CAR RENTAL SYSTEM

DATABASE DESIGN (CS6360-002)

GROUP 3

FALL 2017

Abhijit Jinendrappa (axj176230)

Raj Chauhan (rsc150330)

Yi Li (yxl121030)

o

INDEX

1. REQUIREMENTS

1.1 Introduction

In this project, we aim to design a database which captures the major business activities and transactions in the American car rental business. The car rental companies typically purchase or lease many fleet vehicles and rent them to the customers for a fee. The first known operation in the United States happened in 1916, when Joe Saunders of Omaha, Nebraska started a rental business with only one borrowed Model T Ford. Nevertheless, the sector subsequently expanded rapidly and by 2016, consists of a significant market with 2.3 million cars in services, 20,469 service locations, and over 28.4 billion dollars annual revenue. Today, major US car rental companies include Enterprise Holdings and Hertz Global Holdings, which are in the Fortune 500 list.

1.2 Requirements

a) The car rental company operates multiple branches.

b) Each branch owns a fleet of cars for rent.

c) Each car belongs to a particular category such as sedans, vans, SUVs (sports utility vehicles) or trucks.

d) Customer rents cars from a specific branch.

e) Customers can choose to sign up for the membership, and in such cases, certain personal information (login ID, password, credit card account information) will be collected.

f) The car rental company may offer certain rental promotions, and in such cases, the promotion content (description of the promotion, starting date, ending date) will be available.

g) Both the member and non-member(guest) customers can make reservations online.

h) For each transaction, additional drivers can be added into the contract, provided they have the required legal documents and meet other restrictions such as being 21 years of age or older.

i) For each car rented, the rental insurance can be provided by the car rental company, with both regular and premium options.

j) The bills are generated after the car is returned.

k) For the payment, cash, credit cards, or debit cards are accepted. Personal checks or money orders are not allowed.

l) A late fee may occur and be added to the final cost if the customer returns the car after the due date.

m) A customer can return the car before the due date. However, the cost will be still based on the previously agreed length of the car rental transaction.

n) A default 8.25% sales tax will be applied to the final billing.

o) Car rental price will be calculated based on the selected make and model.

1.2. ENTITIES

a) Branch

The car rental company operates its business in multiple branches. The attributes for the Branch include: Branch ID (unique), Manager ID, Phone, Email ID, Phone and Address (Street, State, City, Zip).

b) Car

The car entity will specify the list of cars available in the system, which are further categorized into four groups based on the car types: Sedan, Van, SUV (sports utility vehicle) and Truck. The attributes for the Car include: Registration (unique, Registered State, Registration Number), VIN (unique), Model, Make, Year, Mileage, Color, Cost-per-day, Late-fee-per-day, Passenger Capacity, and Availability Flag. In addition, the Truck entity has an attribute of Loading Capacity. The Sedan entity has an attribute of Exotic Flag. The VAN entity has an attribute of Cargo Flag. The SUV entity has an attribute of All-wheel Drive Flag.

c) Customer

Customers will rent the car from one of the branches. Customer can opt to apply for the membership offered by the car rental company (Member or Guest subclasses). The attributes for Customer include: Driver License Number (unique), Name (Fname, Minit, Lname), Phone (Primary, Secondary), Email ID, Address (Street, City, State, Zip Code) and Date-of-birth. The derived attribute is Age. For the Member entity, the additional attributes include: Login ID (unique), Password, Membership Validity.

d) Card Information:

The members of the company will save their credit/debit card information in the database. The attributes for the dependent entity Card Information include: Card Number (unique), Card Type, Card Holder Name (Fname, Minit, Lname), Expiration Date, CVV, Billing Address (Street, City, State, Zip Code).

e) Promotion

The car rental company regularly offers certain promotions to their customers. The attributes for Promotion include: Promotion Code (unique), Description, Start Date, End Date, Discount Type, Discount Value.

f) Rental Insurance

The customer can choose to purchase the car rental insurance when picking up the rented vehicles, which include regular and premium options. The attributes for Rental Insurance include: Insurance Code (unique), Bodily Injury Coverage, Property Damage Coverage, Rate-per-day. In addition, the Premium Rental Insurance entity has two additional attributes: Roadside Assistance Coverage and Towing Coverage.

g) Additional Driver

For each car rental transaction, the customer may request to include additional drivers into the contract. In such cases, the dependent entity Additional Driver has attributes including: Driver License Number (unique), Name (Fname, Minit, Lname), Date-of-birth and a derived attribute Age.

h) Booking

For each car rental transaction (booking), the attributes include: Booking ID (unique), Start Date, End Date, Actual End Date, Status (Booked, Cancelled, Active, Completed), Start Mileage, End Mileage, Booking Amount.

i) Billing

The Billing has two subclasses: Booking Billing and Membership Billing. For the Booking Billing, a final bill will be generated when the customer returns the car. For the Membership Billing, a monthly bill will be generated for each customer. The attributes for Billing include: Bill ID (unique), Bill Date, Total Amount, Status (Paid, Partially_Paid, Pending, Over_Charged), Payment Mode. The Booking Bill has an additional attribute of Late Fee.

2. RELATIONS

a) Car to Branch (many-to-one):

The car rental company operates multiple branches at different locations and each car belongs to a location. The relation name is “Belongs To”.

- Each branch will maintain in their stock at least 0, at most N cars.
- Each car belongs to one and only one branch.

b) Member to Card Information (one-to-one):

The members of the car rental company will save their credit card information in the system so that their future bookings can be processed more efficiently. The relation name is “Has”.

- Each member will save one and only one active card in the system.
- Each card belongs to one and only one member.

c) Booking to Car (many-to-many):

For each car rental transaction, a booking made by the customer will have one or multiple vehicles. The relation name is “Chosen”.

- Each booking involves one or more cars.
- Each car could be booked by one or more customers, or never been chosen.

d) Customer to Booking (one-to-many):

The customers will book the rental cars for their use. The relation name is “Makes”.

- Each customer can make one or more bookings, or not make any bookings.
- Each booking involves one and only one customer.

e) Booking to Rental Insurance (many-to-many):

During the booking, the customer may choose to purchase the Rental Insurance offered by the car rental company. The relation name is “Includes”. In case the customer books more than one vehicle, multiple rental insurances will be purchased.

