1. Data Requirements of the System

Airbnb is an online marketplace for holiday rentals and tourism activities, focusing on homestays. Customers and Hosts are the two main sorts of users who use the platform. Hosts posts their services on the site, and passengers can book the host's services.

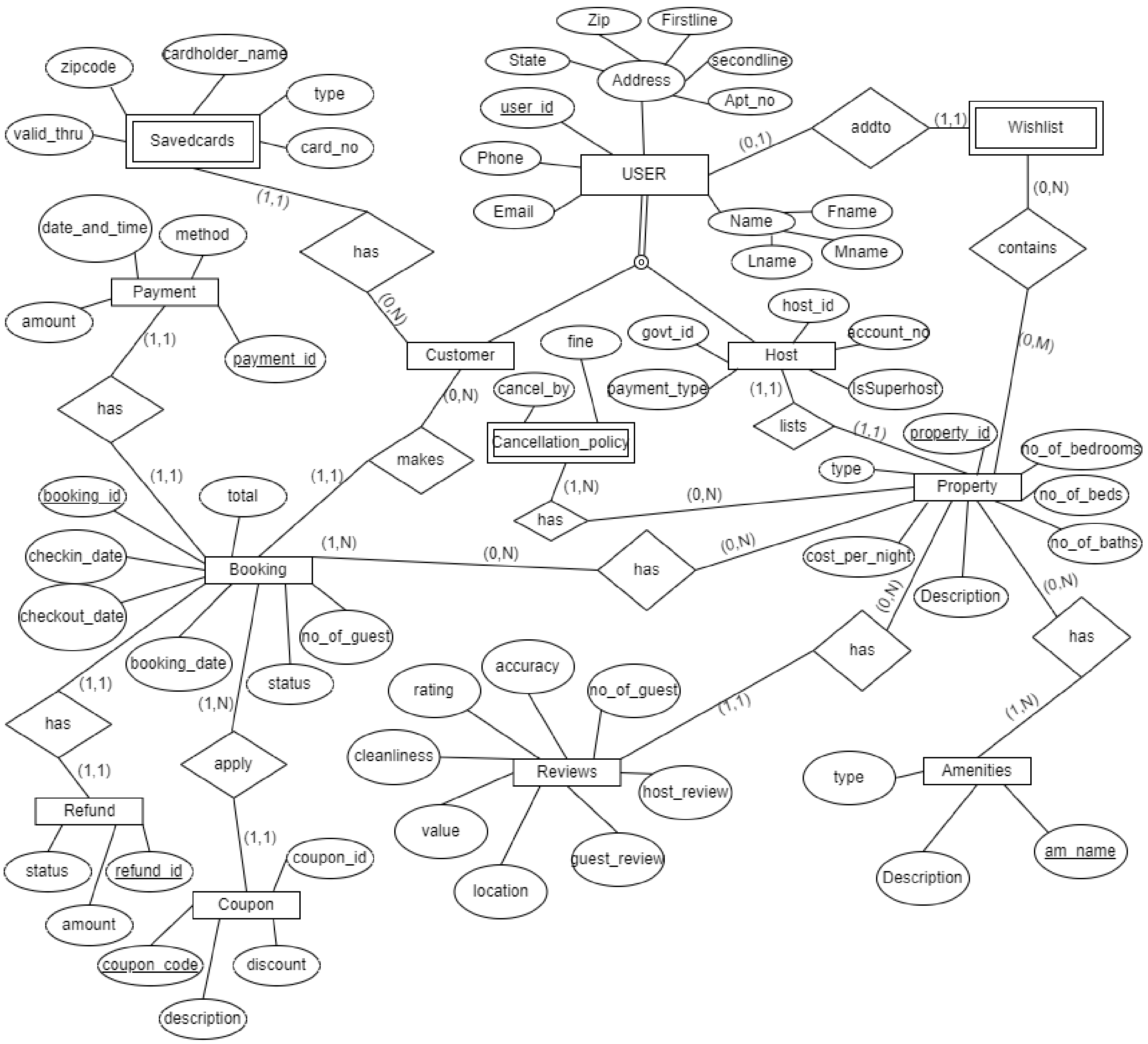
When a user decides to become a host, they must provide a legitimate government-issued ID and a bank account number. A host can advertise properties once they've registered. Details such as the type of property, cost per night, number of rooms, maximum number of guests allowed, facilities, location, and images are collected and saved in the system if it is a property. The host's cancellation policy is specified under the property information.

