Vehicle Recall Management System: Data Analysis and Reporting

Introduction

The "Vehicle Maintenance and Service Tracking System" introduces such need of automotive repair shops for easier management of the service history, maintenance schedule, recall notifications, which may boost their operational efficiency on the one hand, while enhancing vehicle owners to access it with ease, allowing owners of the vehicles to maintain good vehicle health and handle the issues regarding recalls effectively. It is extremely useful in maintaining recall notices, which have become a rising concern in the automotive industry due to safety regulations. It allows manufacturers, repair shops, and owners to stay interconnected, minimizing the risks associated with delayed or missed recalls.

The application uses a freemium model wherein the basic version, comprising features such as tracking of service records and receiving recall notifications, is free for an individual user. Advanced features such as predictive maintenance alerts, cloud storage integration for vehicle documents, and real-time recall updates are part of a subscription-based plan for repair shops and large-scale vehicle fleets. This tiered pricing ensures accessibility for casual users while providing value to commercial entities that need more robust solutions.

My personal interest in this system is because I am an avid fan of motor sports and automotive technology. Having worked on data analytics for different industries, this was an opportunity to merge my interest in cars with my skills in data management and application development. The idea of using technology to improve vehicle safety and convenience resonates with me, making this project both a professional challenge and a personal ambition.

User Personas

1. Individual Vehicle Owners

The individual car owners are a very important user group of the application. They use the system to stay informed about recall events, track their vehicles' recall statuses, and manage maintenance effectively.

- Queries like "Retrieve Recall Events by Manufacturer" enable vehicle owners to view detailed recall event data such as the severity of the event, affected vehicles, and corrective actions.
- Owners can also review their responses to the query "Retrieve User Responses for a Specific Recall Event" to see if they have responded to the most critical issues on time.
- The system will enable owners to stay updated, schedule repairs, and retrieve contact details for follow-up with ease, thus enhancing safety and convenience.

2. Automotive Repair Shops

Repair shops are also benefited as the application enables them to access recall data and vehicle information in an organized manner to carry out repair services in a smooth manner.

- The query "Listing All Vehicles and Their Recall Statuses" helps repair shops identify vehicles requiring attention, ensuring they can prioritize repairs.
- Using "Retrieve Full Details About a Vehicle, Its Owner, and Its Manufacturer", shops can personalize their services and improve communication with vehicle owners.
- By providing real-time recall information and repair tracking, the application enhances operational efficiency and customer satisfaction for repair shops

3. Vehicle Manufacturers

It is used by manufacturers to monitor recall campaigns, track recall severity, and analyze user responses.

- For example, the query "Find All Recall Events with the Highest Severity and Corresponding User Responses" will help the manufacturer identify urgent safety concerns and measure the success of their recall campaigns.
- "Categorize Recall Events Based on Severity and Display Response Actions" can be used by manufacturers to see how their users respond to recalls, helping plan future campaigns and increase response rates.
- The application puts ways for manufacturers to remain compliant and safe while building trust with their customers.

4. Fleet Managers

Fleet managers use the application to manage and track the recall and maintenance status of vehicles in their fleets.

- Queries like "Retrieve All Vehicles Involved in Safety Recalls and Their Status" ensure that the fleet managers can identify the affected vehicles and take corrective actions on time.
- Access to such in-depth data through "Retrieve Full Details About a Vehicle, Its Owner, and Its Manufacturer" helps the fleet manager to manage his fleet and stay in compliance with safety requirements.

It reduces downtime and improves the overall safety of the fleet operation, thereby reducing risks and costs.

Business Rules or Logic:

Several business rules are in place and enforced by the application to make sure it works as expected and data integrity is maintained:

- One and only one manufacturer for each Recall Event, but multiple recall events for one manufacturer.
- A Recall Event can involve multiple vehicles, and vehicles can be involved in multiple recall events - many-to-many relationship via Vehicle Recall table.
- An owner can have multiple vehicles, and each vehicle is from one and only one manufacturer.
- A User Response comes from a single vehicle owner and can be linked to multiple recall events.
- Recall events can be categorized based on the severity of the recall; each severity dictates the priority of corrective actions.

Table Design and Analysis

The Vehicle Recall Tracking System is designed to efficiently manage and automate the tracking, repair, and resolution of vehicle recall events. It integrates key entities like manufacturers, vehicles, and user responses to ensure smooth operations and data integrity. Below is a detailed breakdown of the entities, relationships, and business rules.

1. Manufacturer

Primary Key (PK): manufacturer_id (Unique identifier for each manufacturer)
Attributes:

- name: Name of the manufacturer
- vehicle_type: Type of vehicle, e.g., Sedan, SUV, Truck
- country: Country of origin
- contact_info: Manufacturer's contact information

Business Logic:

- **One-to-Many Relationship with Recall_Event**: Each manufacturer can initiate multiple recall events, but each recall event belongs to only one manufacturer. This relationship is represented as one-to-many between the Manufacturer table and the Recall_Event table.
- **One-to-Many Relationship with Vehicle**: Each manufacturer produces multiple vehicles, establishing a one-to-many relationship between Manufacturer and Vehicle tables.