- Each booking involves one or more types of rental insurance, or no rental insurance at all.
- Each rental insurance offered by the branch could be added to one or more bookings, or never be used.

f) Booking to Promotion (many-to-many):

The car rental company will regularly offer promotions to their customers. The customers will be eligible to apply for any applicable promotions at the time of their bookings. The relation name is "Apply".

- Each booking may involve one or more promotions, or no promotion is applicable for that specific booking.
- Each promotion may be taken advantage of by one or many bookings, or not been used at all.

g) Booking to Branch (many-to-one):

The customer may return the car to its original branch or a different branch location. The relation name is "Drop off".

- For each booking, the rented car will be retuned back to one and only one branch.
- One or more rented cars may be returned to the branch, or no car is ever returned to that branch.

h) Booking to Additional Driver (many-to-many):

The customer can add additional eligible drivers to their bookings. The relation name is "add".

- For each booking, the customer can add one or more additional drivers, or not add anyone.
- The additional drivers could be added to one or multiple bookings.

i) Booking to Booking Billing (one-to-one):

For each car rental transaction, a single bill will be generated. The relation name is "generates".

- For each booking, one and only one bill will be generated.
- Each booking bill is derived from one and only one booking transaction.

j) Member to Membership Billing (one-to-many):

A monthly membership bill will be generated for each member. The relation name is "generates".

- For each member, one or more membership bills will be generated.
- Each membership bill is linked to one and only one member.

k) Card Information to Billing

For customers, their saved card information could be used to pay any outstanding balances from either monthly membership fees or the transaction costs. The relation name is “pays”.

- Each card could be used to pay for one or more bills, or not be used for any payments.
- For each bill, at most a single card will be used for the payment.

3. ASSUMPTIONS

a) For each booking, the customer can rent one or multiple vehicles.

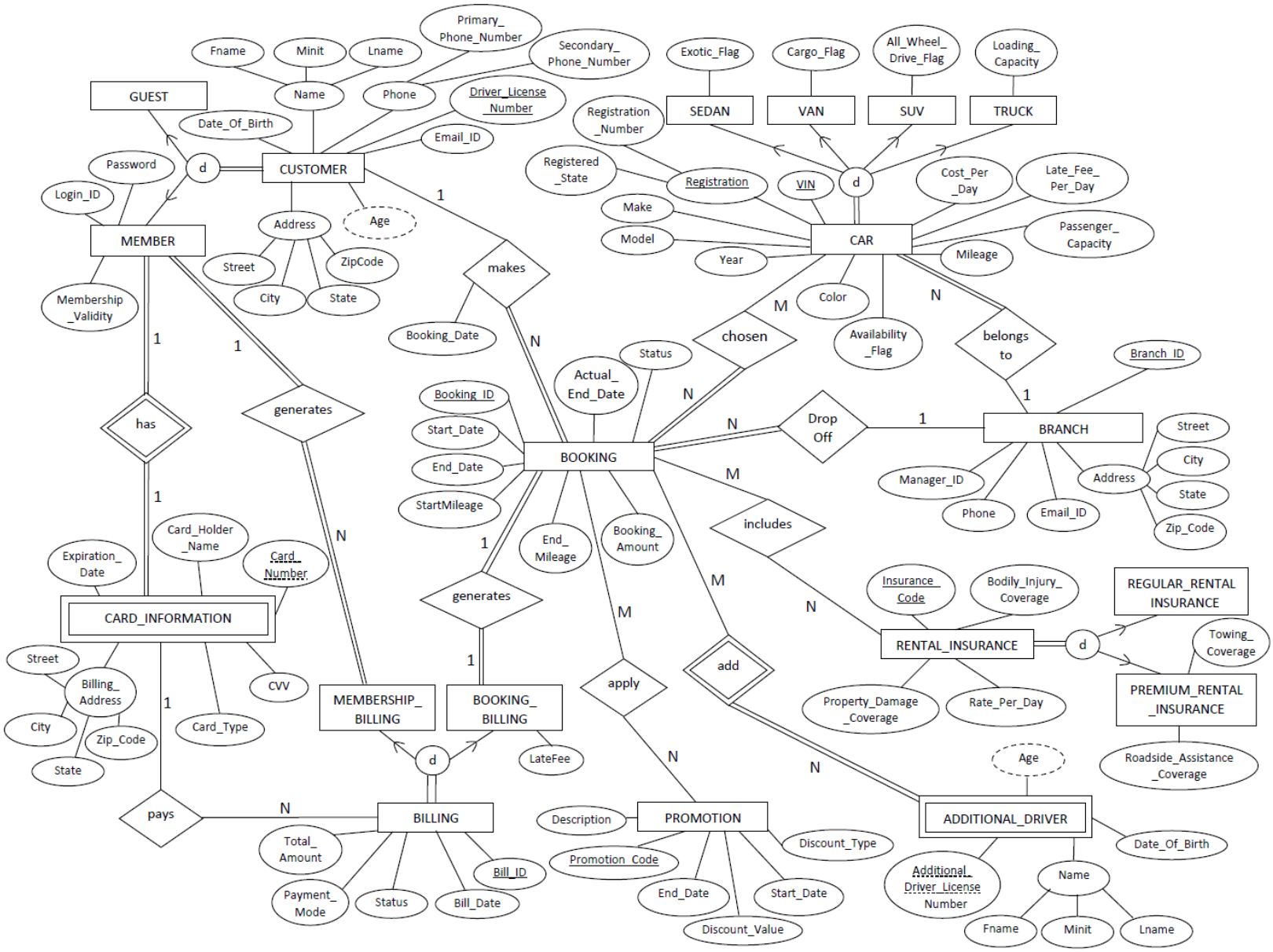
b) The member customers of the car rental company can save the information of only one active credit card in the system.

c) Multiple promotions may be applicable and offered to a specific booking transaction.

d) The customer can return the car at a different branch with no additional charge.

e) The customer can purchase the car rental insurance offered by the company or use his/her own car insurance coverage.

