***Registered Vehicle Database Management***

**Under the Guidance of** Professor Kannan Srikanth  
  
**Team Members: Group 6**

* Tejas Kumar Reddy Acholu - (TXA230028)
* Dheeraj Atmakuri -(DXA230020)
* Abhilash Reddy Kadukuntla - (ARK220002)
* Gurjar Jitender Kumar- (GJK230001)
* Robin Shukla - (RVS230001)



# ***Meeting Logs***

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Meeting date/time | Members present | Items discussed |
| 1 | Date: 17/02/2024  Time: 11m 7s | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * General Topic Discussion * Brainstorming on the topic * Decided to prepare and come for the next meeting |
| 2 | Date: 18/02/2024  Time: 41m 16s | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Discussed the problem statement. * Brainstormed some entities to be included in the project |
| 3 | Date: 19/02/2024  Time: 31m 23s | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Reviewed Contributed Content * Discussed the Target Audience to be included in the charter |
| 4 | Date: 23/02/2024  Time: 46m | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Worked on Project draft and proposals and discussed tables |
| 5 | Date: 29/02/2024  Time: 1.5Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Assigned and Created tables. * Worked on tables, data insertion |
| 6 | Date: 04/03/2024  Time: 2Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Discussed complex queries. * Worked on complex queries, discussed about video format |
| 7 | Date: 10/03/2024  Time: 58m | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Discussed Functions. * Worked on Functions, discussed about Triggers, Sops |
| 8 | Date: 17/03/2024  Time: 2.5Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Worked on creating Triggers |
| 9 | Date: 24/03/2024  Time: 3Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Created SOPs. * Added comments to all the queries |
| 10 | Date: 25/04/2024  Time: 3Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Discussed Project Presentation, Video, and Report |
| 11 | Date: 06/05/2024  Time: 2Hrs | Tejas Kumar Reddy Acholu  Dheeraj Atmakuri  Abhilash Reddy Kadukuntla  Gurjar Jitender Kumar  Robin Shukla | * Final Project Report * Presentation Video Editing and Submission |

# ***Problem Statement:***

In our fast-paced modern world, the vehicle registration process is still bogged down by outdated and manual procedures that are time-consuming and prone to errors and inefficiencies. The reliance on paperwork and physical records creates significant administrative burdens, processing delays, and challenges in maintaining accurate and up-to-date vehicle registration and owner information. This disjointed approach is further hampered by fragmented data storage and a lack of integration between different agencies' systems. This leads to difficulty tracking compliance with regulations such as emissions standards and safety requirements. A modernized, centralized vehicle registration management system is critical to automate processes, enhance data integrity, and ensure seamless compliance with regulations. By transforming the vehicle registration landscape into a more efficient, accurate, and user-friendly environment, law enforcement and regulatory bodies can more effectively monitor and enforce vehicular laws, enhancing public safety, environmental standards, and operational efficiency.

# ***Project Overview***

**Revolutionizing Vehicle Registration for the Digital Age**

**Introduction to Innovation:** Our solution presents the Registered Vehicle Database Management System in response to the obstacles posed by outdated and ineffective vehicle registration methods. This technology-driven platform is designed to revamp and streamline vehicle registration and management. It is a prime example of how innovation can simplify complex bureaucratic procedures, resulting in a more efficient and user-friendly experience.

**Centralization and Ease of Use:** Our system boasts a centralized platform simplifying vehicle-related data management. From initial registration to subsequent renewals, all information is consolidated in one easily accessible location. By streamlining this process, we minimize the potential for errors and make the system user-friendly for both administrators and users.

**Automation and Regulatory Compliance:** Streamlined Processes and Adherence to Regulations: Our system's foundation lies in automation, replacing laborious manual tasks with swift and precise operations. By doing so, the registration process becomes faster and guarantees strict compliance with crucial regulations related to vehicle emissions, safety, and other standards. This, in turn, bolsters public safety and safeguards the environment.

**Data Accuracy and Accessibility:** Ensuring Accurate Data and Facilitating Information Retrieval: In a world where data reigns supreme, our system ensures the dependability and precision of vehicle information. This is vital for making well-informed decisions and shaping policies. Additionally, the system offers convenient access to vehicle data for both authorities and the public, fostering transparency and enhancing supervision.

**Expanding Beyond Registration:** Our system goes beyond simple registrations and renewals. It strives to become a crucial tool for law enforcement, environmental agencies, and policy researchers by providing comprehensive data on vehicle trends and regulatory compliance. This valuable insight will assist in the development of future policies and strategies.

**Vision for the Future:** Our project represents a groundbreaking utilization of innovative technology to transform the vehicle registration process, enhancing its efficiency, precision, and transparency. This marks a crucial stride towards a more intelligent, secure, and eco-friendly vehicular environment, in keeping with our mission to drive public service advancements through innovation and technology.

# ***Primary Objective***

To develop and deploy a robust database management system tailored for managing registered vehicles that enhances regulatory compliance improves operational efficiency and facilitates seamless service delivery.

1. Data Integration: Integrate diverse data sources, including vehicle registration information, ownership records, and inspection history, into a unified database system for comprehensive vehicle management.

2. Regulatory Compliance: Ensure adherence to legal requirements and standards governing vehicle registration and documentation, mitigating risks, and ensuring regulatory compliance.

3. Operational Efficiency: Streamline registration renewal, license plate issuance, and documentation handling to optimize operational workflows and reduce processing times.

4. Data Security: Implement stringent security measures to safeguard sensitive vehicle information, ensuring confidentiality, integrity, and data availability to authorized personnel only.

