Fall 2021

Database Design

- CS 6360.001
- Project: AirBnB-2
- Team No: 27
- Instructor: Nurcan Yuruk

Members

- Shashank Saran (SXS210186)
- Abirami Dhayalan (AXD210002)
- Aman Raj Suman (ARS200002)

Contents

Figure No.	Description	Page No.
Fig 1	Entity Relationship Diagram	4
Fig 2	Relational Schema Diagram	5
Fig 3	Functional Dependency Diagram	6, 7
Fig 4	Normalization Diagram	8

Requirements

AirBnB’s primary service is to provide a platform which connects guests and hosts for the purpose of booking vacation stays around the world. The system will include the following functionality:

1. The system will store some basic information about the user such as Name, Email, Phone and Date of Birth. Email uniquely identifies a customer. Only users above the age of 18 can register.
2. Each user will have a single account associated with them. Each account has a unique username along with additional details such as the password, profile photo URL and date the account was created. An account could be a guest account, a host account or both.
3. The system maintains details about properties owned by hosts. Each property has the following details: a unique property ID, the type of property (shared, private, entire place, hotel), availability (booked, available, unavailable), price, the location, cancelation policy, number of rooms, beds and bathrooms, photo URL and description.
4. Hosts have some additional details stored for them like, number of years hosting, if they are super hosts or not, and a short bio about themselves.
5. A location is comprised of the street address, city, zip code and country.
6. Guests should be able to book available places, offered by hosts, by providing the dates they would like to stay for, the number of guests and the number of rooms.
7. The System also allows a guest to make a payment for their booking. Each booking allows for a single payment and requires that some bank information is provided for both the host and guest along with the amount being transferred and the payment method.
8. Guests can leave reviews about their stay. Hosts can leave reviews about guests as well. A review consists of a title, a short comment, a rating, the username of the reviewer and when the review was left.
9. In addition to booking rooms, guests can also search for events around a location. Each event has a title, address, category, time, distance from the location and price.