4. ENITY-RELATIONSHIP/ENHANCED ENTITY-RELATIONSHIP DIAGRAM



5. FUNCTIONAL DEPENDENCIES

a) CAR

- (Registered_State, Registration_Number) → VIN, Make, Model, Year, Color, Mileage, Availability_Flag, Cost_Per_Day, Late_Fee_Per_Day, Passenger_Capacity, Owning_Branch_ID
- VIN → Registered_State, Registration_Number, Make, Model, Year, Color, Mileage, Availability_Flag, Cost_Per_Day, Late_Fee_Per_Day, Passenger_Capacity, Owning_Branch_ID
- Model → Make
- Make, Model, Mileage, Year → Cost_Per_Day

b) SEDAN

- (Registered_State, Registration_Number) → Exotic_Flag

c) VAN

- (Registered_State, Registration_Number) → Cargo_Flag

d) SUV

- (Registered_State, Registration_Number) → All_Wheel_Drive_Flag

e) TRUCK

- (Registered_State, Registration_Number) → Loading_Capacity

f) BRANCH

- Branch_ID → Manager_ID, Email_ID, Phone, Street, City, State, Zip_Code
- Zip_Code → City, State

g) RENTAL_INSURANCE

- Insurance_Code → Bodily_Injury_Coverage, Property_Damage_Coverage, Rate_Per_Day

h) REGULAR_RENTAL_INSURANCE

- Insurance_Code → Insurance_Code

i) PREMIUM_RENTAL_INSURANCE

- Insurance_Code → Towing_Coverage, Roadside_Assistance_Coverage

j) ADDITIONAL_DRIVER

- (Booking_ID, Additional_Driver_License_Number) → Fname, Minit, Lname, Date_Of_Birth

k) PROMOTION

- Promotion_Code → Description, Discount_Type, Discount_Value, Start_Date, End_Date

l) BILLING

- Bill_ID → Payment_Mode, Status, Total_Amount, Bill_Date, Driver_License_Number, Card_Number

m) MEMBERSHIP_BILLING

- Bill_ID → Member_Driver_License_Number

n) BOOKING_BILLING

- Bill_ID → Booking_ID, Late_Fee

o) CARD_INFORMATION

- (Driver_License_Number, Card_Number) → Fname, Minit, Lname, CVV, Expiration_Date, Card_Type, Street, City, State, Zip_Code
- Zip_Code → City, State

p) CUSTOMER

- Driver_License_Number → Fname, Minit, Lname, , Date_Of_Birth, Primary_Phone_Number, Secondary_Phone_Number, Email_ID, Street, City, State, Zip_Code
- Zip_Code → City, State

q) MEMBER

- Driver_License_Number → Login_ID, Password, Membership_Validity

r) GUEST

- Driver_License_Number → Driver_License_Number

s) BOOKING

- Booking_ID → Start_Date, End_Date, Actual_End_Date, Start_Mileage, End_Mileage, Status, Booking_Amount, Drop_Off_Branch_ID, Driver_License_Number

t) BOOKING_RENTAL_INSURANCE

- (Booking_ID, Insurance_Code) → Booking_ID, Insurance_Code

u) BOOKING_PROMOTION

- (Booking_ID, Promotion_Code) → Booking_ID, Promotion_Code

v) BOOKING_CAR

- (Registered_State, Registration_Number, Booking_ID) → Booking_Date

6. FUNCTIONAL DEPENDENCIES VIOLATING THE 3NF RULES

a) In CAR relation, Model (non-prime attribute) → Make (non-prime attribute) violates the 3NF rule.

To normalize into the 3NF, theoretically we need to remove the Make attribute from the CAR relation and generate a new RELATION MODEL_MAKE with the functional dependency: Model → Make.

To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

b) In BRANCH relation, Zip (non-prime attribute) → City, State (non-prime attributes) violates the 3NF rule.

To normalize into the 3NF, theoretically we need to remove the City and State attributes from the BRANCH relation and generate a new RELATION ZIP_CODE with the functional dependency: Zip → City, State.

To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

c) In CARD_INFORMATION relation, Driver_License_Number (prime attribute) → State (non-prime attribute) violates the 2NF rule.

To normalize into the 3NF, theoretically we need to remove the State attribute from the CARD_INFORMATION relation and generate a new RELATION DRIVER_LICENSE_ISSUING_STATE with the functional dependency: Driver_License_Number → State.

To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

d) In CARD_INFORMATION relation, Zip (non-prime attribute) → City, State (non-prime attributes) violates the 3NF rule.

To normalize into the 3NF, theoretically we need to remove the City and State attributes from the CARD_INFORMATION relation and generate a new RELATION ZIP_CODE with the functional dependency: Zip → City, State.

To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

e) In CUSTOMER relation, Zip (non-prime attribute) → City, State (non-prime attributes) violates the 3NF rule.

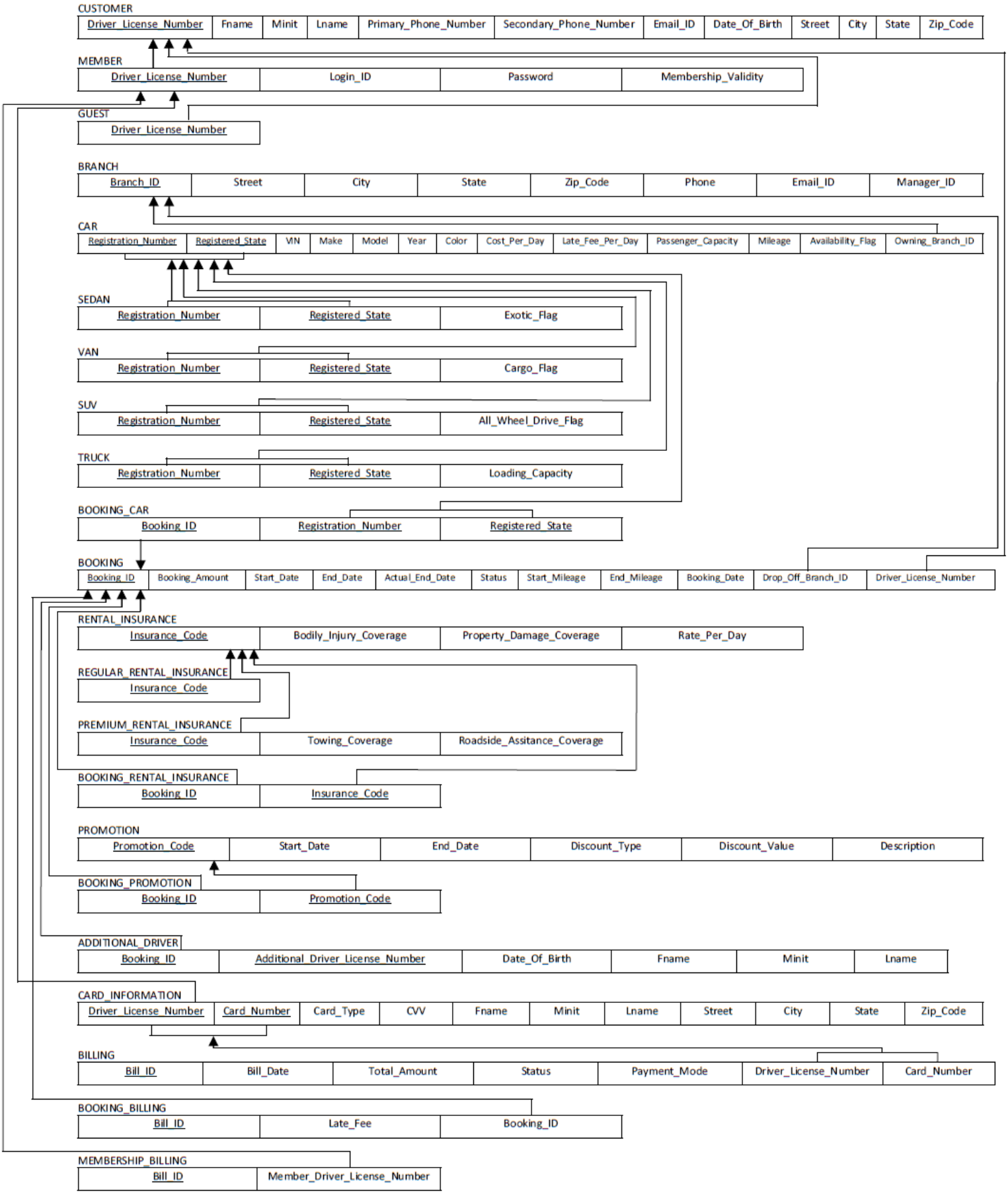
To normalize into the 3NF, theoretically we need to remove the City and State attributes from the CUSTOMER relation and generate a new RELATION ZIP_CODE with the functional dependency: Zip → City, State.