5. User Accessibility: Develop an intuitive user interface that facilitates easy navigation and seamless interaction with the system, enhancing user satisfaction and productivity.

6. Scalability and Integration: Design a scalable system architecture capable of accommodating future growth and seamlessly integrating with external systems for enhanced functionality and interoperability.

7. Support and Training: Provide users with comprehensive training and ongoing technical support to maximize system utilization and ensure efficient operation over time.

# ***Key Features***

* **Centralized Data Repository:** Implementing a unified platform that integrates and manages all pertinent vehicle-related information, comprising owner particulars, registrations, and renewals, in a centralized location is imperative to streamline the management of such data.
* **Automated Process Management:** Automating routine tasks, such as processing applications, data entry, and renewals, can significantly reduce manual effort, time consumption, and the potential for human error.
* **Regulatory Compliance Tracking:** Our system has advanced features to closely monitor and enforce compliance with vehicle emissions, safety standards, and other regulatory requirements. This helps facilitate better enforcement of vehicular laws and ensures that all vehicles adhere to the highest safety and environmental protection standards.
* **Data Integrity and Security:** We maintain exceptionally high data accuracy and reliability standards. Our robust security measures ensure that sensitive information is protected, and that privacy is always maintained.
* **Enhanced Accessibility:** User-friendly access to vehicle information for various stakeholders, including vehicle owners, regulatory bodies, and law enforcement, through a secure, intuitive interface.
* **Advanced Analytics and Reporting:** Get insights into vehicle registration trends, compliance rates, and other critical metrics with powerful analytics tools. These tools support data-driven decision-making and policy formulation.
* **Multi-Stakeholder Support:** Features designed to cater to the needs of a broad range of users, from government agencies and law enforcement to environmental groups and researchers, enhancing cooperation and information sharing.
* **Future-Ready Infrastructure:** Infrastructure that is scalable and adaptable to accommodate future technological advances and regulatory changes, ensuring it remains relevant and practical.
* **Community Engagement and Education:** Tools and resources to help the public and stakeholders understand vehicle regulations, safety standards, and environmental impacts, promoting responsible compliance.
* **Incident and History Tracking:** Recording and accessing historical data on vehicle incidents, such as accidents and traffic violations, can aid investigations and policy evaluations.

# ***Benefits***

* **Increased Efficiency:** Automates and simplifies the registration process, reducing administrative workload and improving service delivery.
* **Improved Data Integrity:** Ensures accurate and up-to-date vehicle and owner information, reducing errors and fraud.
* **Enhanced Compliance:** Facilitates better vehicular laws and regulations enforcement through effective tracking and reporting.
* **Accessible Information:** Makes vehicle information readily available for stakeholders, enhancing transparency and public safety.
* **Data-Driven Decision Making:** Enables informed policy and decision-making through comprehensive analytics and reporting.

# ***Target Audience***

**Government Regulatory Bodies**

* **Department of Motor Vehicles (DMV):** Primary users overseeing vehicle registrations, renewals, and compliance with regulations.
* **Environmental Agencies:** Interested in monitoring vehicle emissions and enforcing environmental standards.
* **Transportation Authorities:** Focus on the broader aspects of transportation policy, safety, and infrastructure planning.

**Law Enforcement Agencies**

* **Traffic Police:** Require access to vehicle and owner information for enforcing traffic laws, investigating incidents, and ensuring public safety.
* **Accident Investigation Units:** Utilize the system to retrieve vehicle histories and owner details during accident investigations.

**Vehicle Owners**

* **Individual Owners:** Benefit from an easy-to-use platform for registering vehicles, renewing registrations, and staying informed about compliance requirements.
* **Commercial Fleet Operators:** Businesses that manage multiple vehicles and need efficient ways to handle registrations, compliance, and record-keeping.

**Insurance Companies**

* **Underwriters and Claims Adjusters:** Use the system to access vehicle information for policy underwriting, risk assessment, and processing claims.

**Researchers and Policy Makers**

* **Transportation and Environmental Researchers:** Analyze data from the system for studies on traffic patterns, environmental impacts, and the effectiveness of vehicle regulations.
* **Policy Makers:** Leverage insights from the system to inform transportation policies, safety regulations, and environmental standards.

**General Public and Advocacy Groups**

* **Safety Advocates:** Interested in how vehicle compliance contributes to road safety.
* **Environmental Advocates:** Focus on the environmental impact of vehicles and support for green policies.

**Technology and Data Analysts**

* **IT Professionals:** Involved in the system's development, maintenance, and security.
* **Data Analysts:** Work with the data generated by the system for reporting, insights, and improvements.

# ***Executive Summary***

The management of vehicle registrations is taking a significant step forward with implementing the Registered Vehicle Database Management project. By utilizing technology to streamline processes and improve data accuracy, compliance, and reporting, this project will serve as a cornerstone for modernizing vehicular administration. It is expected to simplify administrative tasks and enhance public safety, environmental compliance, and operational efficiency. This project is pivotal in achieving a more organized, transparent, and practical vehicle registration system.

# ***Assumptions and Notes***

* **Vehicle Uniqueness:** Each `VehicleID` corresponds to one unique vehicle, identifiable by its VIN, ensuring data integrity.
* **Ownership Dynamics:** A vehicle can be associated with multiple owners over its lifecycle, but only one at any given time, managed through `Vehicle Ownership` with specified `PurchaseDate` and `SaleDate.`
* **Registration Details:** Vehicle registrations are unique to each vehicle-owner pair and can vary by state, captured comprehensively to manage diverse regulatory requirements.
* **Comprehensive Tracking:** The system meticulously tracks interactions of vehicles with regulatory (via Traffic Violations and parking Tickets) and maintenance systems (Maintenance Records), ensuring a rich historical data set.
* **Insurance Management:** Vehicles are linked to insurance policies that can stay with the vehicle as it changes owners. Insurance claims are detailed and include descriptions of incidents to facilitate management and resolution.
* **Service and Payments:** External services are tracked via Service Providers, and financial transactions are logged in Payment History, capturing essential financial interactions related to vehicle management.
* **Fuel Usage:** Fuel Logs offer detailed records of fuel usage by type and cost, essential for cost management and environmental compliance considerations.