2. Recall_Event

Primary Key (PK): nhtsa_id (Unique identifier for each recall event)
Foreign Key (FK): manufacturer_id (References Manufacturer table)
Attributes:

- report_date: Date the recall was reported
- recall_type: Type of recall (Safety, Emissions, etc.)
- description: Detailed description of the recall issue
- corrective_action: Steps suggested to fix the issue
- severity: Severity level of the recall (Low, Moderate, High)
- affected_vehicles: Total number of vehicles affected
-

Business Logic:

- **Many-to-One Relationship with Manufacturer**: Each recall event is associated with a single manufacturer, establishing a many-to-one relationship between the Recall_Event and Manufacturer tables.
- **Many-to-Many Relationship with Vehicle**: A recall event can affect multiple vehicles, and a vehicle can be part of multiple recall events. This relationship is modeled through the Vehicle_Recall junction table, enabling a many-to-many relationship.

3. Vehicle

Primary Key (PK): vehicle_id (Unique identifier for each vehicle)
Foreign Keys (FKs):

- manufacturer_id (References Manufacturer table)
- owner_id (References Vehicle_Owner table)
  Attributes:
- vin_number: Vehicle Identification Number
- model_year: Year of manufacture
- model: Vehicle model name
- status: Status of the vehicle (Active, Retired, Sold)

Business Logic:

- **Many-to-One Relationship with Manufacturer**: Each vehicle is produced by one manufacturer, establishing a many-to-one relationship between the Vehicle and Manufacturer tables.
- **One-to-Many Relationship with Vehicle_Owner**: A vehicle is owned by one vehicle owner, and an owner can have multiple vehicles.
- **Many-to-Many Relationship with Recall_Event**: A vehicle can be involved in multiple recall events, and a recall event can affect multiple vehicles. This relationship is represented through the Vehicle_Recall table.

4. Vehicle_Owner

Primary Key (PK): owner_id (Unique identifier for each vehicle owner)
Attributes:

- name: Owner's full name
- location: Owner's geographical location
- owner_contact_info: Contact details of the owner

Business Logic:

- **One-to-Many Relationship with Vehicle**: A vehicle owner can own multiple vehicles, establishing a one-to-many relationship between the Vehicle_Owner and Vehicle tables.
- **One-to-Many Relationship with User_Response**: A vehicle owner can respond to multiple recall events, establishing a one-to-many relationship between Vehicle_Owner and User_Response tables.

5. Vehicle_Recall

Primary Key (PK): vehicle_recall_id (Unique identifier for each vehicle recall entry)
Foreign Keys (FKs):

- vehicle_id (References Vehicle table)
- recall_event_id (References Recall_Event table)
  Attributes:
- recall_status: Status of the recall for the vehicle (Pending, Completed, Ignored)
- repair_date: Date when the recall repair was completed

Business Logic:

- **Many-to-One Relationship with Vehicle**: Each vehicle recall record is linked to a specific vehicle.
- **Many-to-One Relationship with Recall_Event**: Each vehicle recalls record links to a specific recall event.
- **Many-to-Many Relationship between Vehicle and Recall_Event**: This junction table tracks the specific recall status and repairs performed for each affected vehicle.

