CS-GY 6823

Database Systems

Daniel Rajaram

Welcome to our Mobile Car Wash & Detailing Service!

Explore our services and book your appointment today.

Enjoy a clean car from the comfort of your home!

Book Appointment | Service/Package Prices | Check Appointment | Customer Reviews

ER Diagram

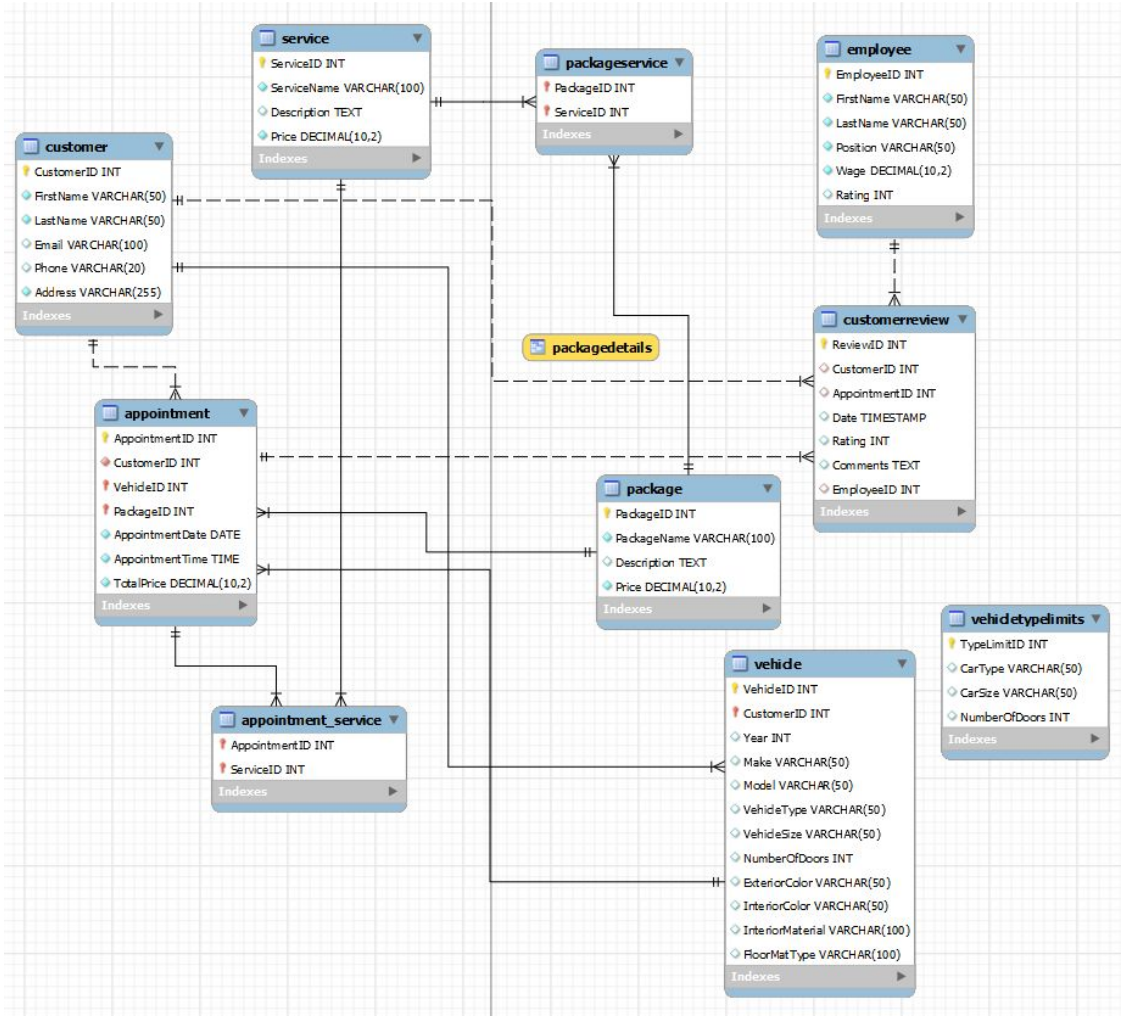
- ER Diagram Overview:
  - Illustrates the relationships between entities in the database.
  - Highlights one-to-many and one-and-only relationships.