# ***Entities and Their Attributes:***

* **Vehicles**: Identified by a unique VehicleID. Attributes include VIN, LicensePlate, Make, Model Year, Color, and Vehicle Type.
* **Owners**: Identified by a unique OwnerID. Attributes include FirstName, LastName, Address, ZipCode, State, Phone Number, Email, and PurchaseDate.
* **Vehicle Ownership**: Identified by VehicleId. It references VehicleID from Vehicles and OwnerID from Owners. Attributes include PurchaseDate and SaleDate.
* **Vehicle Registration**: Identified by a combination of VehicleID and OwnerID. It references VehicleID from Vehicles and OwnerID from Owners. Attributes include PurchaseDate, Sale Date, Registration Type, Registration Fee, Registration Status, State of Registration Number, and Renewal Date.
* **Registration Renewals**: Identified by Renewal ID. It references Registration ID from Vehicle Registrations. The only attribute is the Renewal Date.
* **Insurance Companies**: Identified by Insurance Company ID. Attributes include CompanyName, Contact Details, Coverage Type, Policy Holder Name, Start Date, and End Date.
* **Insurance Policies**: Identified by Insurance ID. It references Vehicle ID from Vehicles and Insurance Company ID from Insurance Companies.
* **Parking Tickets**: Identified by Ticket Number. It references Vehicle ID from Vehicles. Attributes include Issue Date and Fine Amount.
* **Maintenance Records**: Identified by RecordID. It references Vehicle ID from Vehicles. Attributes include Service Date and Service Type.
* **Traffic Violations**: Identified by ViolationID. It references Vehicle ID from Vehicles. Attributes include Date, Penalty, Vehicle Type, and GasType.
* **Vehicle Specifications**: Identified by SpecID. It references Vehicle ID from Vehicles. Attributes include Vehicle Type, GasType, Engine Size, and Horsepower.
* **Service Providers**: Identified by ProviderID. Attributes include Provider Name and Service Type.
* **Payment History**: Identified by PaymentID. It references OwnerID from Owners. Attributes include Amount, Payment Date, and Mode of Payment.

# ***Information about Tables***

**1. Vehicles Table:** Stores detailed vehicle information, including Vehicle ID, VIN, license plate, make, model, year, color, and type.

**2. Owners Table:** Information about vehicle owners, such as Owner ID, first and last names, address, zip code, state, phone number, email, and the dates of vehicle purchase and sale.

**3. Vehicle Ownership Table:** Associates vehicles with their owners and tracks ownership dates, handling the relationship between vehicle IDs and owner IDs.

**4. Vehicle Registration Table:** Contains vehicle registration details, including registration type, fees, status, and the state of registration linked to vehicle and owner IDs.

**5. Registration Renewals Table:** Tracks renewal dates for vehicle registrations, identifying each renewal by a unique ID.

**6. Insurance Company Table:** Lists insurance companies with details like company name and contact information.

**7. Insurance Policies Table:** Manages vehicle insurance policies, detailing coverage types, policyholder names, start and end dates, and links to specific vehicles and insurance companies.

**8. Parking Tickets Table:** Records instances of parking violations, noting the vehicle involved, issue date, and acceptable amount.

**9. Maintenance Records Table:** Keeps a log of vehicle maintenance activities, specifying the type of service and the date it was performed.

**10. Traffic Violations Table:** Documents traffic violations by vehicle, including the date, penalty amount, vehicle type, and gas type.

**11. Vehicle Specifications Table:** Provides specific technical details about each vehicle, such as gas type, engine size, and horsepower.

