_VEHICLE\_ROUTE table

CREATE OR REPLACE TRIGGER trg\_generate\_dvr\_id

BEFORE INSERT ON DRIVER\_VEHICLE\_ROUTE

FOR EACH ROW

BEGIN

:NEW.DriverVehicleRouteID := 'DVR' || TO\_CHAR(driver\_vehicle\_route\_seq.NEXTVAL, 'FM0000');

END;

/

-- Trigger to generate MaintenanceID for MAINTENANCE table

CREATE OR REPLACE TRIGGER trg\_generate\_maintenance\_id

BEFORE INSERT ON MAINTENANCE

FOR EACH ROW

BEGIN

:NEW.MaintenanceID := 'MNT' || TO\_CHAR(maintenance\_seq.NEXTVAL, 'FM0000');

END;

/

-- Trigger to generate AccidentID for ACCIDENT table

CREATE OR REPLACE TRIGGER trg\_generate\_accident\_id

BEFORE INSERT ON ACCIDENT

FOR EACH ROW

BEGIN

:NEW.AccidentID := 'ACC' || TO\_CHAR(accident\_seq.NEXTVAL, 'FM0000');

END;

/

CREATE OR REPLACE TRIGGER trg\_update\_availability

AFTER INSERT ON DRIVER\_VEHICLE\_ROUTE

FOR EACH ROW

BEGIN

UPDATE DRIVER

SET Availability = 'Unavailable'

WHERE DriverID = :NEW.DriverID;

UPDATE VEHICLE

SET Availability = 'Unavailable',

DriverID = :NEW.DriverID -- Update DriverID in the VEHICLE table

WHERE BANO = :NEW.VehicleID;

END;

/

CREATE OR REPLACE TRIGGER trg\_revert\_availability

AFTER DELETE ON DRIVER\_VEHICLE\_ROUTE

FOR EACH ROW

BEGIN

UPDATE DRIVER

SET Availability = 'Available'

WHERE DriverID = :OLD.DriverID;

UPDATE VEHICLE

SET Availability = 'Available',

DriverID = NULL -- Optionally set DriverID to NULL when the driver is available again

WHERE BANO = :OLD.VehicleID;

END;

/

create or replace TRIGGER trg\_handle\_pol\_issue

AFTER INSERT ON POL\_ISSUE

FOR EACH ROW

DECLARE

v\_count NUMBER := 0;

v\_tank\_state NUMBER := 0;

BEGIN

-- Check if a row exists in the VDRA table for the given BANo and VDRA\_Date

SELECT COUNT(\*)

INTO v\_count

FROM VDRA

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

-- Fetch the most recent non-null Tank\_State for Diesel

IF :NEW.POL\_Grade = 'Diesel' THEN

BEGIN

SELECT NVL(Tank\_State, 0) INTO v\_tank\_state

FROM VDRA

WHERE BANo = :NEW.VehicleID

AND VDRA\_Date = (

SELECT MAX(VDRA\_Date)

FROM VDRA

WHERE BANo = :NEW.VehicleID

AND Tank\_State IS NOT NULL

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_tank\_state := 0; -- Default tank state if no prior data exists

END;

END IF;

IF v\_count > 0 THEN

-- Update the existing row with the new POL\_Grade amount

IF :NEW.POL\_Grade = 'Diesel' THEN

UPDATE VDRA

SET Diesel = NVL(Diesel, 0) + :NEW.IssueAmount,

Tank\_State = v\_tank\_state + :NEW.IssueAmount -- Update tank state with new amount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'Hd\_30' THEN

UPDATE VDRA

SET Hd\_30 = NVL(Hd\_30, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'GX\_90' THEN

UPDATE VDRA

SET GX\_90 = NVL(GX\_90, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'K2' THEN

UPDATE VDRA

SET K2 = NVL(K2, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'Greese' THEN

UPDATE VDRA

SET Greese = NVL(Greese, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'Break\_Fluid' THEN

UPDATE VDRA

SET Break\_Fluid = NVL(Break\_Fluid, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'MS-74' THEN

UPDATE VDRA

SET MS\_74 = NVL(MS\_74, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

ELSIF :NEW.POL\_Grade = 'Octen\_100' THEN

UPDATE VDRA

SET Octen\_100 = NVL(Octen\_100, 0) + :NEW.IssueAmount

WHERE BANo = :NEW.VehicleID AND VDRA\_Date = :NEW.IssueDate;

END IF;

ELSE

-- If no existing row, insert a new row into the VDRA table with the appropriate POL\_Grade

IF :NEW.POL\_Grade = 'Diesel' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, Diesel, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, :NEW.IssueAmount + v\_tank\_state); -- Include fetched Tank\_State

ELSIF :NEW.POL\_Grade = 'Hd\_30' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, Hd\_30, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'GX\_90' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, GX\_90, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'K2' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, K2, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'Greese' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, Greese, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'Break\_Fluid' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, Break\_Fluid, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'MS-74' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, MS\_74, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

ELSIF :NEW.POL\_Grade = 'Octen\_100' THEN

INSERT INTO VDRA (BANo, VDRA\_Date, Octen\_100, Tank\_State)

VALUES (:NEW.VehicleID, :NEW.IssueDate, :NEW.IssueAmount, NULL);

END IF;

END IF;

-- Update the POL table with the issued amount

UPDATE POL

SET Expense\_Running\_Month = Expense\_Running\_Month + :NEW.IssueAmount,

Expense\_Total = Expense\_Total + :NEW.IssueAmount,

Remaining\_Pol = Remaining\_Pol - :NEW.IssueAmount

WHERE POL\_ID = :NEW.POL\_ID;

END;

/

CREATE OR REPLACE TRIGGER trg\_generate\_user\_id

BEFORE INSERT ON USERS

FOR EACH ROW

BEGIN

IF :NEW.UserID IS NULL THEN

:NEW.UserID := user\_seq.NEXTVAL;

END IF;

END;

/