A customer can create a wish list of properties and experiences that they want to visit. When customers decide to book a property, availability is checked using existing booking data, and the booking is confirmed once payment is received. Payments can be made using stored cards if they are available. To create the invoice, information such as the total amount, payment ID, promotions applied, and timestamp are captured. The status of the reservation can also be changed: Confirmed (after payment), Completed (after a successful stay), and Cancelled (if the user decides to cancel the booking). If a booking is canceled, refunds are tracked separately.

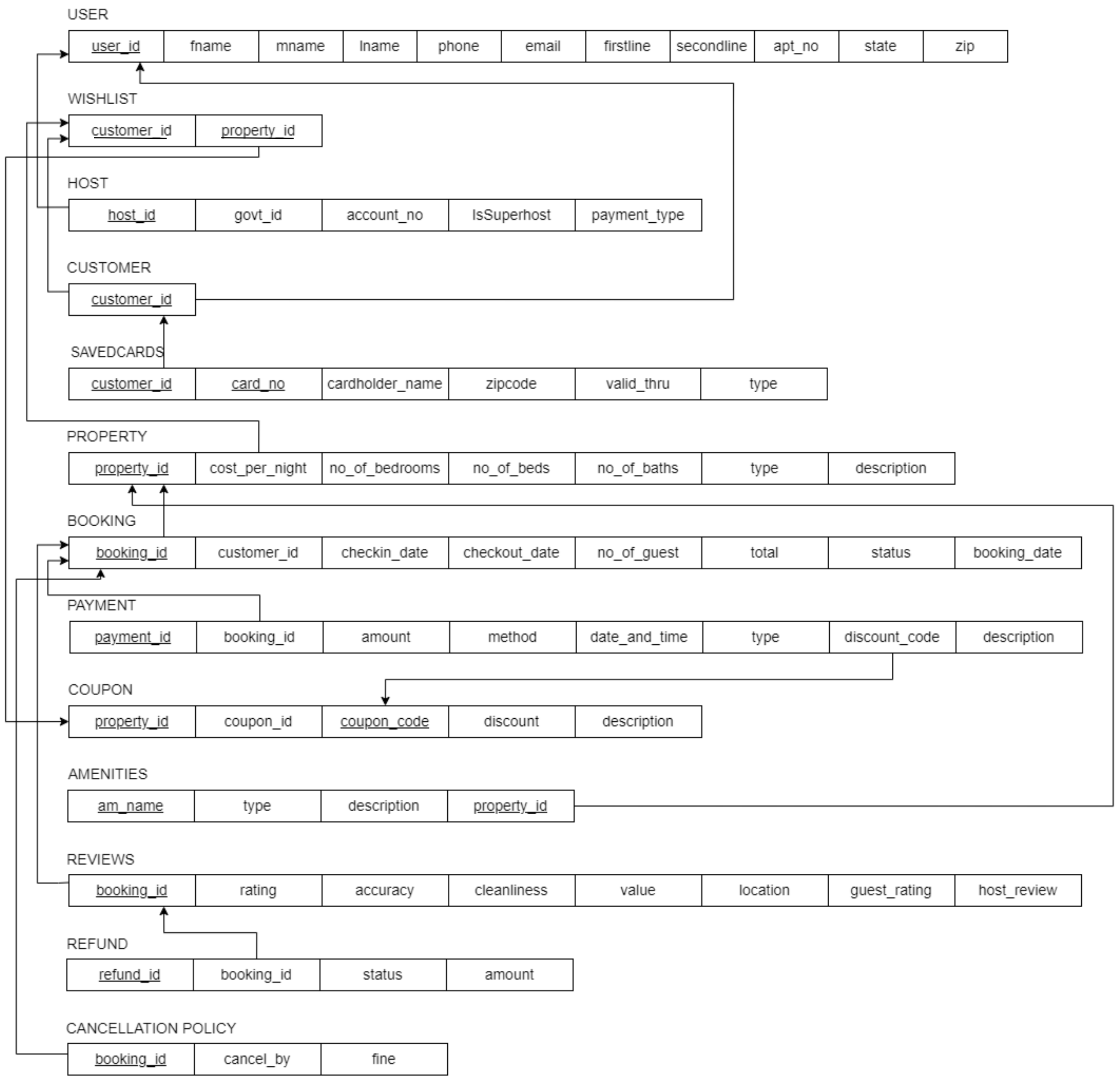
Once the tourists have completed their stay, their feedback is recorded using a five-star rating system. For property, the feedback is on things like location, cleanliness, accuracy, check-in experience, and host reaction. All types of bookings receive an overall rating as well as a written review. A host might become a super host based on the number of positive responses received. Airbnb uses a two-way review system, so the hosts can also provide feedback on their visitors.