Enhanced Entity-Relation Diagram

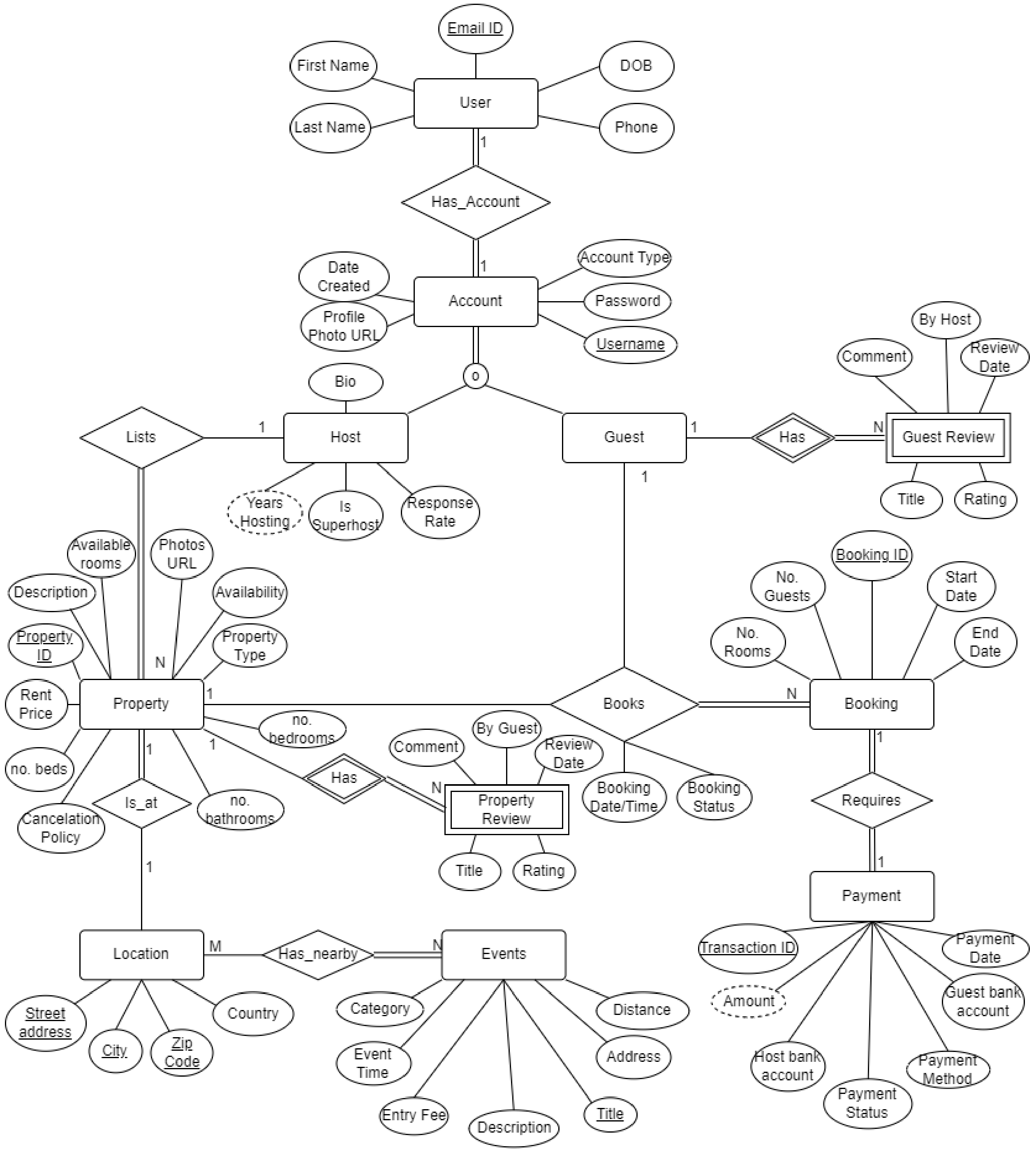


***Fig1:** Entity Relationship Diagram*

Mapping to