6. User_Response

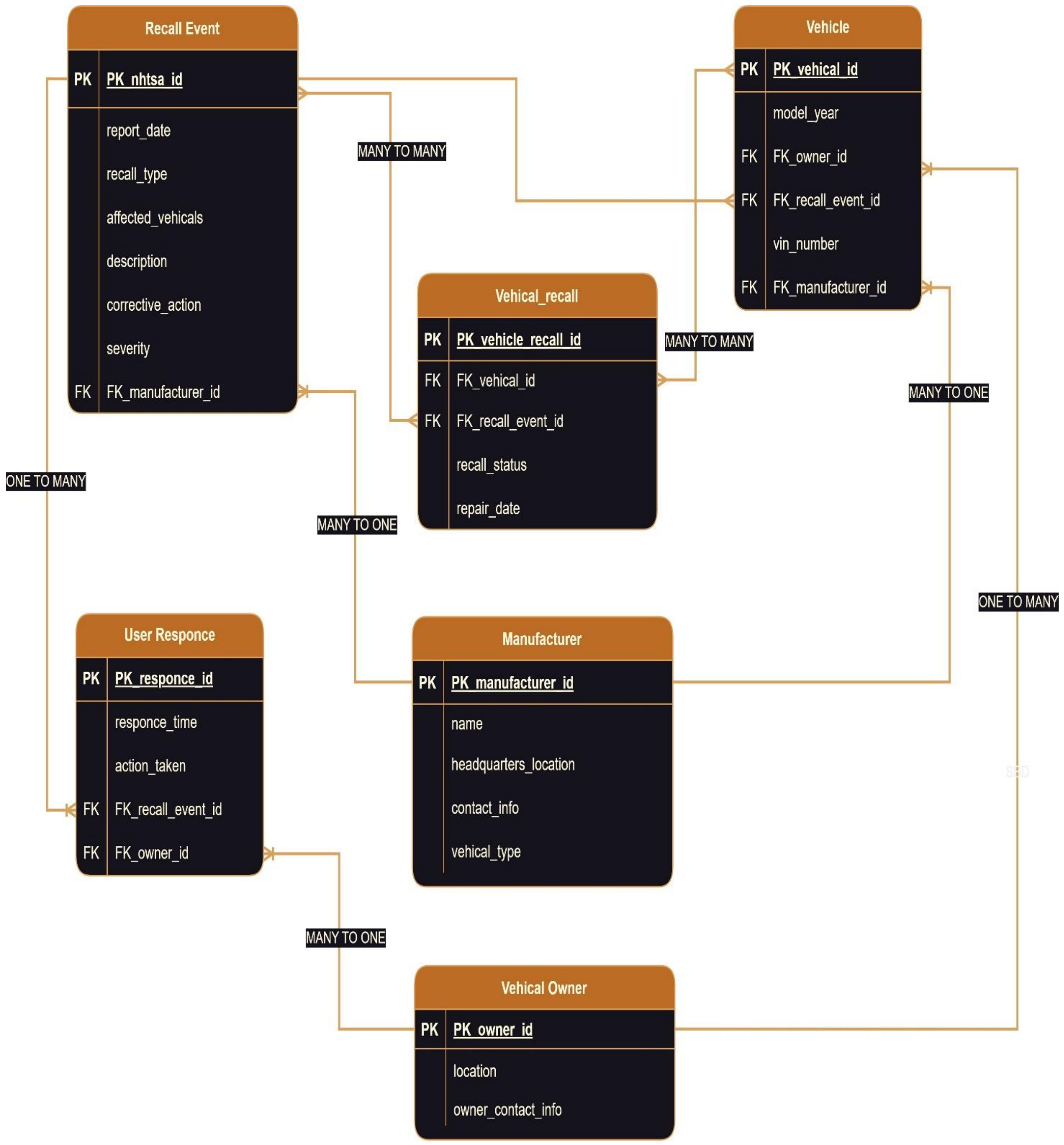
Primary Key (PK): response_id (Unique identifier for each user response)
Foreign Keys (FKs):

- owner_id (References Vehicle_Owner table)
- recall_event_id (References Recall_Event table)
  Attributes:
- action_taken: Response by the user (Repaired, Ignored, Scheduled)
- response_time: Timestamp of the user's response

Business Logic:

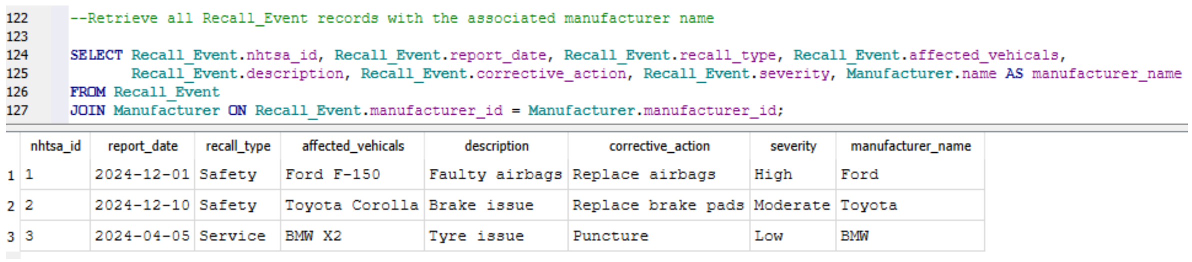
- **Many-to-One Relationship with Vehicle_Owner**: Each user response comes from a specific vehicle owner.
- **Many-to-One Relationship with Recall_Event**: Each response is linked to a specific recall event.
- **Tracking User Actions**: This table tracks the user's response to a recall event, helping measure the success of the recall campaign and the timely actions taken by vehicle owners.