- One-to-Many Relationships:
  - Examples include:
    - Customer to Appointments.
    - Vehicle to Appointments.

- One-and-Only Relationships:
  - Emphasizes unique associations between entities.
  - Crucial for maintaining data integrity.

- Optional Relationships:
  - Represented for non-mandatory functionalities on the website.
  - Examples:
    - Customer Review.
    - Appointment Check.

- Mandatory Relationships:
  - Essential for specific processes within the system.
  - Examples:
    - Vehicle Description for Appointment.
    - Car Wash/Detail Package for Appointment.



Vehicle

Vehicle Table:

- Purpose: Stores information about vehicles registered by customers.
- Attributes include VehicleID, CustomerID (foreign key referencing Customer table), Year, Make, Model, VehicleType, VehicleSize, NumberOfDoors, ExteriorColor, InteriorColor, InteriorMaterial, FloorMatType.
- Primary key: VehicleID.
- Foreign key: CustomerID ensures a one-to-many relationship with the Customer table, enabling association between vehicles and their owners.
- Integrity enforcement: Foreign key constraint on CustomerID ensures referential integrity, ensuring that every vehicle is associated with a valid customer, and cascading delete (ON DELETE CASCADE) ensures that if a customer is deleted, their associated vehicles are also deleted for data consistency.

vehicletypelimits Table:

- Purpose: Stores limits and specifications related to different vehicle types.
- Attributes include TypeLimitID, CarType, CarSize, NumberOfDoors.
- Primary key: TypeLimitID.
- Integrity enforcement: Primary key constraint ensures uniqueness of TypeLimitID. ForeignKey Constraint not present, as it's not connected to any other table, but it ensures data consistency within this table.

```
-- Vehicle table
CREATE TABLE Vehicle (
    VehicleID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT NOT NULL,
    Year INT,
    Make VARCHAR(50),
    Model VARCHAR(50),
    VehicleType VARCHAR(50),
    VehicleSize VARCHAR(50),
    NumberOfDoors INT,
    ExteriorColor VARCHAR(50),
    InteriorColor VARCHAR(50),
    InteriorMaterial VARCHAR(100),
    FloorMatType VARCHAR(100),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE
);
```

```
CREATE TABLE `vehicletypelimits` (
  `TypeLimitID` int NOT NULL AUTO_INCREMENT,
  `CarType` varchar(50) DEFAULT NULL,
  `CarSize` varchar(50) DEFAULT NULL,
  `NumberOfDoors` int DEFAULT NULL,
  PRIMARY KEY (`TypeLimitID`)
);
```

Appointment

Appointment Table:

- Manages appointments made by customers.
- Primary key: AppointmentID.
- Foreign keys: VehicleID and PackageID.
- Establishes a one-to-many relationship with the Vehicle table and Package table respectively.
- Stores appointment date, time, total price, and status.
- Integrity enforcement used through foreign key constraints to maintain data consistency.

Appointment Service Table:

- Represents the many-to-many relationship between appointments and services.
- Contains foreign keys AppointmentID and ServiceID.
- Ensures that each appointment can have multiple services and vice versa.
- Integrity enforced through foreign key constraints.
- Helps in managing services associated with each appointment efficiently.

```sql
-- Appointment table
CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY AUTO_INCREMENT,
    PackageID INT,
    AppointmentDate DATE,
    AppointmentTime TIME,
    TotalPrice DECIMAL(10, 2) NOT NULL,
    Status VARCHAR(20) NOT NULL,
    VehicleID INT,
    FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID) ON DELETE CASCADE,
    FOREIGN KEY (PackageID) REFERENCES Package(PackageID) ON DELETE CASCADE,
    FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID) ON DELETE CASCADE
);

CREATE TABLE Appointment_Service (
    AppointmentID INT,
    ServiceID INT,
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID) ON DELETE CASCADE,
    FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID) ON DELETE CASCADE
);
```

Package and Service

Package Table:

- Purpose: Manages details of car wash packages available for customers.
- Attributes include PackageID, PackageName, Description, Price.
- Primary key: PackageID.
- Integrity enforcement:
- Primary key constraint ensures uniqueness of PackageID.
- Check constraint ensures Price is greater than 0 for data accuracy.
- Maintains the integrity of package information within the database.