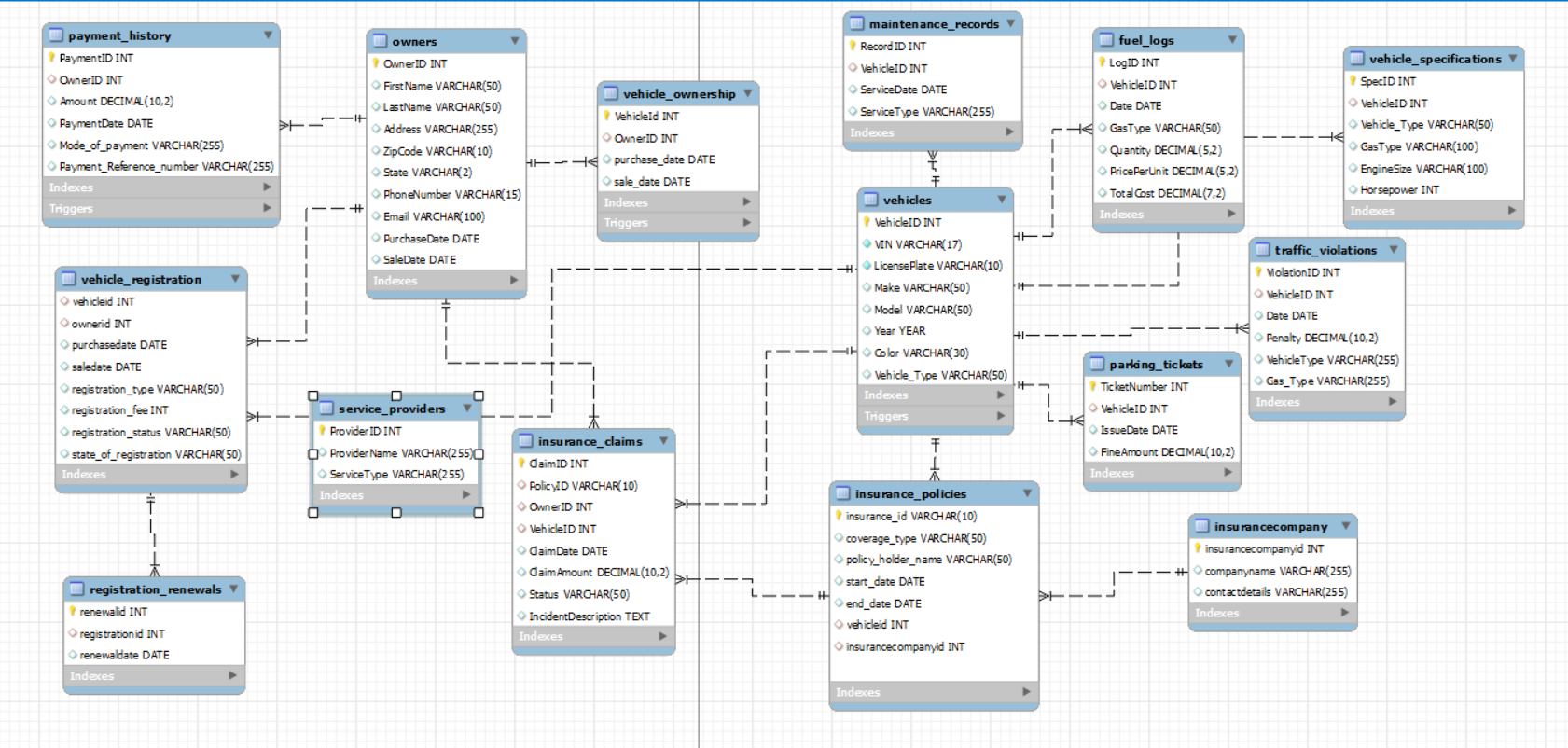
**12. Service Providers Table:** Lists companies or individuals providing vehicle-related services, such as repairs, towing, and insurance.

**13. Payment History Table:** Tracks payments made by vehicle owners, including payment amount, date, mode, and reference number.

**14. Insurance Claims Table:** Details of claims made against vehicle insurance policies, including the nature of the incident, claim amount, status, and related vehicle and owner IDs.

**15. Fuel Logs Table:** Records vehicle fuel purchases, noting the type of gas, quantity, price per unit, and total cost.

# ***Entity Relationship (ER) Diagram***



# ***Scenarios Covered in the Database***

* + Created 15 tables, each containing 50 records.
  + Developed 10 complex queries (Used Left Join, Right Join, Inner Join, Union)
  + Implemented 5 functions.
  + Created 5 stored procedures.
  + Set up 6 triggers for insert, update, and delete operations.

# ***Design of the Database***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Primary Key (PK)** | **Foreign Key (FK)** | **Non-Key Attributes** | **# of Rows in Table** |
| **Vehicles** | VehicleID |  | VIN (unique), LicensePlate (unique), Make, Model, Year, Color, Vehicle\_Type | 50 |
| **Owners** | OwnerID |  | FirstName, LastName, Address, ZipCode, State, PhoneNumber, Email, PurchaseDate, SaleDate | 50 |
| **Vehicle Ownership** | VehicleId | VehicleID (references Vehicles),  OwnerID (references Owners) | PurchaseDate, SaleDate | 50 |
| **Vehicle Registration** | (VehicleID, OwnerID) | VehicleID (references Vehicles),  OwnerID (references Owners) | PurchaseDate, SaleDate, RegistrationType, RegistrationFee, RegistrationStatus, StateOfRegistration | 50 |
| **Registration Renewals** | RenewalID | RegistrationID (references Vehicle\_Registration) | RenewalDate | 50 |
| **Insurance Companies** | InsuranceCompanyID |  | CompanyName, ContactDetails | 50 |
| **Insurance Policies** | InsuranceID | VehicleID (references Vehicles), InsuranceCompanyID (references Insurance Companies) | CoverageType, PolicyHolderName, StartDate, EndDate | 505 |
| **Parking Tickets** | TicketNumber | VehicleID (references Vehicles) | IssueDate, FineAmount | 50 |
| **Maintenance Records** | RecordID | VehicleID (references Vehicles) | ServiceDate, ServiceType | 50 |
| **Traffic Violations** | ViolationID | VehicleID (references Vehicles) | Date, Penalty, VehicleType, Gas\_Type | 50 |
| **Vehicle Specifications** | SpecID | VehicleID (references Vehicles) | Vehicle\_Type, GasType, EngineSize, Horsepower | 50 |
| **Service Providers** | ProviderID |  | ProviderName, ServiceType | 50 |
| **Payment History** | PaymentID | OwnerID (references Owners) | Amount, PaymentDate, Mode\_of\_Payment, Payment\_Reference\_Number | 50 |
| **Insurance Claims** | ClaimID | PolicyID (references Insurance\_Policies), OwnerID (references Owners), VehicleID (references Vehicles) | ClaimDate, ClaimAmount, Status, IncidentDescription | 50 |
| **Fuel Logs** | LogID | VehicleID (references Vehicles) | Date, GasType, Quantity, PricePerUnit, TotalCost (calculated field) | 50 |

# ***Functions***

Functions are reusable blocks of code that perform specific tasks. They encapsulate logic, enhancing code modularity, readability, and reusability. Each function has a name, input parameters, a body containing code to execute, and a return value for output. They enable developers to break down complex tasks into smaller, manageable parts.