relational model and normalization

Relational mode

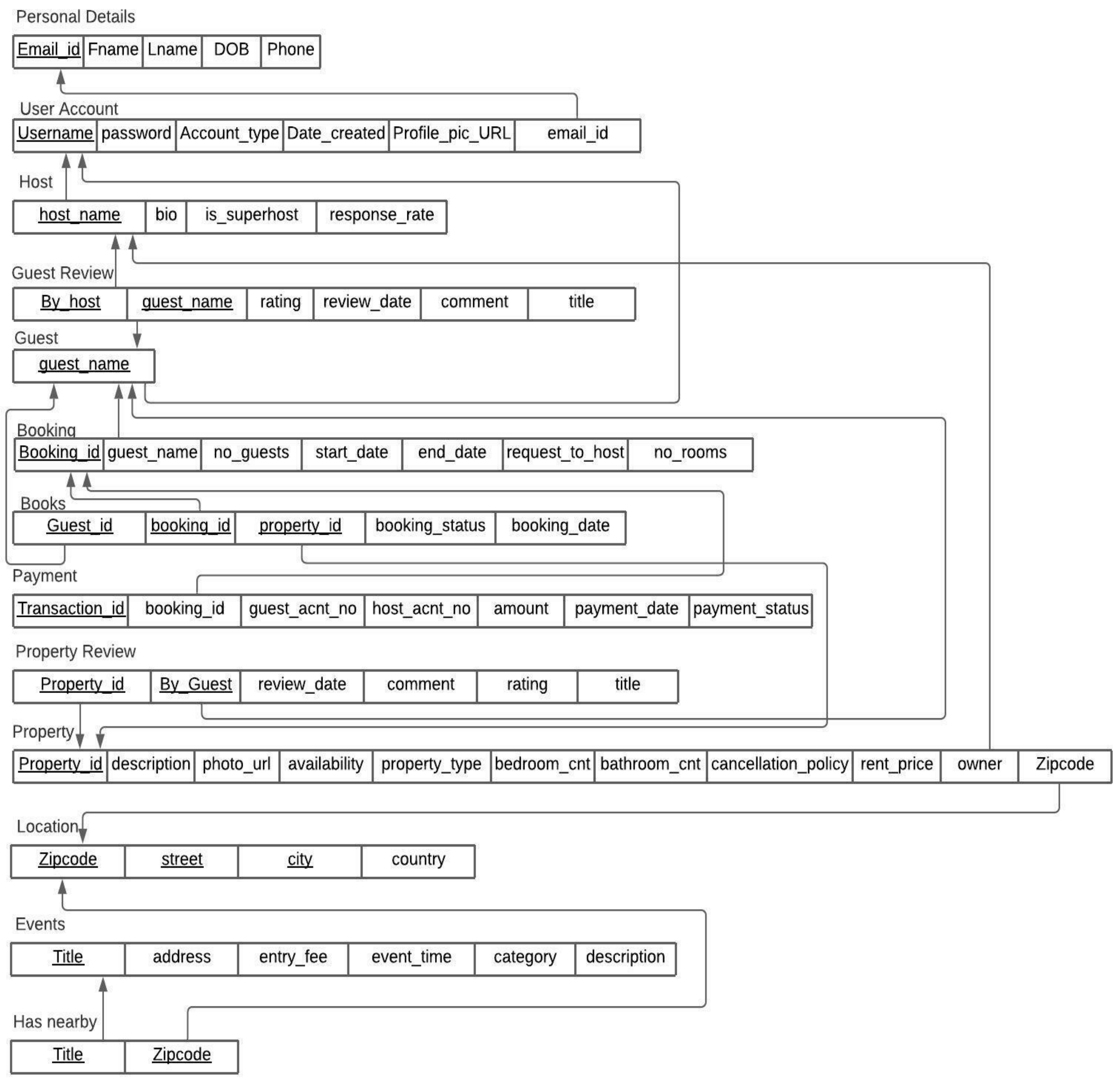
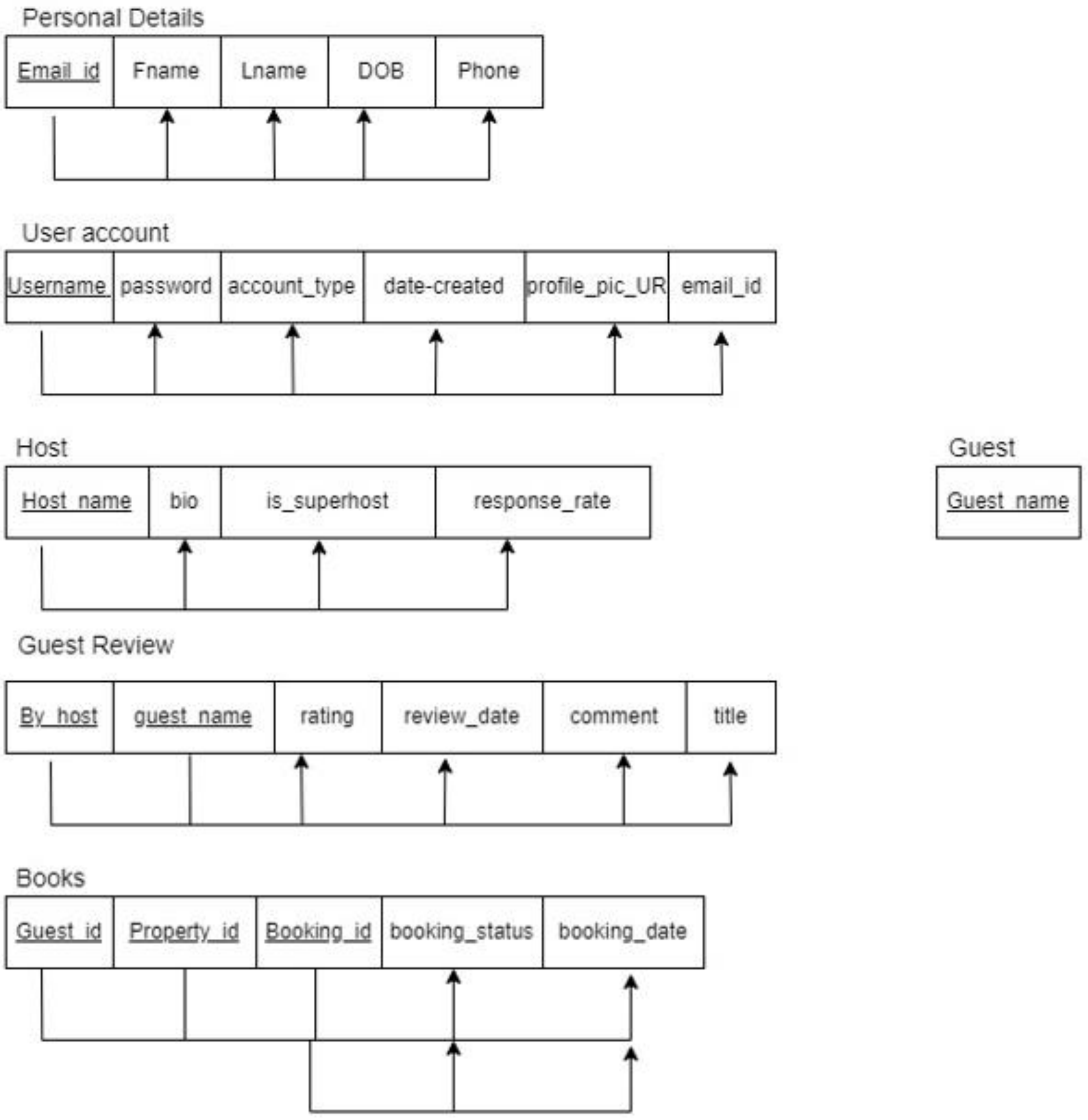