ER DIAGRAM:


Recall Event
PK PK_nhtsa_id
report_date
recall_type
affected_vehicals
description
corrective_action
severity
FK FK_manufacturer_id
Vehicle
PK PK_vehical_id
model_year
FK FK_owner_id
FK FK_recall_event_id
vin_number
FK FK_manufacturer_id
MANY TO MANY
Vehical_recall
PK PK_vehicle_recall_id
FK FK_vehical_id
FK FK_recall_event_id
recall_status
repair_date
MANY TO MANY
MANY TO ONE
ONE TO MANY
MANY TO ONE
ONE TO MANY
User Responce
PK PK_responce_id
responce_time
action_taken
FK FK_recall_event_id
FK FK_owner_id
Manufacturer
PK PK_manufacturer_id
name
headquarters_location
contact_info
vehical_type
MANY TO ONE
Vehical Owner
PK PK_owner_id
location
owner_contact_info

Database Implementation

1. Manufacturer Details with Recall Events



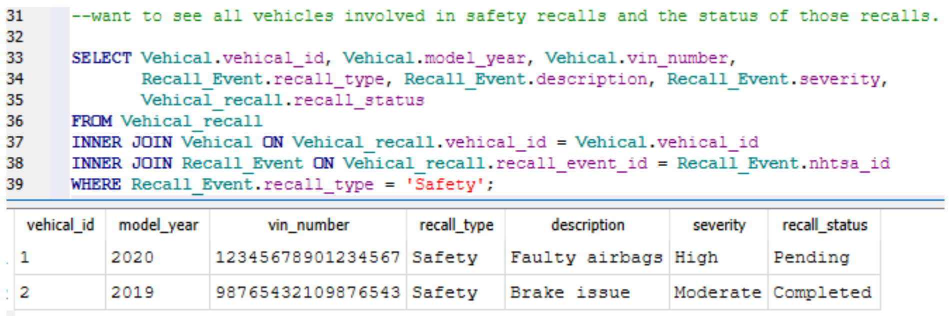
```sql
--Retrieve all Recall_Event records with the associated manufacturer name

SELECT Recall_Event.nhtsa_id, Recall_Event.report_date, Recall_Event.recall_type, Recall_Event.affected_vehicals,
       Recall_Event.description, Recall_Event.corrective_action, Recall_Event.severity, Manufacturer.name AS manufacturer_name
FROM Recall_Event
JOIN Manufacturer ON Recall_Event.manufacturer_id = Manufacturer.manufacturer_id;
```

	nhtsa_id	report_date	recall_type	affected_vehicals	description	corrective_action	severity	manufacturer_name
1	1	2024-12-01	Safety	Ford F-150	Faulty airbags	Replace airbags	High	Ford
2	2	2024-12-10	Safety	Toyota Corolla	Brake issue	Replace brake pads	Moderate	Toyota
3	3	2024-04-05	Service	BMW X2	Tyre issue	Puncture	Low	BMW

Explanation: This query retrieves detailed information about the manufacturer, the vehicles they produce, and the recall events associated with them. It joins the Manufacturer table with the Recall_Event table using manufacturer_id to display recall details along with manufacturer names and vehicle types.

Use Case: This query is useful for tracking which manufacturers are involved in recalling events and what corrective actions they are taking.

2. Recall Events and Vehicle Information



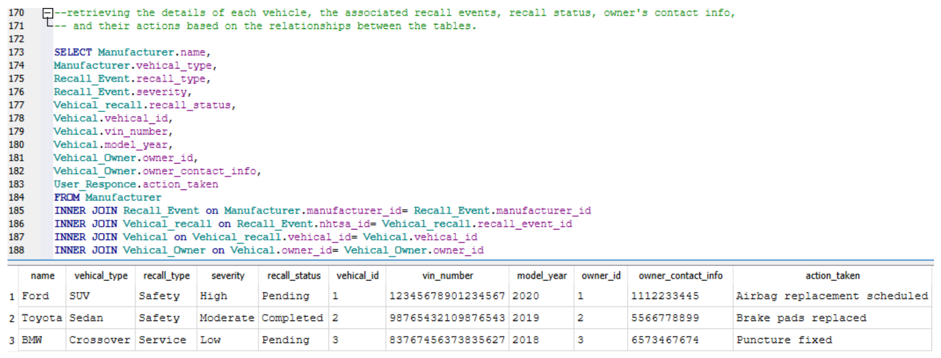
```sql
--want to see all vehicles involved in safety recalls and the status of those recalls.

SELECT Vehical.vehical_id, Vehical.model_year, Vehical.vin_number,
       Recall_Event.recall_type, Recall_Event.description, Recall_Event.severity,
       Vehical_recall.recall_status
FROM Vehical_recall
INNER JOIN Vehical ON Vehical_recall.vehical_id = Vehical.vehical_id
INNER JOIN Recall_Event ON Vehical_recall.recall_event_id = Recall_Event.nhtsa_id
WHERE Recall_Event.recall_type = 'Safety';
```

vehical_id	model_year	vin_number	recall_type	description	severity	recall_status
1	2020	12345678901234567	Safety	Faulty airbags	High	Pending
2	2019	98765432109876543	Safety	Brake issue	Moderate	Completed

Explanation: This query retrieves vehicle details such as vehicle ID, model year, and VIN number, alongside recall event details like the type of recall, description, corrective actions, and severity. It uses INNER JOIN to link the Vehical_recall, Vehical, and Recall_Event tables.