Service Table:

- Purpose: Stores information about car wash services offered.
- Attributes include ServiceID, ServiceName, Description, Price.
- Primary key: ServiceID.
- Integrity enforcement:
- Primary key constraint ensures uniqueness of ServiceID.
- Check constraint ensures Price is greater than 0 for data accuracy.
- Ensures the integrity of service information within the database.

```sql
-- Package table
CREATE TABLE Package (
    PackageID INT PRIMARY KEY AUTO_INCREMENT,
    PackageName VARCHAR(100) NOT NULL,
    Description TEXT,
    Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0)
);
```

PackageID	PackageName	Description	Price
1	Basic Wash	Exterior wash, including body and windows.	25.00
2	Interior Clean	Interior vacuuming and wipe down of surfaces.	30.00
3	Full Wash	Complete exterior and interior cleaning.	50.00
4	Premium Wash	Full wash with additional waxing and polishing.	70.00
5	VIP Wash	Premium wash with added luxury services.	100.00
6	Basic Detail	Thorough cleaning of exterior and interior surfa...	150.00
7	Interior Detail	Deep cleaning of interior surfaces and upholstery.	120.00
8	Full Detail	Comprehensive cleaning and detailing of both in...	200.00
9	Premium Detail	Full detail with added protection and shine enha...	300.00
10	VIP Detail	Top-of-the-line detailing with premium services.	450.00

```sql
-- Service table
CREATE TABLE Service (
    ServiceID INT PRIMARY KEY AUTO_INCREMENT,
    ServiceName VARCHAR(100) NOT NULL,
    Description TEXT,
    Price DECIMAL(10, 2) NOT NULL CHECK (Price > 0)
);
```

ServiceID	ServiceName	Description	Price
1	Basic Wheel and Tire Cleaning	Cleaning of wheels and tires to remove dirt and ...	20.00
2	VIP Wheel and Tire Cleaning	Advanced cleaning and detailing for wheels and...	30.00
3	Tire Shine	Application of tire shine product for a glossy fini...	15.00
4	Wheel Polish	Polishing of wheels to remove oxidation and res...	25.00
5	Undercarriage Wash	Cleaning of the vehicle's undercarriage to remo...	30.00
6	RainX	Application of RainX for improved water repelle...	20.00
7	Paint Scratch Removal	Professional removal of minor paint scratches a...	50.00
8	Triple Polish Wax	Application of three layers of polish wax for dee...	40.00
9	Ceramic Coating	Application of ceramic coating for long-lasting p...	150.00
10	Shampoo Floor Mats	Deep cleaning of floor mats to remove stains an...	25.00
11	Recondition Leather	Restoration and conditioning of leather surfaces.	60.00
12	Tough Stains Removal	Specialized treatment to remove tough stains fr...	40.00
13	Steam Cleaning	Thorough cleaning using steam for deep penetr...	70.00
14	Headlight Restoration	Restoration of cloudy or faded headlights for im...	45.00
15	Convertible Top Cleaning	Cleaning and conditioning of convertible tops.	55.00
16	Odor Elimination	Treatment to remove odors from the vehicle int...	80.00
17	Engine Bay Wash	Cleaning and degreasing of the engine bay area.	50.00
18	Paint Protection Film	Application of protective film to prevent paint d...	200.00
19	RIM Restoration	Restoration and refinishing of alloy rims.	75.00
20	Clay Bar Treatment	Clay bar treatment to remove contaminants an...	60.00
21	Wipe and Shine Trim	All plastics, wood, etc in the interior will be dust...	25.00

Package/Service Table

PackageService Table:

- Purpose: Facilitates the association between car wash packages and services.
- Attributes include PackageID and ServiceID.
- Primary key: Composite key (PackageID, ServiceID) ensures uniqueness of package-service associations.
- Foreign keys:
- PackageID references Package(PackageID), establishing a many-to-many relationship between packages and services.
- ServiceID references Service(ServiceID).
- Integrity enforcement:
- Primary key constraint ensures uniqueness of package-service combinations.
- Foreign key constraints maintain referential integrity with the Package and Service tables.
- Ensures accurate mapping between car wash packages and the services they include.

```sql
-- Package-Service Association Table
CREATE TABLE PackageService (
    PackageID INT,
    ServiceID INT,
    PRIMARY KEY (PackageID, ServiceID),
    FOREIGN KEY (PackageID) REFERENCES Package(PackageID),
    FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID)
);
```

```sql
-- Populate the PackageService table with associations between packages and services.
INSERT INTO PackageService (PackageID, ServiceID) VALUES
(1, 1), (1, 5),
(2, 10),(2,21),
(3, 1), (3, 5), (3, 10),(3,21),
(4, 1), (4, 5), (4, 8), (4, 10), (4,21),
(5, 2), (5, 5), (5, 6), (5, 8),(5, 10),(5,21),
(6, 1), (6, 5), (6, 6), (6, 8), (6, 10), (6,12), (6, 13), (6, 16),(6,21),
(7, 10), (7, 11), (7, 12), (7, 13), (7, 16),(7,21),
(8, 1), (8, 5), (8, 10), (8, 11), (8, 12), (8, 13), (8, 14), (8, 16),(8,21),
(9, 1), (9, 5), (9,6),(9,9), (9, 10), (9, 11), (9, 12), (9, 13), (9, 14), (9, 16),(9,17),(9,21),
(10, 2), (10,3), (10,4), (10, 5), (10,6), (10,7), (10,8), (10,9), (10, 10), (10, 11), (10, 12), (10, 13), (10, 14),
```

Customer and Review Tables

Customer Table:

- Purpose: Stores information about customers of the car wash.
- Attributes include CustomerID, FirstName, LastName, Email, Phone, Address.
- Primary key: CustomerID ensures unique identification of each customer.
- Integrity enforcement:
- Primary key constraint maintains uniqueness of CustomerID.
- UNIQUE constraint on Email and Phone ensures uniqueness of these contact details.
- Ensures accurate record-keeping and contact information for customers.

```sql
-- Customer table
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(20) UNIQUE,
    Address VARCHAR(255) NOT NULL
);
```

CustomerReview Table:

- Purpose: Records reviews submitted by customers for their appointments.
- Attributes include ReviewID, CustomerID, AppointmentID, Date, Rating, Comments, EmployeeID.
- Primary key: ReviewID uniquely identifies each review.
- Integrity enforcement:
- Primary key constraint ensures uniqueness of ReviewID.
- CHECK constraint ensures Rating is within the range of 1 to 10 for data accuracy.
- Foreign key constraints:
- CustomerID references Customer(CustomerID), ensuring that each review is associated with a valid customer.
- AppointmentID references Appointment(AppointmentID), linking each review to a specific appointment.
- EmployeeID references Employee(EmployeeID), indicating the employee involved in the reviewed appointment.
- Facilitates feedback management, performance evaluation of employees, and enhances customer experience by allowing them to share their opinions.

```sql
-- Customer Review table
CREATE TABLE CustomerReview (
    ReviewID INT PRIMARY KEY,
    CustomerID INT,
    AppointmentID INT,
    Date DATE,
    Rating INT CHECK (Rating >= 1 AND Rating <= 10),
    Comments TEXT,
    EmployeeID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
```

Employee Table

Employee Table:

- Purpose: Stores information about employees working at the car wash.
- Attributes include EmployeeID, FirstName, LastName, Position, Wage, Rating.
- Primary key: EmployeeID uniquely identifies each employee.
- Integrity enforcement:
- Primary key constraint ensures uniqueness of EmployeeID.
- NOT NULL constraints on FirstName, LastName, Position, Wage ensure these fields are always populated.
- CHECK constraint on Wage ensures it's non-negative for accurate wage representation.
- CHECK constraint on Rating ensures it falls within the range of 1 to 10, providing a valid rating scale.
- Ensures accurate record-keeping and management of employee details within the car wash system.

```sql
-- Employee table
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Position VARCHAR(50) NOT NULL,
    Wage DECIMAL(10, 2) NOT NULL CHECK (Wage >= 0),
    Rating INT CHECK (Rating >= 1 AND Rating <= 10),
    FOREIGN KEY (EmployeeID) REFERENCES CustomerReview(EmployeeID)
);
```