To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

f) In CAR relation, Make, Model, Mileage, Year (non-prime attribute) → Cost_Per_Day (non-prime attributes) violates the 3NF rule.

To normalize into the 3NF, theoretically we need to remove the Cost_Per_Dday attribute from the CAR relation and generate a new RELATION PRICING with the functional dependency: Make, Model, Mileage, Year → Cost_Per_Day. To make our final schema design more compact, we opt to de-normalize this 3NF back to 2NF and keep the original design.

7. RELATIONAL SCHEMA



a) CAR

Primary Key: (Registered_State, Registration_Number)

Foreign Key: Owning_Branch_ID refers to Branch(Branch_ID)

b) SEDAN

Primary Key: (Registered_State, Registration_Number)

c) VAN

Primary Key: (Registered_State, Registration_Number)

d) SUV

Primary Key: (Registered_State, Registration_Number)

e) TRUCK

Primary Key: (Registered_State, Registration_Number)

f) BRANCH

Primary Key: Branch_ID

g) RENTAL_INSURANCE

Primary Key: Insurance_Code

h) REGULAR_RENTAL_INSURANCE

Primary Key: Insurance_Code

i) PREMIUM_RENTAL_INSURANCE

Primary Key: Insurance_Code

j) ADDITIONAL_DRIVER

Primary Key: (Booking_ID, Additional_Driver_License_Number)

k) PROMOTION

Primary Key: Promotion_Code

l) BILLING

Primary Key: Bill_ID

Foreign Key: (Dirver_License_Number, Card_Number) refers to CARD_INFORMATION(Driver_License_Number, Card_Number)

m) MEMBERSHIP_BILLING

Primary Key: Bill_ID

Foreign Key: Member_Driver_License_Number refers to MEMBER(Driver_License_Number)

n) BOOKING_BILLING

Primary Key: Bill_ID

Foreign Key: Booking_ID refers to BOOKING(BOOKING_ID)

o) CARD_INFORMATION

Primary Key: (Driver_License_Number, Card_Number)

Foreign Key: Driver_License_Number refers to MEMBER(Driver_License_Number)

p) CUSTOMER

Primary Key: Driver_License_Number

q) MEMBER

Primary Key: Driver_License_Number

r) GUEST

Primary Key: Driver_License_Number

s) BOOKING

Primary Key: Booking_ID

Foreign Key: Drop_Off_Branch_ID refers to BRANCH(Branch_ID)

Foreign Key: Driver_License_Number refers to MEMBER(Driver_License_Number)

t) BOOKING_RENTAL_INSURANCE

Primary Key: (Booking_ID, Insurance_Code)

Foreign Key: Booking_ID refers to BOOKING(Booking_ID)

Foreign Key: Insurance_Code refers to RENTAL_INSURANCE(Insurance_Code)

u) BOOKING_PROMOTION

Primary Key: (Booking_ID, Promotion_Code)

Foreign Key: Booking_ID refers to BOOKING(Booking_ID)

Foreign Key: Promotion_Code refers to PROMOTION(Promotion_Code)

v) BOOKING_CAR

Primary Key: (Registered_State, Registration_Number, Booking_ID)

Foreign Key: Booking_ID refers to BOOKING(Booking_ID)

Foreign Key: (Registered_State, Registration_Number) refers to CAR(Registered_State, Registration_Number)