Personal Details
Email_id | Fname | Lname | DOB | Phone
User Account
Username | password | Account_type | Date_created | Profile_pic_URL | email_id
Host
host_name | bio | is_superhost | response_rate
Guest Review
By_host | guest_name | rating | review_date | comment | title
Guest
guest_name
Booking
Booking_id | guest_name | no_guests | start_date | end_date | request_to_host | no_rooms
Books
Guest_id | booking_id | property_id | booking_status | booking_date
Payment
Transaction_id | booking_id | guest_acnt_no | host_acnt_no | amount | payment_date | payment_status
Property Review
Property_id | By_Guest | review_date | comment | rating | title
Property
Property_id | description | photo_url | availability | property_type | bedroom_cnt | bathroom_cnt | cancellation_policy | rent_price | owner | Zipcode
Location
Zipcode | street | city | country
Events
Title | address | entry_fee | event_time | category | description
Has nearby
Title | Zipcode

Fig2: *Relational Schema Diagram*

Functional Dependency



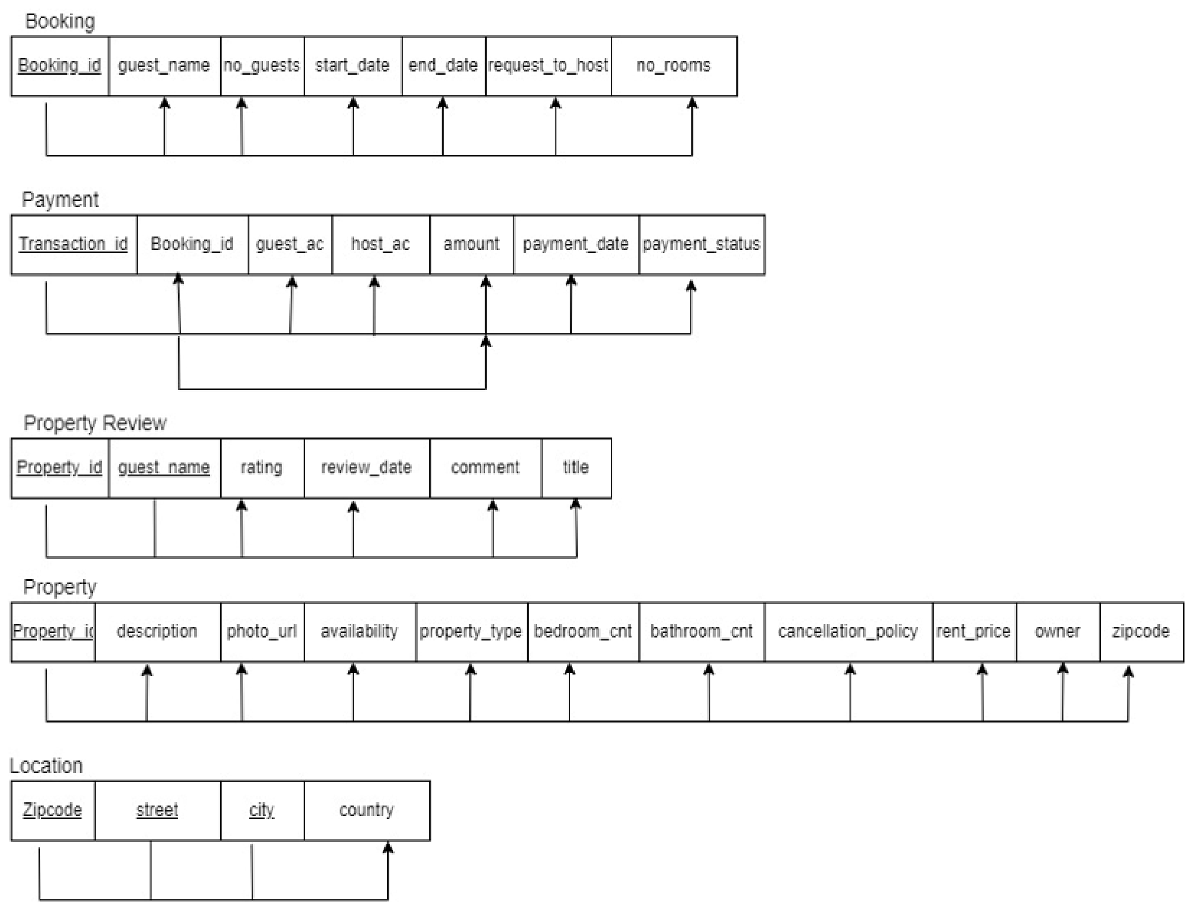


Fig3: *Functional Dependency Diagram*

Normalization

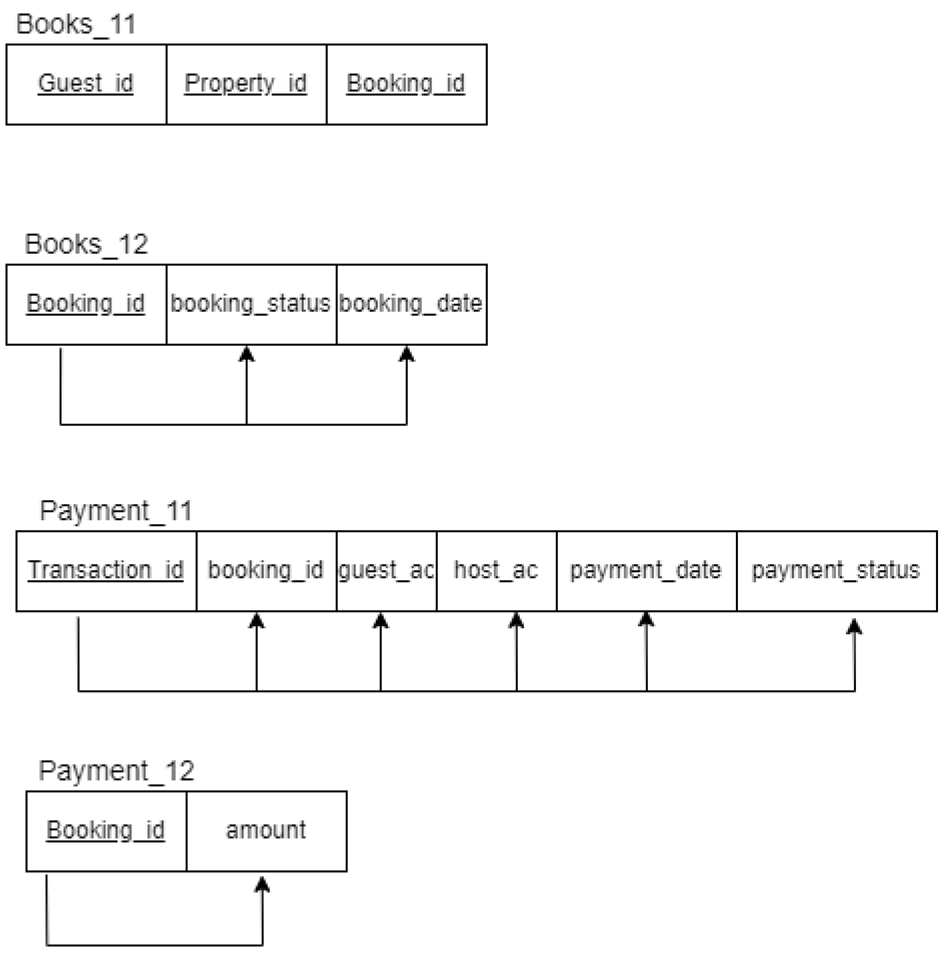


Fig4: *Normalization Diagram*

SQL

```
CREATE TABLE PersonalDetails (
	Email_id VARCHAR(20),
	Fname VARCHAR(20) NOT NULL,
	Lname VARCHAR(20) NOT NULL,
	DOB datetime,
	Phone int(10),
	PRIMARY KEY (Email_id)
);

CREATE TABLE UserAccount (
	Username VARCHAR (15) NOT NULL ,
	password VARCHAR (25) NOT NULL ,
	Account_type VARCHAR(20),
	Date_created datetime,
	Profile_pic_URL VARCHAR(50),
	email_id VARCHAR(20),
	PRIMARY KEY (Username),
	FOREIGN KEY (email_id) REFERENCES PersonalDetails(Email_id) ON DELETE CASCADE
);

CREATE TABLE HostDetails(
	hostname VARCHAR (25) ,
	bio VARCHAR (20),
	is_superhost BOOLEAN,
	response_rate float(10),
	PRIMARY KEY (hostname),
	FOREIGN KEY (hostname) REFERENCES UserAccount(Username) ON DELETE CASCADE
);

CREATE TABLE GuestReview(