View

PackageDetails View:

- Purpose: Provides a comprehensive overview of car wash packages along with the services included in each package.
- PackageID: Unique identifier for each package.
- PackageName: Name of the package.
- PackageDescription: Description of the package.
- PackagePrice: Price of the package.
- IncludedServices: Concatenated list of services included in the package, presented as a single string.
- TotalServices: Count of the total number of services included in the package.
- The view is constructed by joining the Package, PackageService, and Service tables.
- Grouped by PackageID to aggregate services for each package.
- Provides valuable information for customers, allowing them to easily compare packages and understand what services are included in each one.
- Enhances the user experience by presenting package details in a clear and concise manner, facilitating informed decision-making.

```sql
-- View to show packages along with their included services
CREATE VIEW PackageDetails AS
SELECT
    p.PackageID,
    p.PackageName,
    p.Description AS PackageDescription,
    p.Price AS PackagePrice,
    GROUP_CONCAT(s.ServiceName) AS IncludedServices,
    COUNT(s.ServiceName) AS TotalServices
FROM Package p
JOIN PackageService ps ON p.PackageID = ps.PackageID
JOIN Service s ON ps.ServiceID = s.ServiceID
GROUP BY p.PackageID;
```

Get Vehicle Sizes Function

- Purpose: Retrieves the available sizes for a given type of vehicle.
- Main attributes:
- Input parameter: car_type (VARCHAR), specifies the type of vehicle.
- Return value: vehicle_sizes (VARCHAR), a string containing the available sizes for the specified vehicle type.
- The function uses a CASE statement to determine the available sizes based on the input car_type.
- If the input car_type matches one of the predefined vehicle types (Coupe, Sedan, Convertible, Hatchback, SUV/Crossover, Van/Minivan, Pickup Truck), the function sets the vehicle_sizes variable accordingly.
- If the input car_type does not match any of the predefined types, an empty string is returned.
- Enhances the car wash database by providing a convenient way to retrieve information about vehicle sizes, which can be used for various purposes such as pricing, service selection, and resource allocation.

```sql
-- GetVehicleSizes Function
DELIMITER //

CREATE FUNCTION GetVehicleSizes(car_type VARCHAR(50))
RETURNS VARCHAR(255)
BEGIN
    DECLARE vehicle_sizes VARCHAR(255);

    CASE car_type
        WHEN 'Coupe' THEN
            SET vehicle_sizes = 'Small size; Medium size';
        WHEN 'Sedan' THEN
            SET vehicle_sizes = 'Small size; Medium size; Large size';
        WHEN 'Convertible' THEN
            SET vehicle_sizes = 'Small size; Medium size';
        WHEN 'Hatchback' THEN
            SET vehicle_sizes = 'Small size; Medium size';
        WHEN 'SUV/Crossover' THEN
            SET vehicle_sizes = 'Small size; Medium size; Large size';
        WHEN 'Van/Minivan' THEN
            SET vehicle_sizes = 'Medium size; Large size';
        WHEN 'Pickup Truck' THEN
            SET vehicle_sizes = 'Medium size; Large size';
        ELSE
            SET vehicle_sizes = '';
    END CASE;

    RETURN vehicle_sizes;
END;

DELIMITER ;
```