8. SQL STATEMENTS FOR CREATING THE TABLES

```
CREATE TABLE car (
   registered_state        VARCHAR(15) NOT NULL,
   registration_number   VARCHAR(25) NOT NULL,
   vin                          NUMBER(25) NOT NULL,
   model                      VARCHAR(25) NOT NULL,
   make                       VARCHAR(25) NOT NULL,
   color                       VARCHAR(25) NOT NULL,
   year                        NUMBER(4) NOT NULL,
   mileage                    NUMBER(6) NOT NULL,
   owning_branch_id      NUMBER(6) NOT NULL,
   availability_flag          CHAR(1) DEFAULT 'Y' NOT NULL,
   passenger_capacity      NUMBER(2) NOT NULL,
   cost_per_day              NUMBER(7,2) NOT NULL CHECK (cost_per_day > 0),
   late_fee_per_day        NUMBER(7,2) NOT NULL,
   CONSTRAINT car_pk PRIMARY KEY ( registered_state, registration_number )
);

CREATE TABLE customer (
   driver_license_number        VARCHAR(25) NOT NULL,
   primary_phone_number      NUMBER(10) NOT NULL,
   secondary_phone_number   NUMBER(10),
   fname                           VARCHAR(25) NOT NULL,
   minit                           CHAR(1),
   lname                           VARCHAR(25) NOT NULL,
   email_id                        VARCHAR(25) NOT NULL,
   date_of_birth                   DATE NOT NULL,
   street                           VARCHAR(25) NOT NULL,
   city                             VARCHAR(25) NOT NULL,
   state                            VARCHAR(25) NOT NULL,
   zip_code                         NUMBER(5) NOT NULL,
   CONSTRAINT customer_pk PRIMARY KEY ( driver_license_number )
);
```

```
CREATE TABLE booking (
   booking_id                NUMBER(10) NOT NULL,
   start_mileage             NUMBER(6),
   end_mileage               NUMBER(6),
   booking_date              DATE DEFAULT SYSDATE NOT NULL,
   status                    VARCHAR(25) DEFAULT 'Booked' NOT NULL,
   booking_amount            NUMBER(7,2),
   drop_off_branch_id        NUMBER(6) NOT NULL,
   driver_license_number     VARCHAR(25) NOT NULL,
   start_date                DATE NOT NULL,
   end_date                  DATE NOT NULL,
   actual_end_date           DATE,
   CONSTRAINT booking_pk PRIMARY KEY ( booking_id )
);

CREATE TABLE promotion (
   promotion_code   VARCHAR(25) NOT NULL,
   discount_value   NUMBER(7,2) NOT NULL,
   discount_type    VARCHAR(10) NOT NULL,
   start_date       DATE NOT NULL,
   end_date         DATE NOT NULL,
   CONSTRAINT promotion_pk PRIMARY KEY ( promotion_code )
);

CREATE TABLE rental_insurance (
   insurance_code                VARCHAR(25) NOT NULL,
   bodily_injury_coverage        NUMBER(7,2) NOT NULL,
   property_damage_coverage      NUMBER(7,2) NOT NULL,
   rate_per_day                  NUMBER(7,2) NOT NULL,
   CONSTRAINT rental_insurance_pk PRIMARY KEY ( insurance_code )
);

CREATE TABLE card_information (
```

```
    driver_license_number   VARCHAR(25) NOT NULL,
    card_number             NUMBER(10) NOT NULL,
    card_type               VARCHAR(10) NOT NULL,
    fname                   VARCHAR(25) NOT NULL,
    minit                   CHAR(1),
    lname                   VARCHAR(25) NOT NULL,
    cvv                     NUMBER(3) NOT NULL,
    expiration_date         DATE NOT NULL,
    street                  VARCHAR(25) NOT NULL,
    city                    VARCHAR(25) NOT NULL,
    state                   VARCHAR(25) NOT NULL,
    zip_code                NUMBER(5) NOT NULL,
    CONSTRAINT card_information_pk PRIMARY KEY ( driver_license_number, card_number )
);
CREATE TABLE billing (
    bill_id                 NUMBER(10) NOT NULL,
    driver_license_number   VARCHAR(25),
    card_number             NUMBER(10),
    payment_mode            VARCHAR(25),
    status                  VARCHAR(25) DEFAULT 'Pending' NOT NULL,
    total_amount            NUMBER(7,2) NOT NULL,
    bill_date               DATE DEFAULT SYSDATE,
    CONSTRAINT billing_pk PRIMARY KEY ( bill_id )
);

CREATE TABLE branch (
    branch_id     NUMBER(6) NOT NULL,
    manager_id    NUMBER(10) NOT NULL,
    email_id      VARCHAR(25) NOT NULL,
    phone         NUMBER(10) NOT NULL,
    street        VARCHAR(25) NOT NULL,
    city          CHAR(25) NOT NULL,
    state         CHAR(25) NOT NULL,
    zip_code      NUMBER(5) NOT NULL,
```

```
    CONSTRAINT branch_pk PRIMARY KEY ( branch_id )
);

CREATE TABLE booking_car (
    registered_state         VARCHAR(15) NOT NULL,
    registration_number   VARCHAR(25) NOT NULL,
    booking_id                NUMBER(10) NOT NULL,
    CONSTRAINT booking_car_pk PRIMARY KEY ( registered_state, registration_number, booking_id )
);

CREATE TABLE booking_promotion (
    promotion_code   VARCHAR(25) NOT NULL,
    booking_id           NUMBER(10) NOT NULL,
    CONSTRAINT booking_promotion_pk PRIMARY KEY ( booking_id, promotion_code )
);

CREATE TABLE booking_rental_insurance (
    insurance_code   VARCHAR(25) NOT NULL,
    booking_id         NUMBER(10) NOT NULL,
    CONSTRAINT booking_rental_insurance_pk PRIMARY KEY ( booking_id, insurance_code )
);

CREATE TABLE sedan (
    registered_state         VARCHAR(15) NOT NULL,
    registration_number   VARCHAR(25) NOT NULL,
    exotic_flag                CHAR(1) DEFAULT 'N' NOT NULL,
    CONSTRAINT sedan_pk PRIMARY KEY ( registered_state, registration_number )
);

CREATE TABLE van (
    registered_state     VARCHAR(15) NOT NULL,
    registration_number   VARCHAR(25) NOT NULL,
    cargo_flag          CHAR(1) DEFAULT 'N' NOT NULL,
    CONSTRAINT van_pk PRIMARY KEY ( registered_state,
```

```
   registration_number )
);

CREATE TABLE suv (
   registered_state      VARCHAR(15) NOT NULL,
   registration_number    VARCHAR(25) NOT NULL,
   all_wheel_drive_flag   CHAR(1) DEFAULT 'N' NOT NULL,
   CONSTRAINT suv_pk PRIMARY KEY ( registered_state,
   registration_number )
);

CREATE TABLE truck (
   registered_state      VARCHAR(15) NOT NULL,
   registration_number   VARCHAR(25) NOT NULL,
   loading_capacity      INTEGER NOT NULL,
   CONSTRAINT truck_pk PRIMARY KEY ( registered_state,
   registration_number )
);

CREATE TABLE regular_rental_insurance (
   insurance_code   VARCHAR(25) NOT NULL,
   CONSTRAINT regular_rental_insurance_pk PRIMARY KEY ( insurance_code )
);

CREATE TABLE premium_rental_insurance (
   insurance_code               VARCHAR(25) NOT NULL,
   towing_coverage               NUMBER(7,2) NOT NULL,
   roadside_assistance_coverage   NUMBER(7,2) NOT NULL,
   CONSTRAINT premium_rental_insurance_pk PRIMARY KEY ( insurance_code )
);

CREATE TABLE membership_billing (
   member_driver_license_number   VARCHAR(25) NOT NULL,
   bill_id                    NUMBER(10) NOT NULL,
```

```
   CONSTRAINT membership_billing_pk PRIMARY KEY ( bill_id )
);

CREATE TABLE member (
   driver_license_number   VARCHAR(25) NOT NULL,
   login_id                VARCHAR(15) NOT NULL,
   password                VARCHAR(25) NOT NULL,
   membership_validity     DATE NOT NULL,
   CONSTRAINT member_pk PRIMARY KEY ( driver_license_number )
);

CREATE TABLE guest (
   driver_license_number   VARCHAR(25) NOT NULL,
   CONSTRAINT guest_pk PRIMARY KEY ( driver_license_number )
);

CREATE TABLE additional_driver (
   addl_driver_license_number   VARCHAR(25) NOT NULL,
   booking_id                   NUMBER(10) NOT NULL,
   fname                        VARCHAR(25) NOT NULL,
   minit                        CHAR(1),
   lname                        VARCHAR(25) NOT NULL,
   date_of_birth                DATE NOT NULL,
   CONSTRAINT additional_driver_pk PRIMARY KEY ( booking_id, addl_driver_license_number )
);

CREATE TABLE booking_billing (
   bill_id         NUMBER(10) NOT NULL,
   late_fee        NUMBER NOT NULL,
   booking_id      NUMBER(10) NOT NULL,
   CONSTRAINT booking_billing_pk PRIMARY KEY ( bill_id )
);
```

