CREATE OPERATION

```
CREATE TABLE Roles (
   Role_ID INT PRIMARY KEY,
   Role_Name VARCHAR2(255)
);

CREATE TABLE ADMIN (
   ID INT PRIMARY KEY,
   Role_ID INT,
   Designation_ID INT PRIMARY KEY,
   Name VARCHAR2(255),
   FOREIGN KEY (Roles_ID) REFERENCES Roles(Role_ID)
);

CREATE TABLE RTO (
   ID INT PRIMARY KEY,
   Designation_ID INT,
   Name VARCHAR2(255),
   FOREIGN KEY (Designation_ID) REFERENCES ADMIN(Designation_ID)
);

CREATE TABLE Police (
   ID INT PRIMARY KEY,
   Designation_ID INT,
   Name VARCHAR2(255),
   FOREIGN KEY (Designation_ID) REFERENCES ADMIN(Designation_ID)
);

CREATE TABLE Clerk (
   ID INT PRIMARY KEY,
   Designation_ID INT,
   Name VARCHAR2(255),
   FOREIGN KEY (Designation_ID) REFERENCES ADMIN(Designation_ID)
);


CREATE TABLE Owner (
   Owner_ID INT PRIMARY KEY,
   Name VARCHAR2(255),
   Aadhaar INT,
   Vehicle_ID INT,
   Address VARCHAR2(255),
```

```
    Phone INT,
    FOREIGN KEY (Vehicle_ID) REFERENCES VEHICLE(Vehicle_ID),
);

CREATE TABLE Vehicle (
    Vehicle_ID INT PRIMARY KEY,
    Vehicle_Type VARCHAR2(255),
    Vehicle_Model VARCHAR2(255),
    Manufactured_Date DATE,
    Registration_ID INT,
    Owner_ID INT,
    FOREIGN KEY (Registration_ID) REFERENCES REGISTRATION(Registration_ID),
    FOREIGN KEY (Owner_ID) REFERENCES Owner(Owner_ID)
);

CREATE TABLE Registration (
    Registration_ID INT PRIMARY KEY,
    Registration_Date DATE,
    Location VARCHAR2(255),
    Vehicle_ID INT,
    Owner_ID INT,
    FOREIGN KEY (Vehicle_ID) REFERENCES Vehicle(Vehicle_ID),
    FOREIGN KEY (Owner_ID) REFERENCES Owner(Owner_ID)
);

CREATE TABLE Offence (
    Offence_ID INT PRIMARY KEY,
    Offence_Type VARCHAR2(255),
    Vehicle_Type VARCHAR2(255),
    Penalty INT
);

CREATE TABLE Vehicle_Offence (
    ID INT PRIMARY KEY,
    Offence_ID INT,
    Registration_ID INT,
    Location VARCHAR2(255),
    Status BINARY_INTEGER,
    Proof BLOB,
    FOREIGN KEY (Offence_ID) REFERENCES Offence(Offence_ID),
    FOREIGN KEY (Registration_ID) REFERENCES Registration(Registration_ID)
);
```

VIEW OPERATION

1. Create View for Report

```
CREATE OR REPLACE VIEW Report_View AS
SELECT
   vo.ID AS Report_ID,
   c.ID AS Clerk_ID,
   r.Registration_ID,
   o.Name AS Owner_Name,
   vo.ID AS Vehicle_Offence_ID,
   vo.Proof AS Report
FROM
   Clerk c
   JOIN Registration r ON c.Designation_ID = r.Owner_ID
   JOIN Owner o ON r.Owner_ID = o.Owner_ID
   JOIN Vehicle_Offence vo ON r.Registration_ID = vo.Registration_ID;
```

2. Create View for Registration Table

```
CREATE OR REPLACE VIEW vw_registration_info AS
SELECT
   R.ID,  -- Assuming Registration table has ID
   A.Designation_ID,
   R.Registration_ID AS Regis_ID,
   R.Owner_ID,
   O.Aadhaar
FROM
   Registration R
JOIN
   Admin A ON A.Designation_ID = R.Owner_ID  -- If this is the relation, confirm this
JOIN
   Owner O ON O.Owner_ID = R.Owner_ID;
```

3. Create View for Vehicle Offence Table

```
CREATE OR REPLACE VIEW OffenceView AS
SELECT
   vo.ID AS seq_offenceId,
   a.Designation_ID AS designationId,
   r.Registration_ID AS regis_Id,
   vo.Location AS location,
   vo.Status AS status,
   vo.Proof AS proof
FROM
   Vehicle_Offence vo
```

```
JOIN
   Registration r ON vo.Registration_ID = r.Registration_ID
JOIN
   Admin a ON a.Designation_ID IN (
      SELECT Designation_ID FROM Police
      UNION
      SELECT Designation_ID FROM Clerk
   );
```

SEQUENCES

```
CREATE SEQUENCE seq_role START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_owner START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_vehicle START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_registration START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_offence START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_police START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_clerk START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_rto START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_admin START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_report START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seq_vehicle_offence START WITH 1 INCREMENT BY 1;
```