**1. InsuranceStatus(p\_VehicleID INT):** The `InsuranceStatus’ function determines the insurance status of a vehicle by its ID. It checks if there is an active insurance policy for the vehicle in the `insurance\_policies` table with an end date that has not yet expired. If such a policy exists, it returns to `'Insured'`; otherwise, it returns to `'Not Insured'`. This function is helpful for quickly verifying a vehicle's insurance coverage.

* This function checks whether a vehicle with the given VehicleID is insured or not.
* It queries the insurance\_policies table to find any insurance policies associated with the provided VehicleID and checks if the policy's end date is greater than or equal to the current date.
* If an active insurance policy is found, it returns 'Insured,' otherwise 'Not Insured.'

**Result Set:** SELECT InsuranceStatus(8).



**2. TotalParkingFines(p\_VehicleID INT):** The function `TotalParkingFines` calculates the total amount of parking fines for a specified vehicle by its ID. It accepts one parameter, `p\_VehicleID`, which identifies the vehicle. Inside the function, it queries the `Parking\_Tickets` table to sum up all the fines associated with the given vehicle ID. If the vehicle has no parking tickets, the `IFNULL` function ensures that the sum defaults to zero rather than returning a null value. The total calculated fines are stored in the `totalFines` variable, which is then returned by the function. This function is useful for quickly aggregating the total parking fines incurred by a specific vehicle and helping in financial assessments or records management.

* This function calculates the total fines associated with a given VehicleID from the Parking\_Tickets table.
* It summarizes the FineAmount column for all parking tickets associated with the provided VehicleID.
* The function returns the total fines as a decimal value.

**Result Set:** SELECT TotalParkingFines(7).



**3. LatestServiceDetails(p\_VehicleID INT):** The function `LatestServiceDetails` retrieves the most recent service record details for a given vehicle identified by its ID (`p\_VehicleID`). This function uses a query to pull the latest `ServiceDate` and `ServiceType` from the `Maintenance\_Records` table for the specified vehicle. It then concatenates these details into a single string in the format `'Latest Service Date: [ServiceDate], Latest Service Type: [ServiceType]'`. The result is stored in the `latestServiceDetails` variable.

* This function retrieves the latest service date and service type for a given VehicleID from the Maintenance\_Records table.
* It fetches the service date and service type of the most recent maintenance record associated with the provided VehicleID.
* The function returns a string containing the latest service date and type.

**Result Set:** SELECT latestServiceDetails(20);



**4. CalculateTotalPenalty(vehicle\_id INT):** The function “CalculateTotalPenalty” computes the total monetary penalties incurred by a specific vehicle identified by vehicle\_id. It sums up all the penalty amounts from the Traffic\_Violations table where the VehicleID matches the provided vehicle\_id. The sum of these penalties is stored in the total\_penalty variable.

* This function calculates the total penalty amount for a given VehicleID from the Traffic\_Violations table.
* It summarizes the Penalty column for all traffic violations associated with the provided VehicleID.
* The function returns the total penalty amount as a decimal value.

**Result Set:** SELECT CalculateTotalPenalty(4) AS TotalPenaltyForVehicle;

A close up of a sign

Description automatically generated

**5. GetVehicleRegistrationDetailsByVIN(vin\_input VARCHAR(17)):** The SQL function `GetVehicleRegistrationDetailsByVIN` retrieves registration details for a vehicle based on its Vehicle Identification Number (VIN). This function accepts a single parameter, `vin\_input`, representing the vehicle's VIN. It returns a formatted string containing various pieces of registration data such as the Vehicle ID, License Plate, Registration Type, Registration Status, and the State of Registration.

The function performs an inner join between the `Vehicles` table and the `vehicle\_registration` table using the `VehicleID` field to ensure that it only considers entries where the vehicle's registration information exists. It then constructs a single string from these details using the `CONCAT` function, formatting it with labels for clarity.

This function is particularly useful for applications that quickly fetch and display registration information based on a vehicle's VIN, such as administrative panels, user dashboards, or compliance reports in automotive industries.

* This function retrieves vehicle registration details based on the VIN (Vehicle Identification Number).
* It queries the Vehicles and Vehicle\_registration tables to find the vehicle registration details associated with the provided VIN.
* The function returns a string containing various registration details such as VehicleID, LicensePlate, RegistrationType, etc.

**Result Set:** SELECT GetVehicleRegistrationDetailsByVIN("5J6RM4H74EL000005");  


**6. GetVehicleSpecifications(vin VARCHAR(17)):** The SQL function `GetVehicleSpecificationsByVehicleID` is designed to fetch and format critical specifications of a vehicle identified by its vehicle ID. When called with a specific `vehicle\_id`, it compiles details such as the vehicle type, gas type, engine size, and horsepower from the `Vehicle\_Specifications` table. The result is formatted as a single string, which is convenient for displaying in user interfaces or reports.

If no specifications are found for the given vehicle ID (i.e., the query returns null), the function provides a default message indicating no specifications were found. This ensures that the function always returns a meaningful response, even in cases where the database lacks the requested data.

* This function retrieves vehicle specifications based on the VIN (Vehicle Identification Number).
* It fetches details such as Make, Model, Year, Color, vehicle type, etc., along with additional specifications like gas type, engine size, and Horsepower from the Vehicles and Vehicle specifications tables.
* The function returns a string containing the fetched vehicle specifications.