A host can choose to run a promotion for his or her property, offering discounts using coupon codes. Bookings eligible for a discount can utilize a valid coupon code to get a discount on the total price. The money is transferred to the host's bank account following a successful stay. This is the general working style of Airbnb homestays and services.

2. Initial ER Diagram of the System

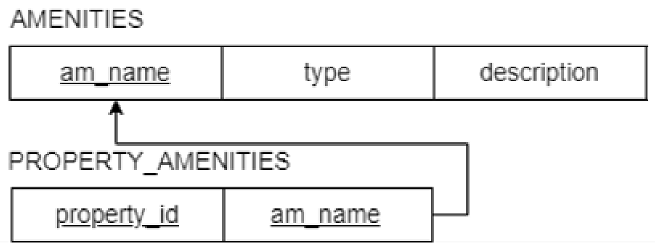


3. Mapping ER Diagram into Relational Schema



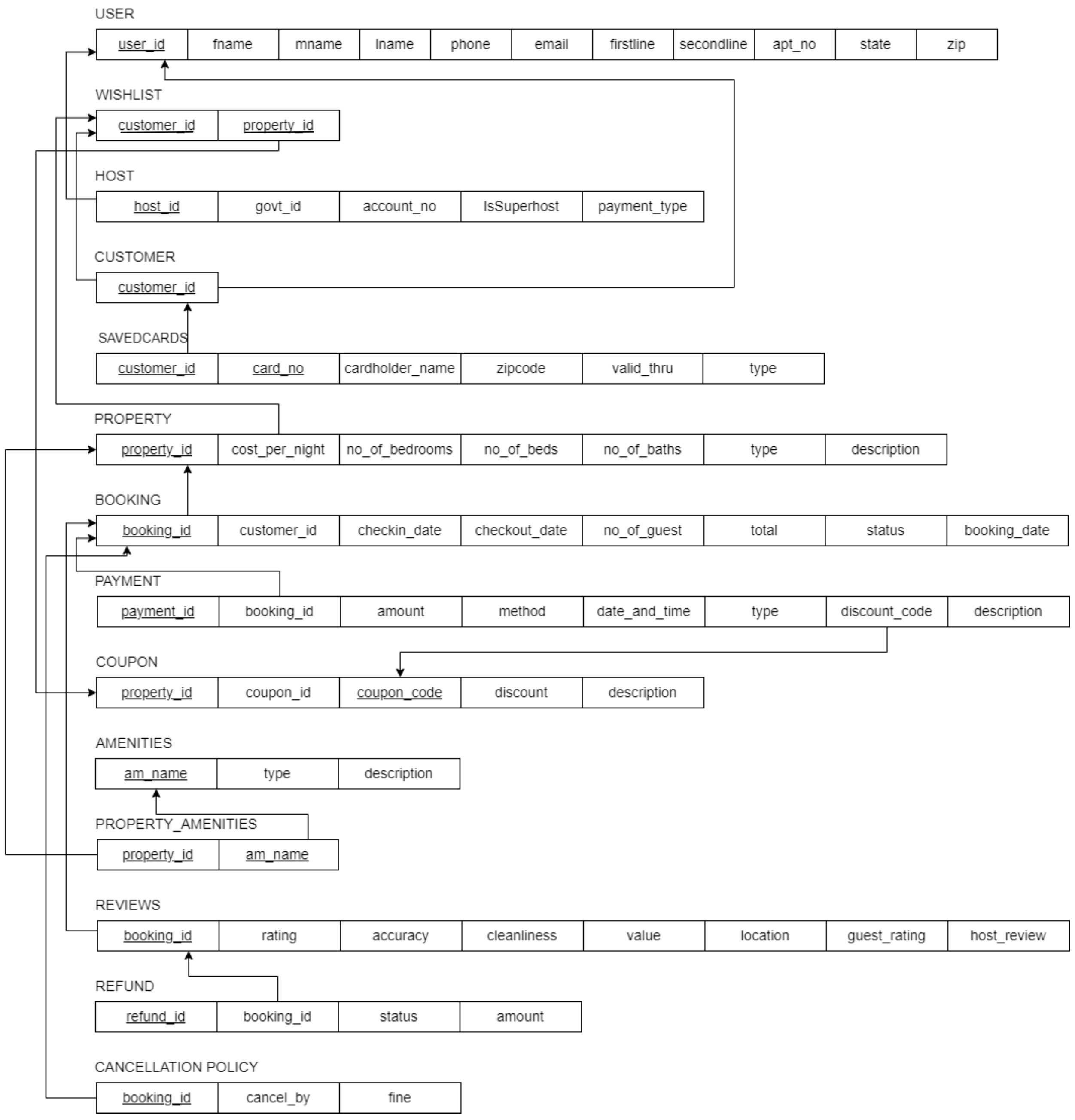
4. Database Normalization of the System

The amenity's relation violates 2NF because of partial dependency on the primary key. So, this relation is split into two. After normalization,