ALTER TABLE car ADD CONSTRAINT car_fk1 FOREIGN KEY (owning_branch_id) REFERENCES branch (branch_id) ON DELETE CASCADE;

ALTER TABLE booking ADD CONSTRAINT booking_fk1 FOREIGN KEY (drop_off_branch_id) REFERENCES branch (branch_id) ON DELETE CASCADE;

ALTER TABLE booking ADD CONSTRAINT booking_fk2 FOREIGN KEY (driver_license_number) REFERENCES customer (driver_license_number) ON DELETE CASCADE;

ALTER TABLE card_information ADD CONSTRAINT card_information_fk1 FOREIGN KEY (driver_license_number) REFERENCES Customer (driver_license_number) ON DELETE CASCADE;

ALTER TABLE billing ADD CONSTRAINT billing_fk1 FOREIGN KEY (driver_license_number, card_number) REFERENCES CARD_INFORMATION (driver_license_number,card_number) ON DELETE CASCADE;

ALTER TABLE booking_car ADD CONSTRAINT booking_car_fk1 FOREIGN KEY (registered_state, registration_number) REFERENCES car (registered_state, registration_number) ON DELETE CASCADE;

ALTER TABLE booking_car ADD CONSTRAINT booking_car_fk2 FOREIGN KEY (booking_id) REFERENCES Booking (booking_id) ON DELETE CASCADE;

ALTER TABLE booking_promotion ADD CONSTRAINT booking_promotion_fk1 FOREIGN KEY (promotion_code) REFERENCES promotion (promotion_code) ON DELETE CASCADE;

ALTER TABLE booking_promotion ADD CONSTRAINT booking_promotion_fk2 FOREIGN KEY (booking_id) REFERENCES booking (booking_id) ON DELETE CASCADE;

ALTER TABLE booking_rental_insurance ADD CONSTRAINT booking_rental_insurance_fk1 FOREIGN KEY (insurance_code) REFERENCES rental_insurance (insurance_code) ON DELETE CASCADE;

ALTER TABLE booking_rental_insurance ADD CONSTRAINT booking_rental_insurance_fk2 FOREIGN KEY (booking_id) REFERENCES booking (booking_id) ON DELETE CASCADE;

ALTER TABLE sedan ADD CONSTRAINT sedan_fk1 FOREIGN KEY (registered_state, registration_number) REFERENCES car (registered_state, registration_number) ON DELETE CASCADE;

ALTER TABLE van ADD CONSTRAINT van_fk1 FOREIGN KEY (registered_state, registration_number) REFERENCES car (registered_state, registration_number) ON DELETE CASCADE;

ALTER TABLE suv ADD CONSTRAINT suv_fk1 FOREIGN KEY (registered_state, registration_number) REFERENCES car (registered_state, registration_number) ON DELETE CASCADE;

ALTER TABLE truck ADD CONSTRAINT truck_fk1 FOREIGN KEY (registered_state, registration_number) REFERENCES car (registered_state, registration_number) ON DELETE CASCADE;

ALTER TABLE regular_rental_insurance ADD CONSTRAINT regular_rental_insurance_fk1 FOREIGN KEY (insurance_code) REFERENCES rental_insurance (insurance_code) ON DELETE CASCADE;

ALTER TABLE premium_rental_insurance ADD CONSTRAINT premium_rental_insurance_fk1 FOREIGN KEY (insurance_code) REFERENCES rental_insurance (insurance_code) ON DELETE CASCADE;

ALTER TABLE membership_billing ADD CONSTRAINT membership_billing_fk1 FOREIGN KEY (member_driver_license_number) REFERENCES Member (driver_license_number) ON DELETE CASCADE;

ALTER TABLE member ADD CONSTRAINT member_fk1 FOREIGN KEY (driver_license_number) REFERENCES customer(driver_license_number) ON DELETE CASCADE;

ALTER TABLE guest ADD CONSTRAINT guest_fk1 FOREIGN KEY (driver_license_number) REFERENCES customer(driver_license_number) ON DELETE CASCADE;

ALTER TABLE additional_driver ADD CONSTRAINT additional_driver_fk1 FOREIGN KEY (booking_id) REFERENCES booking(booking_id) ON DELETE CASCADE;

ALTER TABLE booking_billing ADD CONSTRAINT booking_billing_fk1 FOREIGN KEY (booking_id) REFERENCES booking(booking_id) ON DELETE CASCADE;