Get Number Of Doors Function

- Purpose: Retrieves the number of doors available for a given type of vehicle.
- Main attributes:
- Input parameter: car_type (VARCHAR), specifies the type of vehicle.
- Return value: num_doors (VARCHAR), a string containing the number of doors for the specified vehicle type.
- The function uses a CASE statement to determine the number of doors based on the input car_type.
- If the input car_type matches one of the predefined vehicle types (Coupe, Sedan, Convertible, Hatchback, SUV/Crossover, Van/Minivan, Pickup Truck), the function sets the num_doors variable accordingly.
- If the input car_type does not match any of the predefined types, an empty string is returned.
- Enhances the car wash database by providing a convenient way to retrieve information about the number of doors for different vehicle types, which can be useful for various purposes such as service selection, resource allocation, and customer communication.

```
-- GetNumberOfDoors Function
DELIMITER //
CREATE FUNCTION GetNumberOfDoors(car_type VARCHAR(50))
RETURNS VARCHAR(255)
BEGIN
    DECLARE num_doors VARCHAR(255);

    CASE car_type
        WHEN 'Coupe' THEN
            SET num_doors = '2 doors';
        WHEN 'Sedan' THEN
            SET num_doors = '4 doors';
        WHEN 'Convertible' THEN
            SET num_doors = '2 doors';
        WHEN 'Hatchback' THEN
            SET num_doors = '3 doors; 5 doors';
        WHEN 'SUV/Crossover' THEN
            SET num_doors = '4 doors; 5 doors';
        WHEN 'Van/Minivan' THEN
            SET num_doors = '3 doors; 4 doors; 5 doors';
        WHEN 'Pickup Truck' THEN
            SET num_doors = '2 doors; 4 doors; 5 doors';
        ELSE
            SET num_doors = '';
    END CASE;

    RETURN num_doors;
END;

DELIMITER ;
```

Calculate Adjusted Package Price Function

- Purpose: Calculates the adjusted price for a car wash package based on the package's base price and the type of vehicle being serviced.
- Main attributes:
- Input parameters: packageID (INT), specifies the ID of the package; vehicleTypeLimitID (INT), specifies the ID of the vehicle type limit.
- Return value: adjustedPrice (DECIMAL), the adjusted price of the package after considering the vehicle type price adjustment.
- The function retrieves the base price of the package from the PackageService table based on the provided packageID.
- It then determines the vehicle type associated with the given vehicleTypeLimitID and applies a price adjustment based on the type of vehicle.
- The price adjustment varies depending on the type of vehicle, with specific adjustments for Convertible, SUV/Crossover, Van/Minivan, and Pickup Truck.
- The adjusted price is calculated by adding the base price and the vehicle type price adjustment.
- Enhances the car wash database by providing a dynamic pricing mechanism that considers the type of vehicle being serviced, ensuring fair and accurate pricing for customers.

```sql
-- Calculate_Adjusted_Package_Price Function
DELIMITER //
CREATE FUNCTION Calculate_Adjusted_Package_Price(packageID INT, vehicleTypeLimitID INT)
RETURNS DECIMAL(10, 2)
BEGIN
    DECLARE basePrice DECIMAL(10, 2);
    DECLARE adjustedPrice DECIMAL(10, 2);
    DECLARE vehicleTypePriceAdjustment DECIMAL(10, 2);
    DECLARE vehicleType VARCHAR(50);

    -- Get the base price for the package
    SELECT p.Price INTO basePrice
    FROM PackageService ps
    JOIN Package p ON ps.PackageID = p.PackageID
    WHERE ps.PackageID = packageID
    ORDER BY ps.PackageID LIMIT 1;

    -- Get the vehicle type based on TypeLimitID
    SELECT CarType INTO vehicleType FROM vehicletypelimits WHERE TypeLimitID = vehicleTypeLimitID;

    -- Determine the vehicle type price adjustment
    SELECT
        CASE
            WHEN vehicleType IN ('Convertible') THEN 5.00 -- Adjust price for Convertible
            WHEN vehicleType IN ('SUV/Crossover') THEN 10.00 -- Adjust price for SUV/Crossover
            -- Adjust price for Van/Minivan and Pickup Truck
            WHEN vehicleType IN ('Van/Minivan', 'Pickup Truck') THEN 15.00
            ELSE 0.00 -- No adjustment for other vehicle types
        END INTO vehicleTypePriceAdjustment;

    -- Calculate adjusted price
    SET adjustedPrice = basePrice + vehicleTypePriceAdjustment;
    RETURN adjustedPrice;
END;
DELIMITER ;
```