**Result Set:** SELECT GetVehicleSpecificationsByVehicleID(01);



# ***Stored Procedures***

Stored procedures (SOPs) in database management are precompiled routines stored in a database that perform specific tasks. They enhance data management, security, and efficiency by encapsulating SQL statements and business logic. Each stored procedure has a name, input parameters, and a body containing the code to execute.

**1. ClearViolation(IN p\_ViolationID INT):**

* The ClearViolation stored procedure is designed to clear a specific traffic violation from the system by setting its penalty to zero.
* This procedure accepts a single input parameter, p\_ViolationID, an integer representing the unique identifier of the traffic violation to be cleared.
* If the violation is successfully cleared, it returns a message indicating "Violation cleared successfully."
* If no valid violation is found or associated with an owner/vehicle, it returns a message stating, "Violation not found or not associated with an owner."

**Result Set:** CALL ClearViolation(50);

**A close up of a sign

Description automatically generated**

**2. GetVehicleInsuranceHistory(IN p\_VehicleID INT):**

* The GetVehicleInsuranceHistory stored procedure's primary purpose is to retrieve a specific vehicle's insurance history from the database.
* The procedure expects a single input parameter named p\_VehicleID, an integer representing the unique identifier of the vehicle for which insurance history is to be retrieved.
* The procedure uses a cursor named insurance\_cursor to fetch insurance policy details associated with the specified vehicle from the insurance\_policies table.
* Inside the loop, the fetched insurance policy details and associated insurance claims are retrieved and stored in local variables.

**Result Set:** CALL GetVehicleInsuranceHistory(2);

****

**3. ListVehiclesByTypeWithPagination(IN \_Vehicle\_Type VARCHAR(50)) :**

* The stored procedure is designed to retrieve a paginated list of vehicles based on a specified vehicle type.
* The procedure executes a SELECT query on the Vehicles table, filtering records by the specified vehicle type. It orders the results by VehicleID and applies a LIMIT clause to paginate them based on the provided offset and limit values.
* The procedure can handle large datasets effectively by fetching only the necessary subset of records, making it suitable for applications with growing data volumes.

**Result Set:** CALL ListVehiclesByTypeWithPagination('SUV', 0, 10);

A screenshot of a computer

Description automatically generated

**4. AddNewVehicleWithValidation():**

* The stored procedure is designed to add a new vehicle to the database while ensuring that the VIN (Vehicle Identification Number) and License Plate are unique.
* Before inserting a new record into the Vehicles table, the procedure checks if a vehicle with the provided VIN or License Plate already exists in the database.
* If a vehicle with the provided VIN or License Plate already exists (vehicle\_exists is not equal to 0), the procedure raises an error using the SIGNAL statement. This ensures data integrity by preventing duplicate entries for VIN or License Plate.
* By enforcing uniqueness constraints on VIN and License Plate, the procedure helps maintain data integrity within the Vehicles table, ensuring accurate and reliable information.

**Result Set:** CALL AddNewVehicleWithValidation('JH4TB2H46CC011200', '1ZVAP8AM2', 'Honda', 'Civic', 2020, 'Black', 'Sedan');

A screenshot of a computer

Description automatically generated

**5. CountVehiclesByColor():**

* The stored procedure aims to retrieve a count of vehicles grouped by their colors from the Vehicles table.
* The procedure utilizes the GROUP BY clause to aggregate the data based on the distinct values of the Color column.
* The procedure can be easily modified or extended to include additional criteria or filters for counting vehicles, such as by adding WHERE clauses to restrict the data based on specific conditions.
* The procedure provides valuable insights into the distribution of vehicles based on their colors, which can be helpful for various analytical purposes such as inventory management, market analysis, and trend identification.

**Result Set:** CALL CountVehiclesByColor('GREEN');

****

***TRIGGERS***

Triggers are automatic code blocks executed when specific data change events (INSERT, UPDATE, DELETE) occur on a table, serving critical functions like logging, validation, or additional DML actions.

Trigger1:

Description: Automates the assignment of PaymentDate to the current date upon inserting a new payment record.

1. Insert Trigger: `before\_insert\_payment`

* This trigger is executed before a new record is inserted into the `Payment\_History` table.
* The purpose of the trigger is to automatically set the `PaymentDate` column of the new record to the current date using the `CURDATE()` function.
* This ensures that the payment date is accurately recorded when a new payment is added to the system.

Result set: INSERT INTO Owners (FirstName, LastName, Address, ZipCode, State, PhoneNumber, Email, PurchaseDate, SaleDate)

VALUES ('Rian', 'ge', '123 Main St', '12345', 'NY', '555-1294', 'brad@example.com', '2020-01-01', NULL);

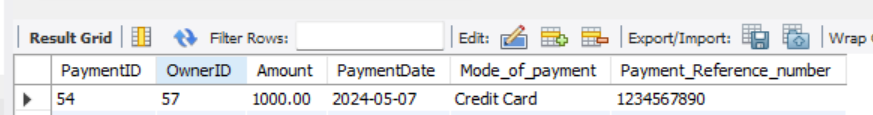
INSERT INTO Payment\_History (OwnerID, Amount, Mode\_of\_payment, Payment\_Reference\_number)

VALUES (57, 1000.00, 'Credit Card', '1234567890');

-- Select data from Payment\_History table

SELECT \* FROM Payment\_History;

SELECT \* FROM Owners;

Output:

Trigger2:

Description: This trigger prevents updates to specific columns in the "Vehicles" table if there are associated parking tickets. It checks for any parking tickets linked to the vehicle being updated and restricts updates to the "Make," "Model," "Year," and "Vehicle\_Type" columns.