	By_host VARCHAR (20),
	guest_name VARCHAR (20),
	rating int(10),
	review_date datetime,
	review_comments VARCHAR (20),
	title VARCHAR (20),
	PRIMARY KEY (By_host,guest_name),
	FOREIGN KEY (By_host) REFERENCES HostDetails(hostname) ON DELETE CASCADE,
	FOREIGN KEY (guest_name) REFERENCES Guest(guest_name) ON DELETE CASCADE
);

CREATE TABLE Guest(
	guest_name VARCHAR (20),
	PRIMARY KEY (guest_name),
	FOREIGN KEY (guest_name) REFERENCES UserAccount(Username) ON DELETE CASCADE
);
```

```sql
CREATE TABLE Booking(
		Booking_id VARCHAR (10),
		Guest_name VARCHAR (20),
		no_guests int(10),
		start_date datetime,
		end_date datetime,
		request_to_host VARCHAR (20),
		no_rooms int(10),
		PRIMARY KEY (Booking_id),
		FOREIGN KEY (Guest_name) REFERENCES Guest(guest_name) ON DELETE CASCADE
);

CREATE TABLE Books_11(
		Guest_id VARCHAR (10),
		booking_id VARCHAR (10),
		property_id VARCHAR (10),
		PRIMARY KEY (Guest_id,booking_id,property_id),
		FOREIGN KEY (Guest_id) REFERENCES Guest(guest_name) ON DELETE CASCADE,
		FOREIGN KEY (booking_id) REFERENCES Booking(Booking_id) ON DELETE CASCADE,
		FOREIGN KEY (property_id) REFERENCES Property(Property_id) ON DELETE CASCADE
);

CREATE TABLE Books_12(
		booking_id VARCHAR (10),
		booking_status VARCHAR (10) DEFAULT 'Awaiting Confirmation',
		booking_date datetime,
		PRIMARY KEY (booking_id),
		FOREIGN KEY (booking_id) REFERENCES Booking(Booking_id) ON DELETE CASCADE
);

CREATE TABLE Payment_11(
		Transaction_id VARCHAR (10),
		booking_id VARCHAR (10) NOT NULL,
		guest_acnt_no int(10),
		host_acnt_no int(10),
		payment_date datetime,
		payment_status VARCHAR (10) DEFAULT 'Pending',
		PRIMARY KEY (Transaction_id),
		FOREIGN KEY (booking_id) REFERENCES Booking(Booking_id) ON DELETE CASCADE
);

CREATE TABLE Payment_12(
		Booking_id VARCHAR (10) NOT NULL,
		amount int(10),
		PRIMARY KEY (Booking_id),
		FOREIGN KEY (Booking_id) REFERENCES Booking(Booking_id) ON DELETE CASCADE