Use Case: This query helps identify which specific vehicles are associated with which recall events, making it easier to track the affected vehicles and their corresponding recall details.

3. Recall Status and Owner Information



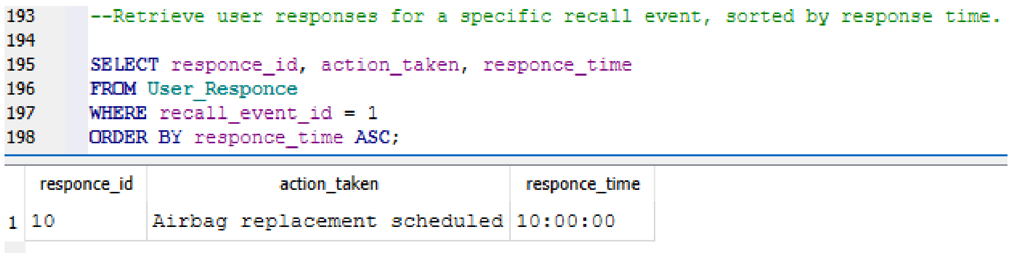
```sql
--retrieving the details of each vehicle, the associated recall events, recall status, owner's contact info,
-- and their actions based on the relationships between the tables.

SELECT Manufacturer.name,
Manufacturer.vehical_type,
Recall_Event.recall_type,
Recall_Event.severity,
Vehical_recall.recall_status,
Vehical.vehical_id,
Vehical.vin_number,
Vehical.model_year,
Vehical_Owner.owner_id,
Vehical_Owner.owner_contact_info,
User_Responce.action_taken
FROM Manufacturer
INNER JOIN Recall_Event on Manufacturer.manufacturer_id= Recall_Event.manufacturer_id
INNER JOIN Vehical_recall on Recall_Event.nhtsa_id= Vehical_recall.recall_event_id
INNER JOIN Vehical on Vehical_recall.vehical_id= Vehical.vehical_id
INNER JOIN Vehical_Owner on Vehical.owner_id= Vehical_Owner.owner_id
```

	name	vehical_type	recall_type	severity	recall_status	vehical_id	vin_number	model_year	owner_id	owner_contact_info	action_taken
1	Ford	SUV	Safety	High	Pending	1	12345678901234567	2020	1	1112233445	Airbag replacement scheduled
2	Toyota	Sedan	Safety	Moderate	Completed	2	98765432109876543	2019	2	5566778899	Brake pads replaced
3	BMW	Crossover	Service	Low	Pending	3	83767456373835627	2018	3	6573467674	Puncture fixed

Explanation: This query joins the Vehical_Recall table with the Vehical_Owner table to retrieve information about the recall status of each vehicle and contact information for the owner of the affected vehicle.

Use Case: This query is vital for tracking the status of recalls for individual vehicles and accessing owner contact information for follow-up actions.

4. Retrieving User Responses for a Specific Recall Event



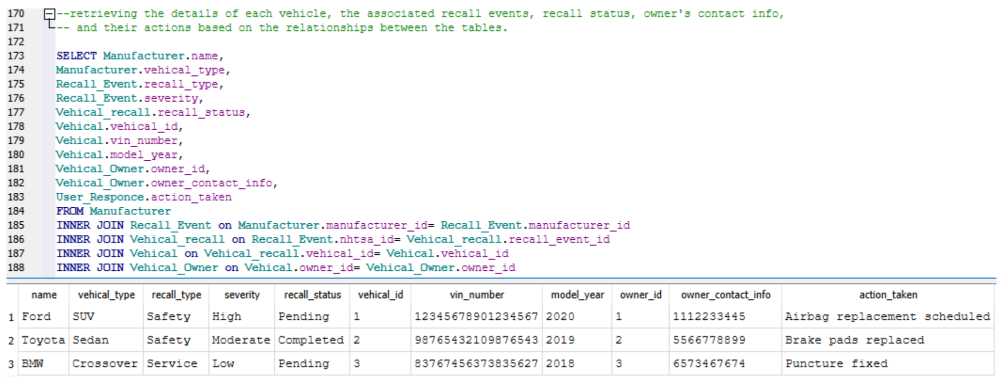
```sql
--Retrieve user responses for a specific recall event, sorted by response time.

SELECT responce_id, action_taken, responce_time
FROM User_Responce
WHERE recall_event_id = 1
ORDER BY responce_time ASC;
```

	responce_id	action_taken	responce_time
1	10	Airbag replacement scheduled	10:00:00

This query fetches user responses for a specific recall event, sorted by response time. It allows administrators to track the timeliness of responses and evaluate how quickly users are addressing recall issues.