2.Update Trigger: `prevent\_vehicle\_update`

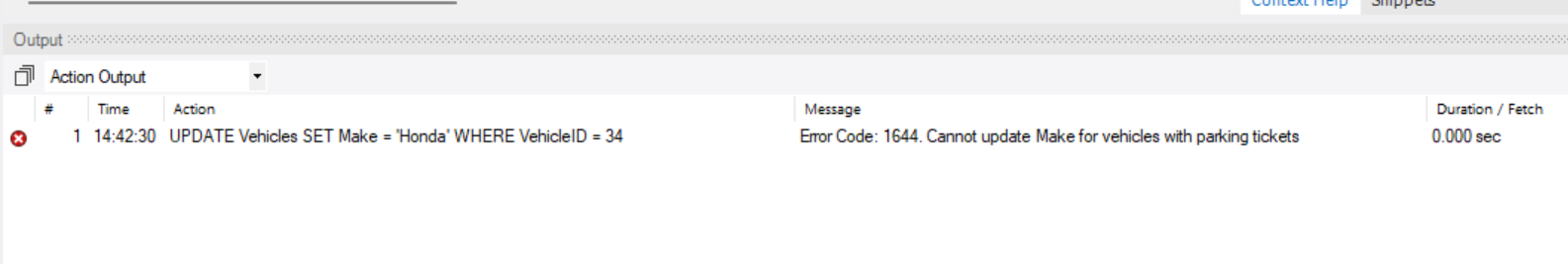
* This trigger is executed before a record in the `Vehicles` table is updated.
* The purpose of the trigger is to prevent updates to the `Make`, `Model`, `Year`, and `Vehicle\_Type` columns if there are associated parking tickets for the vehicle being updated.
* The trigger first checks if there are any parking tickets associated with the vehicle by counting the number of records in the `Parking\_Tickets` table with the same `VehicleID`. If there are associated parking tickets, the trigger checks if any of the protected columns are being updated. If so, it throws an error message and prevents the update.
* This ensures that the core vehicle information cannot be changed if there are outstanding parking tickets associated with the vehicle.

Result set: UPDATE Vehicles

SET Make = 'Honda'

WHERE VehicleID = 34;

Sample output:



Trigger3

Description: The trigger "after\_vehicle\_update" fires after an update operation on the "Vehicles" table. For each updated row, it inserts a new record into the "Maintenance\_Records" table with the vehicle's ID, the current date, and a service type indicating that the vehicle was updated.

3. Trigger: `after\_vehicle\_update`

* This trigger is executed after a record in the `Vehicles` table is updated.
* The trigger inserts a new record into the `Maintenance\_Records` table with the `ServiceDate` set to the current date and the `ServiceType` indicating that the vehicle was updated.
* When a vehicle record is updated in the `Vehicles` table, this trigger inserts a new record into the `Maintenance\_Records` table. The new record includes the `VehicleID` of the updated vehicle, the current date as the `ServiceDate`, and the `ServiceType` set to "Vehicle Updated". This helps maintain a history of vehicle updates and can be useful for tracking maintenance and service activities.

Result set: UPDATE Vehicles

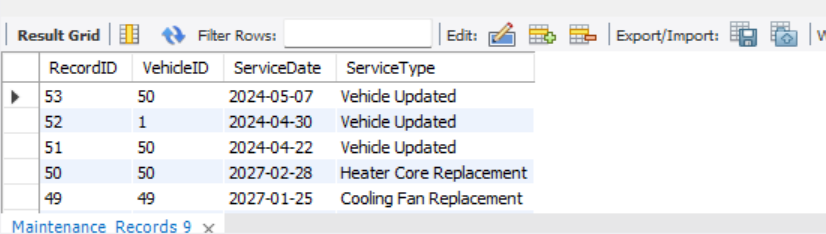
SET Color = 'Blue'

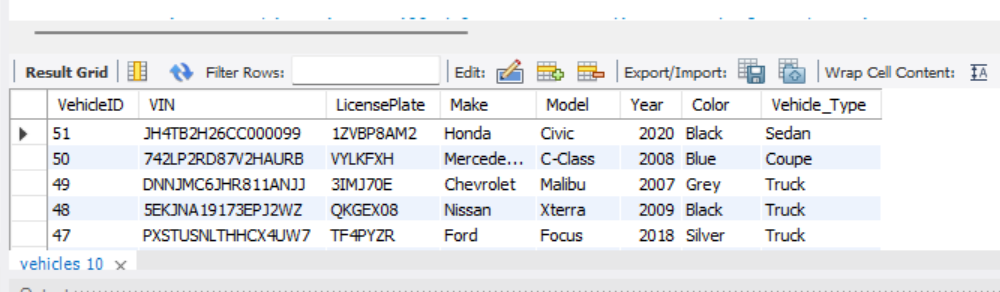
WHERE VehicleID = 50;

SELECT \* FROM Maintenance\_Records;

select \* from vehicles;

Sample result:





4. Delete Trigger: `before\_delete\_vehicle`

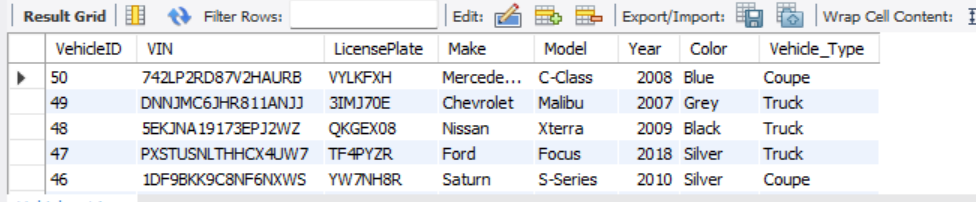
Description: The trigger "before\_delete\_vehicle" is created to execute before the deletion of a row in the "Vehicles" table. It deletes corresponding maintenance records associated with the vehicle being deleted from the "Maintenance\_Records" table.