Adjust Employee Wage Procedure

- Purpose: Automatically adjusts the wage of an employee based on their average rating from customer reviews.
- Main attributes:
- Input parameter: empID (INT), specifies the ID of the employee whose wage needs adjustment.
- The procedure calculates the average rating for the employee by querying the CustomerReview table for reviews associated with the employee specified by empID.
- It then retrieves the current wage of the employee from the Employee table.
- If the average rating is 8 or above, indicating exceptional performance, the procedure increases the employee's wage by 5% or more.
- Finally, it updates the employee's wage in the Employee table with the adjusted value.
- The procedure helps incentivize and reward employees for delivering excellent service, thereby promoting employee satisfaction and enhancing the quality of service provided by the car wash.

```
-- AdjustEmployeeWage Procedure
DELIMITER //

CREATE PROCEDURE AdjustEmployeeWage(IN empID INT)
BEGIN
    DECLARE avgRating DECIMAL(5,2);
    DECLARE empWage DECIMAL(10,2);

    -- Calculate the average rating for the employee
    SELECT AVG(Rating) INTO avgRating
    FROM CustomerReview
    WHERE EmployeeID = empID;

    -- Retrieve the current wage of the employee
    SELECT Wage INTO empWage
    FROM Employee
    WHERE EmployeeID = empID;

    -- Check if the average rating is 8 or above
    IF avgRating >= 8 THEN
        -- Increase the wage by 5% or more
        SET empWage = empWage * 1.05;
        -- Update the employee's wage in the Employee table
        UPDATE Employee
        SET Wage = empWage
        WHERE EmployeeID = empID;
    END IF;
END//

DELIMITER ;
```

Trigger

- Trigger Name: Update_Employee_Rating
- Event: AFTER INSERT ON CustomerReview
- Action Time: FOR EACH ROW
- Purpose: Automatically updates the rating of an employee in the Employee table whenever a new review is inserted into the CustomerReview table.
- Dynamic Rating Calculation: Calculates the average rating for the employee based on all reviews associated with them.
- Functionality:
- Upon each insertion of a new review into the CustomerReview table, the trigger calculates the average rating for the employee corresponding to the inserted review.
- It then updates the employee's rating in the Employee table with the newly calculated average rating.
- Data Integrity: Ensures that employee ratings are always up-to-date and accurately reflect the average of all reviews received.
- Efficiency: Provides an automated mechanism for updating employee ratings, eliminating the need for manual intervention and ensuring consistency across the database.

```
CREATE TRIGGER Update_Employee_Rating
AFTER INSERT ON CustomerReview
FOR EACH ROW
BEGIN
    DECLARE avg_rating DECIMAL(5,2);

    -- Calculate the average rating for the employee
    SELECT AVG(Rating) INTO avg_rating
    FROM CustomerReview
    WHERE EmployeeID = NEW.EmployeeID;

    -- Update the employee's rating in the Employee table
    UPDATE Employee
    SET Rating = avg_rating
    WHERE EmployeeID = NEW.EmployeeID;
END;
```

Normalization

First Normal Form (1NF):

- Each table has a primary key: CustomerID in Customer table, VehicleID in Vehicle table, etc.
- Each column holds atomic values: FirstName, LastName in Customer table, AppointmentDate, AppointmentTime in Appointment table, etc.

Second Normal Form (2NF):

- No partial dependencies exist.
- All non-key attributes are fully functional dependent on the primary key.
- For example, in the Appointment table, AppointmentDate and AppointmentTime depend only on the AppointmentID, not on any subset of the primary key.

Third Normal Form (3NF):

- No transitive dependencies exist.
- All non-key attributes depend only on the primary key, not on other non-key attributes.
- For example, in the Vehicle table, VehicleType and VehicleSize depend only on VehicleID, not on each other.

Fourth Normal Form (4NF):

- No multi-valued dependencies exist.
- Tables are further decomposed to remove multi-valued dependencies.
- For example, the PackageService table separates package-service associations to eliminate multi-valued dependencies.

Isolation Level

Isolation Level Used: REPEATABLE-READ

Where: Throughout the carwash database, especially in transactions involving critical operations such as appointments, customer reviews, and employee data.

Why:

Consistency: Ensures that within a transaction, any data read remains consistent throughout the transaction, even if the same data is read multiple times.

Prevents Phantom Reads: By locking the rows that a transaction reads until the transaction completes, it prevents phantom reads where a transaction reads a set of rows multiple times and sees different results each time due to changes made by other transactions.

Suitable for the Carwash Operations: In the context of a carwash database, where appointments and employee information are frequently accessed and updated, the REPEATABLE-READ isolation level provides a balance between concurrency and consistency, ensuring data integrity without excessively locking resources.