TRAFFIC MANAGEMENT SYSTEM

MYSQL

1. Introduction

A **Traffic Management System** is designed to manage vehicle information, road details, violations, and payments in an organized way using SQL. Traffic rules are critical for public safety, and maintaining digital records of violations helps in monitoring and reducing accidents.

The project implements a database with multiple tables such as **Vehicles, Roads, Violations, and Payments**, ensuring data consistency through primary and foreign keys. SQL queries like **INSERT, UPDATE, DELETE, SELECT, GROUP BY, and Subqueries** are demonstrated.

Additionally, advanced concepts like **Stored Procedures** and **Triggers** are included to make the system more practical and automated.

2. Objectives of the Project

1. To design a database for managing traffic rules and violations.
2. To store vehicle and owner details securely.
3. To track road information and speed limits.
4. To record violations and generate fine reports.
5. To manage payment history for fines.
6. To demonstrate SQL concepts: **DDL, DML, DCL, Aggregates, Subqueries, Stored Procedures, and Triggers.**

3. System Design

The system consists of **four major entities**:

1. **Vehicles** – stores vehicle details, owner names, and contact numbers.
2. **Roads** – stores road information including speed limits.
3. **Violations** – records types of violations, fine amounts, and dates.
4. **Payments** – maintains payment details for clearing fines.

Relationships:

- Each **Violation** is linked to a **Vehicle** and a **Road**.
- Each **Payment** is linked to a **Violation**.

This ensures **referential integrity** with the use of **foreign keys**.

4. SQL Implementation

The following SQL code implements the **Traffic Management System**:

```
-- SQL Project: Traffic Management System

CREATE DATABASE TrafficDB;
```

```
USE TrafficDB;

-- Create Tables
CREATE TABLE Vehicles (
    vehicle_id INT PRIMARY KEY,
    plate_number VARCHAR(20),
    owner_name VARCHAR(100),
    vehicle_type VARCHAR(50),
    contact_number VARCHAR(15)
);

CREATE TABLE Roads (
    road_id INT PRIMARY KEY,
    road_name VARCHAR(100),
    city VARCHAR(50),
    speed_limit INT
);

CREATE TABLE Violations (
    violation_id INT PRIMARY KEY,
    vehicle_id INT,
    road_id INT,
    violation_type VARCHAR(50),
```

```
    fine_amount DECIMAL(10,2),
    violation_date DATE,
    FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),
    FOREIGN KEY (road_id) REFERENCES Roads(road_id)
);

CREATE TABLE Payments (
    payment_id INT PRIMARY KEY,
    violation_id INT,
    payment_date DATE,
    payment_method VARCHAR(50),
    amount DECIMAL(10,2),
    FOREIGN KEY (violation_id) REFERENCES Violations(violation_id)
);

-- Insert Data
INSERT INTO Vehicles VALUES
(1, 'AP05AB1234', 'Ravi Kumar', 'Car', '9876543210'),
(2, 'TS09CD5678', 'Anil Reddy', 'Bike', '9876500012'),
(3, 'KA03EF9876', 'Suma Rao', 'Truck', '9876511111');

INSERT INTO Roads VALUES
(101, 'MG Road', 'Hyderabad', 50),
```

