**Fleet Leasing Management Database System**

**Business Problem Addressed:**

* Here the business problem involves the development and management of a car rental system that is intended to manage the various parameters like administrators, car lenders, and customers, each requiring specific permissions and functionalities.
* It involves car management ensuring the detailed tracking of each vehicle’s availability and maintenance records to ensure smooth operations.
* It is also tasked with efficient reservation handling, incorporating booking details and accurate cost calculations which is accompanied by a payment processing mechanism to record and manage financial transactions.
* There is also a feedback system used to collect and analyze customer opinions and feedback for services they use.
* It also includes maintenance tracking which is keeping a log of the maintenance activities for each car in the inventory.
* Management of insurance is another business problem we are addressing which includes associating the rental transactions with relevant insurance policies.
* Also enabling administrators to generate reports for business insights addresses the business problem of getting reported feedback.

**Entities and their Relationships:**

1. **User:**

Attributes: User ID, Name, Password, User Role, Email, Contact Number.

* **Relationship:**

It is an overlapping specialization.

**Car Lender**: Here, a car lender inherits a car from the user and the lender can provide those cars for rent.

**Administrator**: Admin inherits from the user and can manage system permissions.

**Customer**: The customer also gets from the User and can make car reservations.

1. **Car Lender:**

Attributes: Lender ID, Car ID, Insurance.

* **Relationship:**

It is a One-to-many relationship, here a lender can lend multiple cars and a car belongs to at least one lender.

1. **Administrator:**

* **Relationships:**

It is a one-to-many relationship where one administrator can have one or more cars under his administration, but each car must have one administrator only. Moreover, the admin will have the ability to post/update a car on the portal.

Admin will be able to access the report for a specific car.

1. **Customer:**

Attributes:Registration Date, License Number.

* **Relationships:**

Reservation:One-to-many relationship. A customer can have multiple reservations.

Feedback:One-to-many relationship. A customer can provide feedback for each rental.

1. **Car:**

Attributes: Car ID, Model Name, Make, Year, VIN, Colour, Reservation Status, Mileage, Rate/Hour.

* **Relationship:**

Car Availability:One-to-many relationship. A car can have multiple availability records.

Maintenance Log:One-to-many relationship. A car can have multiple maintenance entries.

Reservation:One-to-many relationship. A car can be associated with multiple reservations.

Feedback:One-to-many relationship. A car can have multiple feedback entries.

User: One-to-many relationship. A user can view multiple cars.

1. **Car Availability:**

Attributes: Availability ID, Availability Date/Time, Car ID, Availability Status.

* **Relationships:**

A One-to-many relationship. Multiple cars can be available for rent, but only one car can be chosen for availability for one customer at a time.

1. **Reservation:**

Attributes: Reservation ID, Customer ID, Car ID, Pickup Date, Return Date, Reservation Status, Cost, Additional Cost.

* **Relationships:**

A user can make either many or no reservations, but one reservation can be made by only one user.

Payment: One-to-one relationship. Each reservation has only one corresponding payment.

Insurance: Optional one-to-one relationship. Reservations can opt for various insurance types provided.

1. **Payment:**

Attributes: Payment ID, Reservation ID, Amount, Date, Payment Method, Payment Status.

* **Relationships:**

A one-to-one relationship. A payment can be made to only one reservation, and a reservation can have only one and no multiple payments.

An option one-to-one relationship. A payment may or may not be made for insurance depending on the existence of the insurance for the vehicle if the customer wishes to buy one.

1. **Insurance:**

Attributes: Insurance ID, Type, Premium.

* **Relationships:**

An optional one-to-one relationship. A car reservation may or may not have insurance depending on the needs of the customer, and vice versa.

1. **Feedback:**

Attributes: Feedback ID, Car ID, Customer ID, Date, Comments, Rating.

* **Relationships:**

A one-to-optional many relationship. A customer can have multiple feedback or no feedback about their car rental service. But each feedback given must be by a customer.

1. **Report:**

Attributes:Report ID, Summary.

* **Relationships:**

A many-to-many relationship. A maintenance log can have multiple reports.

1. **Maintenance Log:**

Attributes: Maintenance ID, Car ID, Date, Type, Cost, Description.

* **Relationships:**

A one-to-many relationship. A car has been logged in many maintenance logs, but each maintenance log can have only one car.

**Key Design Decisions:**

* The User entity acts as a supertype for the Car Lender, Administrator, and Customer to reduce redundancy and ensure consistency in user-related data.
* The database design adheres to normalization rules to improve data integrity.
* Entities like Reservation include derived attributes like cost, to simplify reporting and querying.
* The Feedback entity is directly connected to both the Customer and Car to facilitate quality control and service improvement based on customer experiences.
* A separate entity for the maintenance log is included to keep a history of car maintenance, which is essential for fleet management and operational efficiency.