Use Case: This can be useful for customer service teams to monitor and improve response times for critical recall events.

5. Recall Events with Manufacturer and User Response



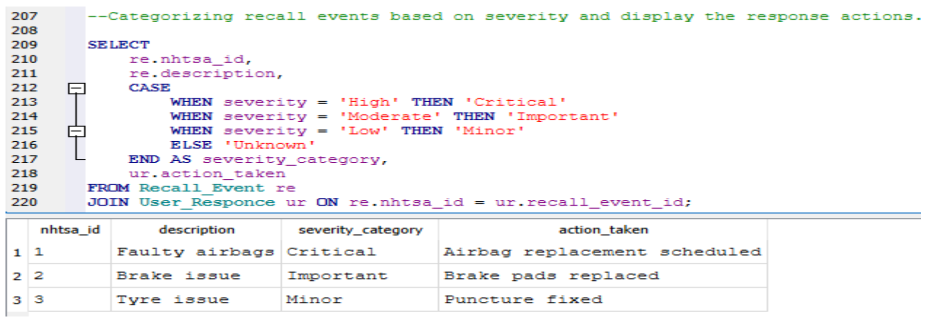
```sql
--retrieving the details of each vehicle, the associated recall events, recall status, owner's contact info,
-- and their actions based on the relationships between the tables.

SELECT Manufacturer.name,
Manufacturer.vehical_type,
Recall_Event.recall_type,
Recall_Event.severity,
Vehical_recall.recall_status,
Vehical.vehical_id,
Vehical.vin_number,
Vehical.model_year,
Vehical_Owner.owner_id,
Vehical_Owner.owner_contact_info,
User_Responce.action_taken
FROM Manufacturer
INNER JOIN Recall_Event on Manufacturer.manufacturer_id= Recall_Event.manufacturer_id
INNER JOIN Vehical_recall on Recall_Event.nhtsa_id= Vehical_recall.recall_event_id
INNER JOIN Vehical on Vehical_recall.vehical_id= Vehical.vehical_id
INNER JOIN Vehical_Owner on Vehical.owner_id= Vehical_Owner.owner_id
```

	name	vehical_type	recall_type	severity	recall_status	vehical_id	vin_number	model_year	owner_id	owner_contact_info	action_taken
1	Ford	SUV	Safety	High	Pending	1	12345678901234567	2020	1	1112233445	Airbag replacement scheduled
2	Toyota	Sedan	Safety	Moderate	Completed	2	98765432109876543	2019	2	5566778899	Brake pads replaced
3	BMW	Crossover	Service	Low	Pending	3	83767456373835627	2018	3	6573467674	Puncture fixed

Explanation: This query joins several tables—Manufacturer, Recall_Event, Vehical_recall, Vehical, and User_Responce—to retrieve recall event details along with manufacturer names, vehicle VIN numbers, and the owner's response to the recall.

Use Case: This is an advanced query for assessing manufacturer involvement in recall events, vehicle details, and tracking how owners are responding to recall notifications.

6. Categorizing Recall Events by Severity and User Response Actions



```
--Categorizing recall events based on severity and display the response actions.

SELECT
    re.nhtsa_id,
    re.description,
    CASE
        WHEN severity = 'High' THEN 'Critical'
        WHEN severity = 'Moderate' THEN 'Important'
        WHEN severity = 'Low' THEN 'Minor'
        ELSE 'Unknown'
    END AS severity_category,
    ur.action_taken
FROM Recall_Event re
JOIN User_Responce ur ON re.nhtsa_id = ur.recall_event_id;
```

	nhtsa_id	description	severity_category	action_taken
1	1	Faulty airbags	Critical	Airbag replacement scheduled
2	2	Brake issue	Important	Brake pads replaced
3	3	Tyre issue	Minor	Puncture fixed

Explanation: This query categorizes recall events based on severity and shows the actions taken by users. It uses a CASE statement to map severity levels like 'High' to 'Critical', 'Moderate' to 'Important', and 'Low' to 'Minor'. The query then joins the Recall_Event table with the User_Responce table to display the action taken by users in response to each recall.

Use Case: This can help manufacturers or regulators track user responses to various recall severities, ensuring critical issues are addressed promptly and appropriately. It assists in prioritizing actions based on the recall's severity.