);
```

```
CREATE TABLE PropertyReview(
            property_id VARCHAR (10),
            By_Guest VARCHAR (50),
            review_date datetime,
            comment_info VARCHAR(50),
            rating float(10),
            title VARCHAR(20),
            PRIMARY KEY (property_id,By_Guest),
            FOREIGN KEY (property_id) REFERENCES Property(Property_id) ON DELETE CASCADE,
            FOREIGN KEY (By_Guest) REFERENCES Guest(guest_name) ON DELETE CASCADE
);

CREATE TABLE Property(
            Property_id VARCHAR (10) ,
            description VARCHAR (20),
            photo_url VARCHAR(50),
            availability BOOLEAN,
            property_type VARCHAR (20),
            bedroom_cnt int(10),
            bathroom_cnt int(10),
            cancellation_policy VARCHAR(20),
            rent_price int(10),
            owner VARCHAR(10),
            Zipcode int(10) NOT NULL,
            available_rooms int(10),
            PRIMARY KEY (Property_id),
            FOREIGN KEY (owner) REFERENCES HostDetails(hostname) ON DELETE CASCADE,
            FOREIGN KEY (Zipcode) REFERENCES Location(Zipcode) ON DELETE CASCADE
);

CREATE TABLE Location(
            Zipcode int(10) NOT NULL,
            street VARCHAR(50),
            city VARCHAR(20),
            country VARCHAR(20),
            PRIMARY KEY (Zipcode,street,city)
);

CREATE TABLE Events(
            Title VARCHAR(20),
            addresss VARCHAR(50),
            entry_fee float(10),
            event_time datetime,
            category VARCHAR(20),
            description VARCHAR(50),
            PRIMARY KEY (Title)
);
```

```
CREATE TABLE Has_nearby(
        title VARCHAR(20),
        zipcode int(10),
        PRIMARY KEY (Title,Zipcode),
        FOREIGN KEY (title) REFERENCES Events(Title) ON DELETE CASCADE,
        FOREIGN KEY (zipcode) REFERENCES Location(Zipcode) ON DELETE CASCADE
);
```