9. PL/SQL STATEMENTS

Procedure 1: CALCULATE_BOOKING_AMOUNT

Once a customer successfully completes a booking transaction, the Booking.Start_Date and Booking.End_Date will be used to calculate the Booking_Amount. Next, currently applicable promotions/discounts will be applied to this transaction. The booking amount will be further updated to include the rental insurance charge if the customer opts to purchase that service. Finally, an 8.25% blanket tax will be added to obtain the expected total charge, provided the customer will return the car on time.

```
create or replace PROCEDURE calculate_booking_amount (
   this_booking_id      IN booking.booking_id%TYPE,
   this_booking_amount   OUT booking.booking_amount%TYPE
) AS

   this_start_date                  booking.start_date%TYPE;
   this_end_date                    booking.end_date%TYPE;
   this_car_cost_per_day            car.cost_per_day%TYPE;
   this_discount_type               promotion.discount_type%TYPE;
   this_discount_value              promotion.discount_value%TYPE;
   this_insurance_rate_per_day      rental_insurance.rate_per_day%TYPE;
   this_booking_rent                booking.booking_amount%TYPE;
   this_discounted_booking_rent     booking.booking_amount%TYPE;
   this_booking_insurance_cost      booking.booking_amount%TYPE;
   this_booking_total_cost          booking.booking_amount%TYPE;
   this_total_insurance_rate_per_day RENTAL_INSURANCE.Rate_Per_Day%TYPE;
   this_total_insurance_rate_per_day:=0.0;

   CURSOR This_Insurance IS
         SELECT RENTAL_INSURANCE.Rate_Per_Day
         FROM RENTAL_INSURANCE, BOOKING_RENTAL_INSURANCE, BOOKING
         WHERE
         BOOKING_RENTAL_INSURANCE.Insurance_Code=RENTAL_INSURANCE.Insurance_Code
         AND
         BOOKING.Booking_ID=BOOKING_RENTAL_INSURANCE.Booking_ID
         AND
         BOOKING.Booking_ID=This_Booking_ID;
```

```
BEGIN
   SELECT bk.start_date, bk.end_date
   INTO this_start_date, this_end_date
   FROM booking bk
   WHERE bk.booking_id = this_booking_id;

   SELECT c.cost_per_day
   INTO this_car_cost_per_day
   FROM booking bk, booking_car bk_c, car c
   WHERE bk.booking_id = this_booking_id
   AND   bk.booking_id = bk_c.booking_id
   AND   bk_c.registered_state = c.registered_state
   AND   bk_c.registration_number = c.registration_number;

   BEGIN
      SELECT pr.discount_type, pr.discount_value
      INTO this_discount_type, this_discount_value
      FROM booking bk, booking_promotion bk_pr, promotion pr
      WHERE bk.booking_id = this_booking_id
      AND   bk.booking_id = bk_pr.booking_id
      AND   bk_pr.promotion_code = pr.promotion_code;

   EXCEPTION
      WHEN no_data_found THEN
         this_discount_type := 'Value';
         this_discount_value := 0;
   END;

   OPEN This_Insurance;
         LOOP
                  FETCH This_Insurance INTO this_insurance_rate_per_day;
                  EXIT WHEN This_Insurance%NOTFOUND;
                                    this_total_insurance_rate_per_day:=
                  this_total_insurance_rate_per_day+tThis_insurance_rate_per_day;
         END LOOP;

   CLOSE This_Insurance;
```

```
   this_booking_rent := ( TO_DATE(TO_CHAR(this_end_date) ) -
TO_DATE(TO_CHAR(this_start_date) ) ) * this_car_cost_per_day;

   IF this_discount_type = 'Percentage'
   THEN this_discounted_booking_rent := this_booking_rent - (this_booking_rent *
( this_discount_value / 100 ));
   ELSIF this_discount_type = 'Value'
   THEN this_discounted_booking_rent := this_booking_rent - this_discount_value;
   END IF;

   this_booking_insurance_cost := ( TO_DATE(TO_CHAR(this_end_date) ) -
TO_DATE(TO_CHAR(this_start_date) ) ) * this_total_insurance_rate_per_day;

   this_booking_total_cost := this_discounted_booking_rent + this_booking_insurance_cost;

   this_booking_amount := this_booking_total_cost * ( 1 + 0.0825 );
   dbms_output.put_line('Rent: ' || this_booking_rent);
   dbms_output.put_line('Rent after discount: ' || this_discounted_booking_rent);
   dbms_output.put_line('Insurance: ' ||this_booking_insurance_cost);
   dbms_output.put_line('Booking Cost: ' || this_booking_total_cost);
   dbms_output.put_line('Booking Cost with tax: ' || this_booking_amount);
END;
```

Output:

```
Connecting to the database CSOracle.
Rent: 100
Rent after discount: 90
Insurance: 0
Booking Cost: 90
Booking Cost with tax: 97.43
Process exited.
Disconnecting from the database CSOracle.
```

Procedure 2: CALCULATE_LATE_FEE

When a customer returns the rented car, a late fee may occur and be added to the final bill if the actual return date is later than the agreed return date. In addition, an 8.25% blanket tax will be added to obtain the final total later fee charge.

```
create or replace PROCEDURE calculate_late_fee (
   this_booking_id   IN booking.booking_id%TYPE,
   this_late_fee     OUT booking_billing.late_fee%TYPE
) AS

   this_end_date                 booking.end_date%TYPE;
   this_actual_end_date          booking.actual_end_date%TYPE;
   this_car_late_fee_per_day     car.late_fee_per_day%TYPE;
   this_late_fee_before_tax      booking_billing.late_fee%TYPE;

BEGIN
   SELECT bk.end_date, bk.actual_end_date
   INTO this_end_date, this_actual_end_date
   FROM booking bk
   WHERE bk.booking_id = this_booking_id;

   SELECT c.late_fee_per_day
   INTO this_car_late_fee_per_day
   FROM booking bk, booking_car bk_c, car c
   WHERE bk.booking_id = this_booking_id
   AND   bk.booking_id = bk_c.booking_id
   AND   bk_c.registered_state = c.registered_state
   AND   bk_c.registration_number = c.registration_number;

   IF ( this_actual_end_date > this_end_date )THEN
      this_late_fee_before_tax := ( TO_DATE(TO_CHAR(this_actual_end_date) ) -
TO_DATE(TO_CHAR(this_end_date) ) ) * this_car_late_fee_per_day;
   ELSE
      this_late_fee_before_tax := 0;
   END IF;

   this_late_fee := this_late_fee_before_tax * ( 1 + 0.0825 );
```

```
   dbms_output.put_line('Late Fee: ' || this_late_fee_before_tax);
   dbms_output.put_line('Late Fee with tax:  ' || this_late_fee);

END;
```