Insights into Business Logic and Application:

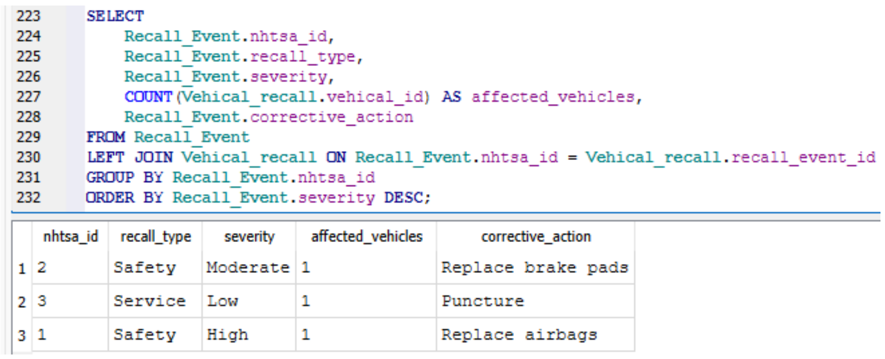
- **Data Integration**: Your SQL queries effectively integrate data across multiple tables, such as vehicle details, recall events, and user responses, offering a comprehensive view of the recall process.
- **Effective Recall Tracking**: By joining the Recall_Event, Vehical_recall, and User_Responce tables, you're able to monitor and analyze the progress of each recall event and track owner actions related to recalls. This can help assess whether recall actions are timely and effective.
- **Customer Communication**: By retrieving owner contact information along with recall statuses and user responses, the system can ensure timely and targeted follow-up communication, improving the chances of getting the owners to act on the recall notice.
- **Response Analysis**: Tracking how owners respond to recalls is critical for assessing the effectiveness of recall campaigns. If many owners do not respond, it may indicate issues with the recall notification process.

Analytics, Reports, and Metrics:

- ### Recall Event Summary Report

Goal: This report provides a summary of all recall events, categorizing them by their severity and corrective actions. It helps stakeholders quickly identify the most critical recalls and prioritize responses accordingly.

SQL Query:



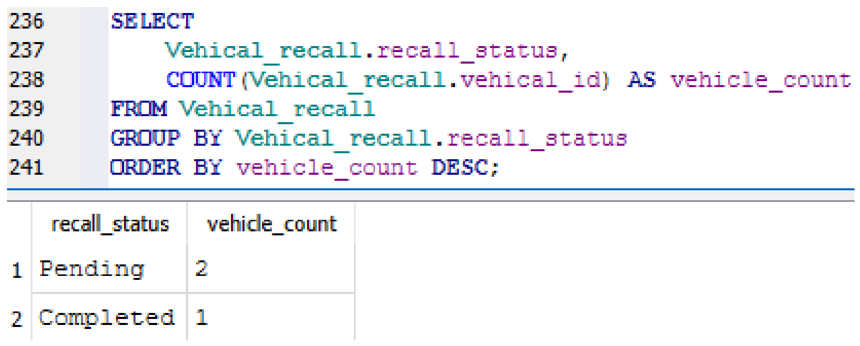
```sql
SELECT
    Recall_Event.nhtsa_id,
    Recall_Event.recall_type,
    Recall_Event.severity,
    COUNT(Vehical_recall.vehical_id) AS affected_vehicles,
    Recall_Event.corrective_action
FROM Recall_Event
LEFT JOIN Vehical_recall ON Recall_Event.nhtsa_id = Vehical_recall.recall_event_id
GROUP BY Recall_Event.nhtsa_id
ORDER BY Recall_Event.severity DESC;
```

	nhtsa_id	recall_type	severity	affected_vehicles	corrective_action
1	2	Safety	Moderate	1	Replace brake pads
2	3	Service	Low	1	Puncture
3	1	Safety	High	1	Replace airbags

- ### Recall Status Tracking for Vehicles:

Goal: This report helps manufacturers and repair shops track the status of recall repairs for vehicles, identifying whether the repairs have been completed, are pending, or have been ignored.

SQL Query:



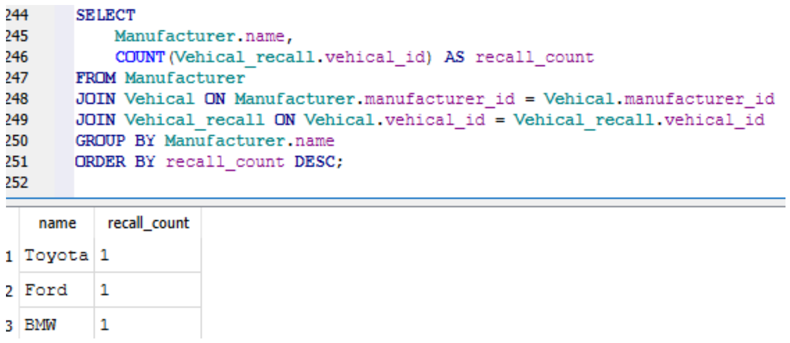
```sql
SELECT
    Vehical_recall.recall_status,
    COUNT(Vehical_recall.vehical_id) AS vehicle_count
FROM Vehical_recall
GROUP BY Vehical_recall.recall_status
ORDER BY vehicle_count DESC;
```

	recall_status	vehicle_count
1	Pending	2
2	Completed	1

- ### Recall Compliance by Manufacturer

Goal: This report shows which manufacturers have the most outstanding recalls, helping prioritize follow-up actions and focus on manufacturers that may need additional attention to meet compliance requirements.