Other than the amenity, all of the other entities are already in normalized form.

5. Final Relational Schema of the System


USER
user_id | fname | mname | lname | phone | email | firstline | secondline | apt_no | state | zip
WISHLIST
customer_id | property_id
HOST
host_id | govt_id | account_no | IsSuperhost | payment_type
CUSTOMER
customer_id
SAVEDCARDS
customer_id | card_no | cardholder_name | zipcode | valid_thru | type
PROPERTY
property_id | cost_per_night | no_of_bedrooms | no_of_beds | no_of_baths | type | description
BOOKING
booking_id | customer_id | checkin_date | checkout_date | no_of_guest | total | status | booking_date
PAYMENT
payment_id | booking_id | amount | method | date_and_time | type | discount_code | description
COUPON
property_id | coupon_id | coupon_code | discount | description
AMENITIES
am_name | type | description
PROPERTY_AMENITIES
property_id | am_name
REVIEWS
booking_id | rating | accuracy | cleanliness | value | location | guest_rating | host_review
REFUND
refund_id | booking_id | status | amount
CANCELLATION POLICY
booking_id | cancel_by | fine

6. Create Tables using SQL Commands

```
CREATE TABLE USERS(
USER_ID VARCHAR(50),
FNAME VARCHAR(30) NOT NULL,
MNAME VARCHAR(30),
LNAME VARCHAR(30),
PHONE INTEGER,
EMAIL VARCHAR(30) NOT NULL,
FIRSTLINE VARCHAR(50),
SECONDINE VARCHAR(50),
APT_NO INTEGER,
STATE VARCHAR(30),
ZIP INTEGER,
PRIMARY KEY (USER_ID),
CONSTRAINT PHONE_NO_CONST CHECK (LENGTH (PHONE) = 10)
);
CREATE TABLE WISHLIST(
 CUSTOMER_ID VARCHAR(50),
 PROPERTY_ID VARCHAR(50),
 PRIMARY KEY(CUSTOMER_ID,PROPERTY_ID)
);
CREATE TABLE HOST(
 HOST_ID VARCHAR(50),
 GOVT_ID VARCHAR(30) NOT NULL,
 ACCOUNT_NO VARCHAR(15),
 ISSUPERHOST NUMBER(1) NOT NULL,
 PAYMENT_TYPE ENUM['CREDIT','DEBIT','PAYPAL']
 PRIMARY KEY(HOST_ID)
);
CREATE TABLE CUSTOMER(
 CUSTOMER_ID VARCHAR(50),
 PRIMARY KEY(CUSTOMER_ID)
);
CREATE TABLE SAVEDCARDS(
 CUSTOMER_ID VARCHAR(50),
 CARD_NO INTEGER,
 CARDHOLDER_NAME VARCHAR(30),
 VALID_THRU DATE,
 ZIP_CODE INT,
 TYPE VARCHAR(30) NOT NULL,
 CONSTRAINT CARD_NO_CONST CHECK (LENGTH (CARD_NO) = 10),
 PRIMARY KEY(CUSTOMER_ID, CARD_NO)
);
CREATE TABLE PROPERTY(
 PROPERTY_ID VARCHAR(50),
 PRICE_PER_NIGHT INTEGER NOT NULL,
 NO_OF_BEDROOMS INTEGER,
 NO_OF_BEDS INTEGER,
 NO_OF_BATHS INTEGER,
 TYPE VARCHAR(20),
 DESCRIPTION VARCHAR(100),
 PRIMARY KEY(PROPERTY_ID)
);
CREATE TABLE BOOKING(
 BOOKING_ID VARCHAR(50),
 CUSTOMER_ID VARCHAR(50) NOT NULL,
 CHECKIN_DATE DATE NOT NULL,
 CHECKOUT_DATE DATE,
 NO_OF_GUEST INTEGER,
 TOTAL INTEGER NOT NULL,
 STATUS VARCHAR(30) NOT NULL,
 BOOKING_DATE DATE DEFAULT CURRENT_DATE,
 PRIMARY KEY (BOOKING_ID)
);
CREATE TABLE PAYMENT(
 PAYMENT_ID VARCHAR(50),
 BOOKING_ID VARCHAR(50) NOT NULL,
 AMOUNT INTEGER NOT NULL,
 METHOD VARCHAR(50) NOT NULL,
 DATE_AND_TIME TIMESTAMP DEFAULT CURRENT_DATE,
 TYPE VARCHAR(30) NOT NULL,
 DISCOUNT_CODE INTEGER,
 DESCRIPTION VARCHAR(100),
 PRIMARY KEY (PAYMENT_ID)
);
CREATE TABLE COUPON(
 PROPERTY_ID VARCHAR(30),
 COUPON_ID VARCHAR(30),
 DISCOUNT_CODE INTEGER,
 DISCOUNT VARCHAR(30) NOT NULL,
 DESCRIPTION VARCHAR(100),