Output:

```
Connecting to the database CSOracle.
Late Fee: 110
Late Fee with tax:  119.075
Process exited.
Disconnecting from the database CSOracle.
```

Procedure 3: GENERATE_QUARTERLY_REPORT

This procedure generates a quarterly report. It generates information regarding the number of bookings and the total revenue generated for a particular zip code in the current quarter. This procedure takes a Date parameter as an input, which can be any date in a particular quarter and the report is generated accordingly. If no input is passed the current quarter report gets generated.

```
create or replace PROCEDURE generate_quarter_report (this_quarter_date IN DATE DEFAULT
SYSDATE) AS

   this_quarter_start_date     DATE := trunc(this_quarter_date,'Q');
   this_quarter_end_date       DATE := add_months(trunc(this_quarter_date,'Q'),3) - 1;
   this_zip_code               branch.zip_code%TYPE;
   this_total_revenue          NUMBER(15,2);
   this_total_booking_count    NUMBER(10);

   CURSOR quarterly_report_cur IS
   SELECT br.zip_code, COUNT(bk.booking_id), SUM(bl.total_amount)
   FROM branch br, car c, booking bk, booking_car bk_c, booking_billing bk_bl, billing bl
   WHERE     br.branch_id = c.owning_branch_id
      AND   bk_c.registration_number = c.registration_number
```

```
      AND   bk_c.registered_state = c.registered_state
      AND   bk_bl.bill_id = bl.bill_id
      AND   bk_bl.booking_id = bk.booking_id
      AND   bk_c.booking_id = bk.booking_id
      AND   bk.status = 'Completed'
      AND   bk.booking_date >= this_quarter_start_date
      AND   bk.booking_date <= this_quarter_end_date
   GROUP BY  br.zip_code;

BEGIN
   dbms_output.put_line(RPAD('Zip Code', 10) || RPAD('Booking Count', 15) || RPAD('Revenue', 17));
   OPEN quarterly_report_cur;
   LOOP
      FETCH quarterly_report_cur INTO this_zip_code,this_total_booking_count,this_total_revenue;
      EXIT WHEN quarterly_report_cur%notfound;
      dbms_output.put_line(RPAD(this_zip_code, 10) || RPAD(this_total_booking_count, 15) ||
RPAD(this_total_revenue, 17));
   END LOOP;

   CLOSE quarterly_report_cur;
END;
```

Output:

```
Connecting to the database CSOracle.
Zip Code  Booking Count  Revenue
75252     4              580
75254     4              740
Process exited.
Disconnecting from the database CSOracle.
```

Trigger 1: GENERATE_BILL

Once the customer returns the car, a bill will be generated which calculates the sum of the booking amount and the late fee, if applicable. A new record in BILLING is added with the total amount specified and status “Pending”. Similarly a new record in BOOKING_BILLING will be added which indiates late fee if any.

```
create or replace TRIGGER generate_bill AFTER
   UPDATE OF status ON booking
   FOR EACH ROW
DECLARE
   this_bill_id                 billing.bill_id%TYPE;
   this_booking_amount          booking.booking_amount%TYPE;
   this_late_fee                booking_billing.late_fee%TYPE;
   this_late_fee_before_tax     booking_billing.late_fee%TYPE;
   this_car_late_fee_per_day    car.late_fee_per_day%TYPE;
   this_total_amount            billing.total_amount%TYPE;
BEGIN
   IF :new.status = 'Completed'
   THEN
      SELECT MAX(bill_id)
      INTO this_bill_id
      FROM billing;

      this_bill_id := this_bill_id + 1;

      SELECT c.late_fee_per_day
      INTO this_car_late_fee_per_day
      FROM booking_car bk_c, car c
      WHERE bk_c.booking_id =:NEW.booking_id
      AND   bk_c.registered_state = c.registered_state
      AND   bk_c.registration_number = c.registration_number;

      this_booking_amount :=:NEW.booking_amount;

      IF (:new.actual_end_date > :NEW.end_date )
      THEN
         this_late_fee_before_tax := ( TO_DATE(TO_CHAR(:NEW.actual_end_date) ) -
TO_DATE(TO_CHAR(:NEW.end_date) ) ) * this_car_late_fee_per_day;
```

```
    ELSE
      this_late_fee_before_tax := 0;
    END IF;

    this_late_fee := this_late_fee_before_tax * ( 1 + 0.0825 );
    this_total_amount := this_booking_amount + this_late_fee;

    INSERT INTO billing (bill_id, total_amount)
      VALUES (this_bill_id, this_total_amount);

    INSERT INTO booking_billing (bill_id, late_fee, booking_id)
      VALUES (this_bill_id, this_late_fee, :NEW.booking_id);

  END IF;
END;
```

Trigger 2: UPDATE_CAR_AVAILIBILITY

When a car is returned by a customer, we update the status of its availability for booking as “A”. Similarly, when the booking status is changed from a “Booked” state to “Active” state, indicating the customer has taken the car, we update the status of the car to “N” indicating not available for booking at this point.

```
create or replace TRIGGER update_car_availabilty AFTER
  UPDATE OF status ON booking
  FOR EACH ROW

DECLARE
  this_registered_state        car.registered_state%TYPE;
  this_registration_number   car.registration_number%TYPE;
  this_owning_branch_id      car.owning_branch_id%TYPE;

BEGIN
  SELECT registered_state, registration_number
  INTO this_registered_state,this_registration_number
  FROM booking_car bk_c
  WHERE bk_c.booking_id =:new.booking_id;
```

```
    SELECT owning_branch_id
    INTO this_owning_branch_id
    FROM car
    WHERE registered_state = this_registered_state
    AND   registration_number = this_registration_number;

    IF :new.status = 'Completed' THEN
       IF :new.drop_off_branch_id = this_owning_branch_id THEN
          UPDATE car
          SET availability_flag = 'A', mileage =:new.end_mileage
          WHERE registered_state = this_registered_state
          AND   registration_number = this_registration_number;
       ELSE
          UPDATE car
          SET availability_flag = 'N', mileage =:new.end_mileage
          WHERE registered_state = this_registered_state
          AND   registration_number = this_registration_number;
       END IF;
    ELSIF :new.status = 'Active' THEN
       UPDATE car
       SET availability_flag = 'N'
       WHERE registered_state = this_registered_state
       AND   registration_number = this_registration_number;
    END IF;
END;
```