STORED PROCEDURE

a) **View All Details of Report**

```
CREATE OR REPLACE PROCEDURE View_Report_Details AS
BEGIN
   FOR rec IN (SELECT * FROM Report_View) LOOP
      DBMS_OUTPUT.PUT_LINE(
         'Report ID: ' || rec.Report_ID ||
         ', Clerk ID: ' || rec.Clerk_ID ||
         ', Reg ID: ' || rec.Registration_ID ||
         ', Owner Name: ' || rec.Owner_Name ||
         ', Offence ID: ' || rec.Vehicle_Offence_ID
      );
   END LOOP;
END; /
```

b) **Insert report entry**

```
CREATE OR REPLACE PROCEDURE Insert_Report(
   p_reg_id IN INT,
   p_report IN BLOB
) AS
   v_owner_name VARCHAR2(255);
   v_vo_id INT;
BEGIN
   -- Get Owner Name
   SELECT o.Name INTO v_owner_name
   FROM Registration r
   JOIN Owner o ON r.Owner_ID = o.Owner_ID
   WHERE r.Registration_ID = p_reg_id;

   -- Get Vehicle Offence ID
   SELECT ID INTO v_vo_id
   FROM Vehicle_Offence
   WHERE Registration_ID = p_reg_id;

   -- Update report blob into Vehicle_Offence
   UPDATE Vehicle_Offence
   SET Proof = p_report
   WHERE ID = v_vo_id;

   DBMS_OUTPUT.PUT_LINE('Report inserted successfully for Owner: ' || v_owner_name);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Invalid Registration ID or no offence found.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

BEGIN
   Insert_Report_For_All_Offences(101, EMPTY_BLOB());
END;
/
```

c) **Add Roles in Roles table**

```
CREATE OR REPLACE PROCEDURE AddRole(p_role_name IN VARCHAR2) AS
   v_role_id INT;
BEGIN
   v_role_id := seq_role.NEXTVAL;

   INSERT INTO Roles(Role_ID, Role_Name)
   VALUES (v_role_id, p_role_name);

   DBMS_OUTPUT.PUT_LINE('Role added: ' || p_role_name || ' with Role_ID: ' || v_role_id);
END;
/

BEGIN
   AddRole('RTO');
   AddRole('PLC');
   AddRole('CLK');
END;
/
```

d) **AddDesignation in Admin table**

```
CREATE OR REPLACE PROCEDURE AddDesignation(
   p_name IN VARCHAR2,
   p_role_id IN INT
) AS
   v_id INT;
   v_role_name VARCHAR2(50);
   v_count INT;
   v_designation_id VARCHAR2(50);
BEGIN
   -- Get role name for given role_id
   SELECT Role_Name INTO v_role_name
   FROM Roles
   WHERE Role_ID = p_role_id;

   -- Count existing designations with the same role prefix
   SELECT COUNT(*) + 1 INTO v_count
   FROM ADMIN a
   JOIN Roles r ON a.Role_ID = r.Role_ID
   WHERE r.Role_Name = v_role_name;

   -- Generate Designation_ID like 'RTO01'
   v_designation_id := v_role_name || LPAD(v_count, 2, '0');
```

```
    -- Generate ID using sequence
    v_id := seq_admin.NEXTVAL;

    -- Insert into ADMIN
    INSERT INTO ADMIN(ID, Role_ID, Designation_ID, Name)
    VALUES (v_id, p_role_id, v_designation_id, p_name);

    DBMS_OUTPUT.PUT_LINE('Admin added: ' || p_name || ' with Designation_ID: ' ||
v_designation_id);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Invalid Role_ID: ' || p_role_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

BEGIN
    AddDesignation('Rahul Kumar', 1); -- Will assign like RTO01
    AddDesignation('Priya Sharma', 1); -- Will assign like RTO02
    AddDesignation('Vikram Singh', 2); -- PLC01
END;
/
```

e) **AddRegistration in Registration and vehicle table**

```
CREATE OR REPLACE PROCEDURE addRegis (
    p_owner_name IN VARCHAR2,
    p_aadhaar    IN NUMBER,
    p_vehicle_id IN NUMBER
) AS
    v_owner_id       Owner.Owner_ID%TYPE;
    v_registration_id Registration.Registration_ID%TYPE := seq_regis.NEXTVAL;
BEGIN
    -- Fetch Owner ID
    SELECT Owner_ID INTO v_owner_id
    FROM Owner
    WHERE Name = p_owner_name AND Aadhaar = p_aadhaar;

    -- Insert into Registration table
    INSERT INTO Registration (
        Registration_ID,
        Registration_Date,
        Location,
        Vehicle_ID,
```

```
        Owner_ID
    ) VALUES (
        v_registration_id,
        SYSDATE,
        'Default Location', -- You can modify to accept a parameter
        p_vehicle_id,
        v_owner_id
    );

    -- Update Vehicle with registration ID if needed
    UPDATE Vehicle
    SET Registration_ID = v_registration_id
    WHERE Vehicle_ID = p_vehicle_id;

    DBMS_OUTPUT.PUT_LINE('Registration Added with ID: ' || v_registration_id);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Owner not found for given name and Aadhaar.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

f) **UpdateRegistration in Registration, Owner table and vehicle table**