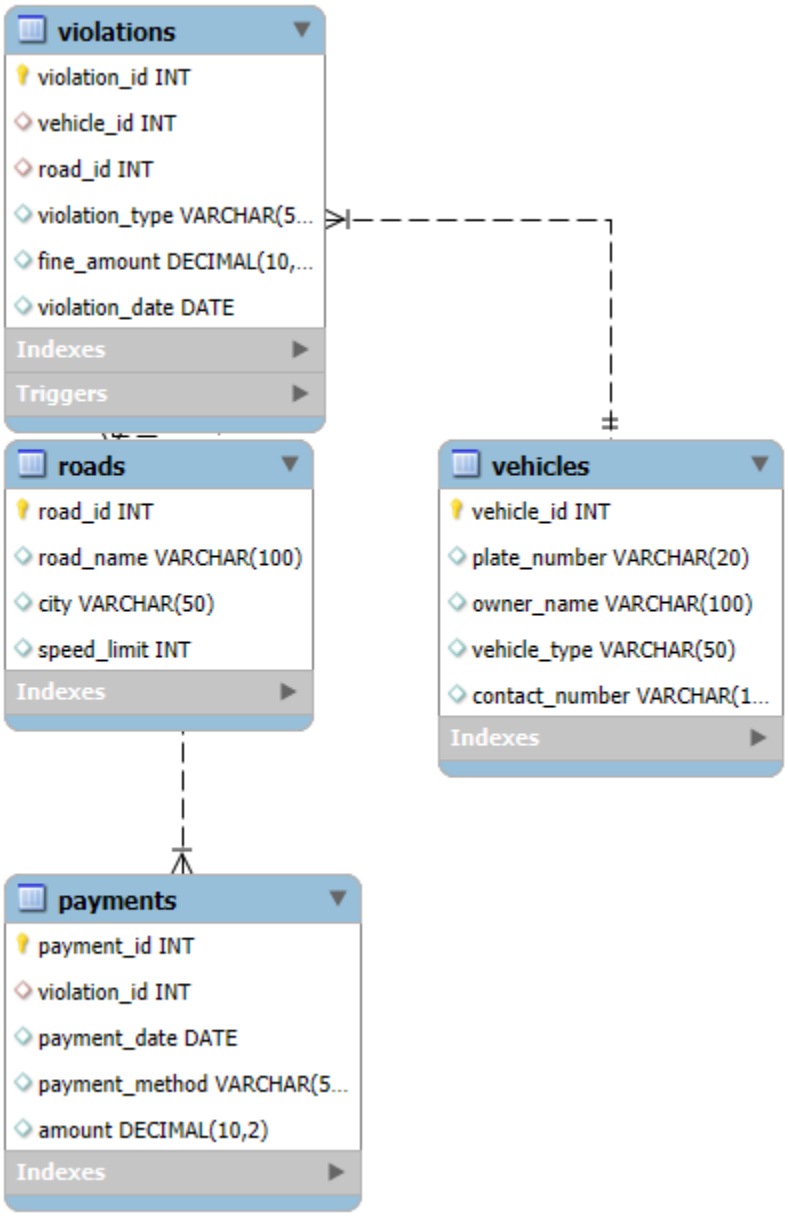
```
(102, 'Ring Road', 'Bangalore', 80);

INSERT INTO Violations VALUES
(201, 1, 101, 'Over Speed', 500.00, '2025-01-10'),
(202, 2, 101, 'Signal Jump', 1000.00, '2025-01-15'),
(203, 3, 102, 'Over Speed', 1500.00, '2025-02-05');

INSERT INTO Payments VALUES
(301, 201, '2025-01-12', 'Credit Card', 500.00),
(302, 202, '2025-01-20', 'UPI', 1000.00);

-- Update Example
UPDATE Vehicles
SET contact_number = '9999999999'
WHERE vehicle_id = 1;

-- Delete Example
DELETE FROM Payments
WHERE payment_id = 302;

-- WHERE Example
SELECT * FROM Violations
WHERE fine_amount > 800;
```

```sql
-- Aggregate Functions
SELECT COUNT(*) AS total_violations, SUM(fine_amount) AS total_fines
FROM Violations;

-- GROUP BY + HAVING
SELECT violation_type, COUNT(*) AS cases, SUM(fine_amount) AS total_fines
FROM Violations
GROUP BY violation_type
HAVING COUNT(*) > 1;

-- LIKE Example
SELECT * FROM Vehicles
WHERE owner_name LIKE 'R%';

-- Subquery Example
SELECT owner_name
FROM Vehicles
WHERE vehicle_id IN (
    SELECT vehicle_id
    FROM Violations
    WHERE fine_amount > 1000
);
```

```
-- Stored Procedure
DELIMITER //
CREATE PROCEDURE GetVehicleFines(IN v_id INT)
BEGIN
    SELECT v.owner_name, SUM(fine_amount) AS total_fine
    FROM Violations vio
    JOIN Vehicles v ON vio.vehicle_id = v.vehicle_id
    WHERE v.vehicle_id = v_id
    GROUP BY v.owner_name;
END //
DELIMITER ;

CALL GetVehicleFines(1);

-- Trigger
DELIMITER //
CREATE TRIGGER AutoPaymentInsert
AFTER INSERT ON Violations
FOR EACH ROW
BEGIN
    INSERT INTO Payments(payment_id, violation_id, payment_date, payment_method,
amount)
```

```
    VALUES (NEW.violation_id + 300, NEW.violation_id, CURDATE(), 'Pending',
NEW.fine_amount);

END //

DELIMITER ;
```

5. ER Diagram



6. Results & Analysis

1. Vehicles and road details are successfully inserted.
2. Violations are tracked along with fine amounts.
3. Payments table maintains transaction history.
4. The **Stored Procedure** helps generate quick fine reports for a vehicle.
5. The **Trigger** automatically inserts a payment record when a new violation occurs.

7. Conclusion

The **Traffic Management System SQL Project** effectively demonstrates the use of SQL in real-life scenarios. By maintaining digital records of vehicles, violations, and payments, authorities can:

- Reduce manual paperwork.
- Ensure transparency in fine collection.
- Improve traffic law enforcement.

This project can be extended further by adding modules like **CCTV integration, real-time alerts, and online payment gateways**.