 PRIMARY KEY (PROPERTY_ID, COUPON_CODE)
);
CREATE TABLE AMENITIES(
 AM_NAME VARCHAR(50),
 DESCRIPTION VARCHAR(100),
 TYPE VARCHAR(50),
 PRIMARY KEY (AM_NAME)
);
CREATE TABLE PROPERTY_AMENITIES(
 AM_NAME VARCHAR(50),
 PROPERTY_ID VARCHAR(50),
 PRIMARY KEY (AM_NAME, PROPERTY_ID)
);
CREATE TABLE REVIEWS(
 BOOKING_ID VARCHAR(50),
 RATING INTEGER,
 ACCURACY INTEGER,
 CLEANLINESS INTEGER,
 VALUE INTEGER,
 LOCATION INTEGER,
 GUEST_RATING INTEGER,
 HOST_REVIEW VARCHAR(100)
 PRIMARY KEY (BOOKING_ID),
 CONSTRAINT RATING_CONST CHECK (OVERALL_RATING BETWEEN 1 AND 5),
 CONSTRAINT CLEANLINESS_CONST CHECK (CLEANLINESS BETWEEN 1 AND 5),
 CONSTRAINT ACCURACY_CONST CHECK (ACCURACY BETWEEN 1 AND 5),
 CONSTRAINT VALUE_CONST CHECK (VALUE BETWEEN 1 AND 5),
 CONSTRAINT LOCATION_CONST CHECK (LOCATION_RATING BETWEEN 1 AND 5),
 CONSTRAINT GUEST_RATING_CONST CHECK (GUEST_RATING BETWEEN 1 AND 5)
);
CREATE TABLE REFUND(
 REFUND_ID NUMBER GENERATED ALWAYS AS IDENTITY,
 BOOKING_ID VARCHAR(50) NOT NULL,
 AMOUNT INTEGER NOT NULL,
 STATUS VARCHAR(30) NOT NULL,
 PRIMARY KEY (REFUND_ID)
);
CREATE TABLE CANCELLATION_POLICY(
 BOOKING_ID VARCHAR(50),
 CANCEL_BY INTEGER NOT NULL,
 FINE INTEGER NOT NULL,
 PRIMARY KEY(BOOKING_ID)
);
ALTER TABLE CUSTOMER ADD CONSTRAINT CUSTOMER_FKEY FOREIGN KEY(CUSTOMER_ID)
REFERENCES USERS(USER_ID);
ALTER TABLE WISHLIST ADD CONSTRAINT WISH_USER_FKEY FOREIGN KEY(CUSTOMER_ID)
REFERENCES CUSTOMER(CUSTOMER_ID) ON DELETE CASCADE;
ALTER TABLE WISHLIST ADD CONSTRAINT WISH_PROPERTY_FKEY FOREIGN KEY(BOOKING_ID)
REFERENCES BOOKING(BOOKING_ID) ;
ALTER TABLE HOST ADD CONSTRAINT HOST_FKEY FOREIGN KEY(HOST_ID) REFERENCES
USERS(USER_ID);
ALTER TABLE PROPERTY ADD CONSTRAINT PROPERTY_FKEY FOREIGN KEY(PROPERTY_ID)
REFERENCES
BOOKING(BOOKING_ID);
ALTER TABLE CANCELLATION_POLICY ADD CONSTRAINT CANCEL_FKEY FOREIGN
KEY(BOOKING_ID)
REFERENCES BOOKING(BOOKING_ID);
```

```
ALTER TABLE PROPERTY_AMENITIES ADD CONSTRAINT AMENITY_FKEY FOREIGN
KEY(PROPERTY_ID)
REFERENCES PROPERTY(PROPERTY_ID);
ALTER TABLE PROPERTY_AMENITIES ADD CONSTRAINT AMNAME_FKEY FOREIGN
KEY(AM_NAME) REFERENCES AMENITIES(AM_NAME);
ALTER TABLE BOOKINGS ADD CONSTRAINT CUSTOMER_BOOK_FKEY FOREIGN KEY(CUSTOMER_ID)
REFERENCES CUSTOMER(CUSTOMER_ID);
ALTER TABLE REVIEWS ADD CONSTRAINT BOOKING_FKEY FOREIGN KEY(BOOKING_ID)
REFERENCES
BOOKINGS(BOOKING_ID) ON DELETE CASCADE;
ALTER TABLE PAYMENT ADD CONSTRAINT BOOKING_PAYMENT_FKEY FOREIGN KEY(BOOKING_ID)
REFERENCES BOOKINGS(BOOKING_ID) ON DELETE CASCADE;
ALTER TABLE REFUND ADD CONSTRAINT BOOKING_REFUND_FKEY FOREIGN KEY(BOOKING_ID)
REFERENCES BOOKINGS(BOOKING_ID) ON DELETE CASCADE;
ALTER TABLE PAYMENT ADD CONSTRAINT COUPON_FKEY FOREIGN KEY(COUPON_CODE)
REFERENCES
COUPON(COUPON_CODE) ON DELETE SET NULL;
ALTER TABLE SAVEDCARDS ADD CONSTRAINT CARDHOLDER_NAME_FK FOREIGN KEY(CUSTOMER_ID)
REFERENCES
CUSTOMER(CUSTOMER_ID) ON DELETE CASCADE;
ALTER TABLE BOOKINGS ADD CONSTRAINT CHECK_OUT_CONSTRAINT
CHECK (
 (CHECK_OUT_DATE > CHECK_IN_DATE)
);
ALTER TABLE USERS ADD CONSTRAINT EMAIL_CONSTRAINT
CHECK (
 EMAIL LIKE '%@%.%'
);
```