```
CREATE OR REPLACE PROCEDURE UpdateRegis (
    p_regis_id      IN NUMBER,
    p_new_location  IN VARCHAR2,
    p_new_aadhaar   IN NUMBER,
    p_new_name      IN VARCHAR2
)
AS
    v_vehicle_id    NUMBER;
BEGIN
    -- Step 1: Get Vehicle_ID from Registration table
    SELECT Vehicle_ID
    INTO v_vehicle_id
    FROM Registration
    WHERE Registration_ID = p_regis_id;

    -- Step 2: Update Registration table with new location, aadhaar, and name
    UPDATE Registration
    SET Location = p_new_location,
        Aadhaar = p_new_aadhaar,
        Owner_Name = p_new_name  -- Assuming these fields exist in Registration
```

```
    WHERE Registration_ID = p_regis_id;

    -- Step 3: Update Vehicle table with correct Registration_ID
    UPDATE Vehicle
    SET Registration_ID = p_regis_id
    WHERE Vehicle_ID = v_vehicle_id;

    DBMS_OUTPUT.PUT_LINE('Registration and Vehicle records updated successfully.');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for the given Registration ID.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

g) **AddRules in the Offense table**

```
CREATE OR REPLACE PROCEDURE addRules (
    p_offence_type   IN VARCHAR2,
    p_penalty        IN NUMBER
)
AS
    v_offence_id NUMBER;
BEGIN
    -- Get next value from sequence
    SELECT seq_Rules.NEXTVAL INTO v_offence_id FROM dual;

    -- Insert into Offence table
    INSERT INTO Offence (Offence_ID, Offence_Type, Penalty)
    VALUES (v_offence_id, p_offence_type, p_penalty);

    DBMS_OUTPUT.PUT_LINE('Offence rule added with ID: ' || v_offence_id);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error while adding rule: ' || SQLERRM);
END;
```

h) **Update Rules in the Offense table**

```
CREATE OR REPLACE PROCEDURE updateRules (
   p_offence_type   IN VARCHAR2,
   p_new_penalty    IN NUMBER
)
AS
BEGIN
   -- Update the penalty for the given offence
   UPDATE Offence
   SET Penalty = p_new_penalty
   WHERE Offence_Type = p_offence_type;

   IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No matching offence found to update.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Penalty updated for offence: ' || p_offence_type);
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error while updating rule: ' || SQLERRM);
END;
/
```

i) **Add Vehicle_offence in the Vehicle Offence table**

```
CREATE OR REPLACE PROCEDURE addVehicleOffence(
   p_offence_id      IN NUMBER,
   p_registration_id IN NUMBER,
   p_location        IN VARCHAR2,
   p_proof           IN BLOB
)
AS
   v_exists_offence  NUMBER;
   v_exists_regis    NUMBER;
BEGIN
   -- Check if Offence ID exists
   SELECT COUNT(*) INTO v_exists_offence
   FROM Offence
   WHERE Offence_ID = p_offence_id;

   -- Check if Registration ID exists
   SELECT COUNT(*) INTO v_exists_regis
   FROM Registration
   WHERE Registration_ID = p_registration_id;
```

```
    IF v_exists_offence = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Offence ID does not exist.');
    ELSIF v_exists_regis = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Registration ID does not exist.');
    ELSE
        -- Insert into Vehicle_Offence with Status = 1
        INSERT INTO Vehicle_Offence (
            ID,
            Offence_ID,
            Registration_ID,
            Location,
            Status,
            Proof
        ) VALUES (
            seq_vehicle_offence.NEXTVAL,
            p_offence_id,
            p_registration_id,
            p_location,
            1, -- Status is always set to 1
            p_proof
        );

        DBMS_OUTPUT.PUT_LINE('Vehicle Offence inserted successfully.');
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/
```

j) **Update Vehicle_offence in the Vehicle Offence table**

```
CREATE OR REPLACE PROCEDURE deleteVehicleOffenceByRegAndVO(
    p_reg_id IN NUMBER,
    p_vehicle_offence_id IN NUMBER
)
AS
    v_count NUMBER;
BEGIN
    -- Check if the vehicle_offence_id matches the given reg_id
    SELECT COUNT(*) INTO v_count
    FROM Vehicle_Offence
    WHERE Registration_ID = p_reg_id AND ID = p_vehicle_offence_id;
```

```
    IF v_count = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No Vehicle_Offence found for the given Registration ID and
Offence ID.');
    ELSE
        DELETE FROM Vehicle_Offence
        WHERE Registration_ID = p_reg_id AND ID = p_vehicle_offence_id;

        DBMS_OUTPUT.PUT_LINE('Vehicle_Offence ID ' || p_vehicle_offence_id || ' under
Registration_ID ' || p_reg_id || ' deleted successfully.');
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/
```