SQL Query:



```sql
SELECT
    Manufacturer.name,
    COUNT(Vehical_recall.vehical_id) AS recall_count
FROM Manufacturer
JOIN Vehical ON Manufacturer.manufacturer_id = Vehical.manufacturer_id
JOIN Vehical_recall ON Vehical.vehical_id = Vehical_recall.vehical_id
GROUP BY Manufacturer.name
ORDER BY recall_count DESC;
```

	name	recall_count
1	Toyota	1
2	Ford	1
3	BMW	1

Key Analytics Goals for Stakeholders:

- Manufacturers: Monitor the number of affected vehicles, severity of recalls, and corrective actions. Ensure conformance to safety standards.
- Repair Shops: Follow-up on recall status, prioritize and manage the workflow to complete the repairs in time.
- System Administrators: Evaluate the efficiency of the recall communications and monitor response rates for regulatory compliance in a timely manner.
- Regulatory Bodies: Get reports on the status of vehicle recalls across manufacturers, ensuring safety and compliance standards are met.

Security and Privacy Concerns for Vehicle Recall Database:

- Personal Identifiable Information (PII)

One concern is that the storage of PII, such as owner contact information, will increase identity theft and violations of privacy.

Encrypt PII, utilize RBAC, and anonymize data when possible.

- Vehicle Identification Numbers (VIN)

There is a concern that VINs linked to personal data can result in privacy risks.

Use tokenization to protect VINs; restrict access by roles.

- Recall Event Data

Concern: Data on high-risk vehicle defects could be exposed to compromise privacy or safety.

Action: Aggregate data and limit access to sensitive recall information.

- User Responses

Concern: Response data could expose personal preferences or actions.

Action: Anonymize and aggregate user responses to prevent tracing.

- Regulatory Compliance

Concern: Data may fall under GDPR, CCPA, or HIPAA and hence risk non-compliance.

Action: Comply with all the privacy laws in terms of data residency, right to access, and deletion.

- Access Control and Authentication

Unauthorized access to sensitive data: Implement MFA, SSO, and segregation of duties.

- Data Integrity

Data manipulation could compromise the integrity of the system or applications: Use blockchain for audit trails; ensure strong data validation.

By addressing these concerns, the database will better protect sensitive information and help maintain compliance with regulations.

References:
Resources like the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) outline critical privacy regulations for protecting sensitive data. For cybersecurity best practices, the NIST Cybersecurity Framework and the OWASP Top Ten provide guidelines to mitigate risks and secure systems effectively.

Architecture:

The Vehicle Recall and Management System will be deployed on cloud-native architecture in AWS or Google Cloud for scalability, reliability, and security.

1. **Cloud Infrastructure:**
    - Serverless compute (AWS Lambda / Google Cloud Functions) for event-driven tasks like recall notifications.
    - EC2/Compute Engine Instances for heavy-duty processing

2. Database & Data Storage:
    - Amazon Aurora / Google Cloud Spanner for storing relational data.
    - AWS S3 / Google Cloud Storage.
    - Amazon Redshift / Google BigQuery for data warehousing and analytics.

3. Microservices Architecture:
    - API Gateway for routing and load balancing.
    - Message Queues (AWS SQS / Google Pub/Sub) for decoupled service communication.

4. Data Analytics & Reporting:
    - AWS Kinesis / Google Dataflow for real-time analytics.
    - BI tools like QuickSight / Data Studio for reporting dashboards.

5. Security & Compliance:
    - Data encryption (AES-256, SSL/TLS).
    - Role-Based Access Control (RBAC) using AWS IAM / Google IAM.
    - Audit logging with AWS CloudTrail / Google Audit Logs.

6. Scalability & Disaster Recovery:
   - Elastic Load Balancing and Auto-scaling to manage high traffic.
   - Automated backups and cross-region replication for disaster recovery.

This architecture ensures the system is scalable, secure, and efficient, capable of handling large volumes of vehicle recall data while ensuring compliance and availability.

Project Wrap-up and Future Considerations:

Key Takeaways:

- Database Design: Improved understanding of relational database management, SQL queries, and table relationships.
- SQL Proficiency: Enhanced skills in writing complex queries for data analysis.
- Reporting: Learned to generate actionable insights and metrics for different personas.
- Security: Gained awareness of the importance of data privacy and security for sensitive information.

Future Considerations:

- Scalability: Emphasize optimization for larger datasets and performance.
- Automation: Plan to automate reporting and notifications.
- Integration: Explore integrating external systems for more comprehensive data analysis.
- Security: Continue prioritizing encryption and access control for sensitive data.