7. PL/SQL

Stored Procedures:

a) Adding Payment Information:

```
CREATE PROCEDURE ADD_CARD_INFO (
    CARD_NO     IN VARCHAR,
    USER_ID     IN INT,
    VALID_THRU  IN DATE,
) AS
BEGIN
    INSERT INTO CARD_INFO VALUES(CARD_NO, USER_ID, VALID_THRU,0);
END ADD_CARD_INFO;
```

b) Setting Default Payment Information:

```
CREATE PROCEDURE SET_DEFAULT_CARD (
    CARD_NO_VAR IN INT,
    USER_ID_VAR IN INT,
    ) AS
BEGIN
    UPDATE CARD_INFO SET
    IS_DEFAULT = 1
    WHERE USER_ID = USER_ID_VAR AND CARD_NO = CARD_ID_VAR;
END SET_DEFAULT_CARD_INFO;
```

Triggers:

a) Booking Confirmation:

```
CREATE TRIGGER BOOKING_STATUS
AFTER INSERT ON PAYMENT_DETAILS
FOR EACH ROW
UPDATE BOOKING SET BOOKING_STATUS = 'BOOKED'
WHERE BOOKING_ID = NEW.BOOKING_ID
```

b) Deleting a Property:

Assumption: If a property is booked by a customer and then it is deleted by the host or it's no longer available for stay. Then the refund is initiated for the customer.

```
CREATE TRIGGER DELETE_PROPERTY
AFTER UPDATE OF IS_THERE ON PROPERTY
FOR EACH ROW
BEGIN
    IF(:NEW.IS_THERE = 0)
    THEN
    INSERT INTO REFUND (BOOKING_ID, AMOUNT, STATUS)
    SELECT B.BOOKING_ID, P.AMOUNT, 'REFUND INITIATED'
    FROM BOOKINGS B JOIN
    PAYMENT P ON P.BOOKING_ID = B.BOOKING_ID
    WHERE B.STATUS = 'CONFIRMED' AND B.PROPERTY_ID = :NEW.PROPERTY_ID;
    UPDATE BOOKINGS SET STATUS = 'CANCELLED'
    WHERE PROPERTY_ID = :NEW.PROPERTY_ID
    AND STATUS = 'CONFIRMED';
    END IF;
END;
```