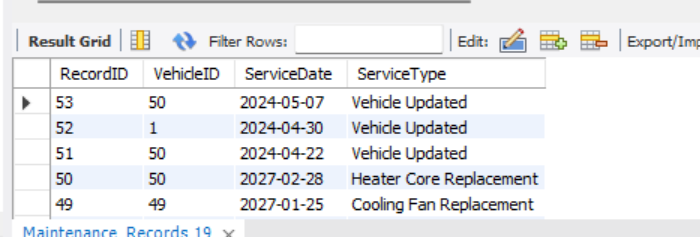
* This trigger is executed before a record is deleted from the `Vehicles` table.
* The purpose of the trigger is to delete the corresponding records from the `Maintenance\_Records` table to maintain referential integrity.
* When a vehicle record is deleted from the `Vehicles` table, this trigger first deletes all the corresponding records from the `Maintenance\_Records` table where the `VehicleID` matches the `VehicleID` of the deleted vehicle.
* This ensures that the maintenance records are also removed, maintaining the consistency of the data and the referential integrity between the `Vehicles` and `Maintenance\_Records` tables.

Result set: DELETE FROM Vehicles WHERE VehicleID = 51;

SELECT \* FROM Vehicles;

SELECT \* FROM Maintenance\_Records;

Sample result:



***Complex Queries***

**Query 1: Total Fuel Costs by Vehicle Type**

Utility: This query aggregates total fuel expenditure for each vehicle type, helping in budgeting and financial forecasting specific to vehicle categories.  
**Result Set:**

A screenshot of a graph

Description automatically generated

**Query 2: Total Fuel Consumption by Vehicle Type**

Utility: Assists in environmental and operational performance assessment by showing fuel usage per vehicle type, which can guide eco-friendly initiatives and fuel efficiency improvements.

**Result Set:**

A screenshot of a graph

Description automatically generated

**Query 3: Most Common Vehicle Color**

Utility: Identifies the most popular vehicle color, which helps marketing teams understand trends and customer preferences in vehicle aesthetics.

**Result Set:**



**Query 4: Total Traffic Violations by Vehicle Type**

Utility: Provides insights into the frequency of traffic violations across different vehicle types, aiding in risk assessment and safety improvement strategies.

**Result Set:**

**A screenshot of a computer

Description automatically generated**

**Query 5: Vehicle Types Lacking Insurance**

Utility: Identifies vehicle types without insurance policies, which is critical for compliance checks and risk management.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 6: Average Horsepower by Vehicle Type**

Utility: Offers a statistical insight into the average horsepower per vehicle type, useful for performance benchmarking and marketing.

**Result Set:**

A screenshot of a data

Description automatically generated

**Query 7: Total Payments Received by Month**

Utility: This query helps in financial tracking and trend analysis by summarizing monthly income from various payments.

**Result Set:**

A screenshot of a table

Description automatically generated

**Query 8: Most Common Vehicle Services**

Utility: Pinpoints the most frequently required vehicle services, guiding inventory management and promotional focuses in service offerings.

**Result Set:**

A screenshot of a service list

Description automatically generated

**Query 9: Total Insurance Claims by Company**

Utility: Assists insurance companies in financial planning by showing the total claims processed, helping to gauge fiscal impacts.

**Result Set:**

A screenshot of a document

Description automatically generated

**Query 10: Monthly Fuel Expenditure Overview**

Utility: Tracks fuel costs over time, providing a detailed monthly expense report that can help with budgeting and operational planning.

**Result Set:**

A screenshot of a number

Description automatically generated

**Query 11: Vehicles with Expired Insurance Policies**

Utility: Essential for ensuring legal compliance and continuous coverage by identifying vehicles whose insurance policies have expired.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 12: Average Fine Amount by Vehicle Type for Traffic Violations**

Utility: Useful for assessing the fiscal impact of traffic violations by vehicle type, helping manage risk and policy adjustments.

**Result Set:**

A screenshot of a computer screen

Description automatically generated

**Query 13: Service Providers by Service Type**

Utility: Aggregates all service providers according to the types of services offered, facilitating resource allocation and partnership decisions.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 14: Total Insurance Claims by Vehicle Type**

Utility: Provides a breakdown of insurance claims amounts filed per vehicle type, aiding insurers in risk assessment and premium setting.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 15: Comprehensive Owner and Vehicle Insurance Details**

Utility: Ensures up-to-date tracking of vehicles with their owners and current insurance status, vital for operational integrity and customer service.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 16: Complete Owner to Vehicle Mapping**

Utility: Displays a comprehensive list of all owners and any vehicles they own, ensuring no details are overlooked in records.

**Result Set:**

A screenshot of a computer

Description automatically generated

**Query 17: Full Overview of Vehicle and Owner Associations**

Utility: Combines comprehensive details of vehicles and owners, providing a full spectrum view useful for audits, customer service, and compliance checks.

**Result Set:**

A screenshot of a computer

Description automatically generated

***CONCLUSION***

Our project is like giving the old, slow vehicle registration system a turbo boost. We are replacing the clunky paperwork with a sleek digital system that makes registering and managing vehicles a breeze. It is all about making things easier, faster, and smoother for everyone involved, like upgrading from a horse-drawn carriage to a high-speed sports car.