PL/SQL Triggers

Procedural Language

1. ***Procedure to set a host as a super host if they have an overall rating 4.8 or above in the past year, have >=90% response rate and have done 10 or more stays in the past year.***

```
CREATE OR REPLACE PROCEDURE upgradeSuperhost(hname IN VARCHAR) AS

overall_rating HostDetails.hostname%TYPE;
total_bookings NUMBER;
thisresponserate HostDetails.response_rate%TYPE;


BEGIN

        -- Get overall rating in the past year for the host.

        SELECT AVG(P.rating) INTO overall_rating
        FROM PropertyReview PR, Property P
        WHERE PR.Property_id = P.Property_id
                AND P.owner = hname
                AND PR.review_date >= DATEADD(year,-1,GETDATE());

        -- Get total bookings made for all properties under the host in the past year and have status as
        "Completed".

        SELECT COUNT(*) INTO total_bookings
        FROM Books_11 B1, Books_12 B2, Property P
        WHERE B1.Property_id = P.Property_id
                AND P.owner = hname
                AND P.booking_date >= DATEADD(year, -1, GETDATE())
                AND B2.booking_status = "Completed",
                AND B1.booking_id = B2.booking_id;


        -- Get response rate for the host.

        SELECT response_rate INTO thisresponserate
        FROM HostDetails
        WHERE hostname = hname;
```

```
    -- Set is_Superhost to True if host matches the criteria.

    IF overall_rating >= 4.8 AND total_bookings >= 10 AND thisresponserate THEN
        UPDATE HostDetails
        SET is_superhost = TRUE
        WHERE hostname = hname;
    END IF;
END;
```

---

2. ***Procedure to calculate payment amount based on room rent,number of rooms, number of nights and taxes. Assume tax to be 10% of total amount.***

```
CREATE OR REPLACE PROCEDURE calculateAmount(prop_id IN Property.Property_id%TYPE, b_id IN
Booking.Booking_id%TYPE, amt OUT Payment.amount%TYPE) AS

num_rooms Booking.no_rooms%TYPE;
num_nights NUMBER;
room_rent Property.rent_price%TYPE;


BEGIN

    -- Fetch number of rooms booked.

    SELECT no_rooms INTO num_rooms
    FROM Booking
    WHERE Booking_id = b_id;


    -- Calculate number of nights based on start and end date

    SELECT (TRUNC(end_date) - TRUNC(start_date) - 1) INTO num_nights
    FROM Booking
    WHERE Booking_id = b_id;
    -- Fetch price of the room.

    SELECT rent_price into room_rent
    FROM Property
    WHERE Property_id = prop_id;


    -- Calculate total amount and store it in OUT variable amt.

    amt := 1.1 * (num_rooms * num_nights * room_rent);

END;
```

Triggers

---

1. ***Trigger to insert a payment into the payment table whenever a booking is created. Amount is calculated using the previous procedure.***

```
CREATE OR REPLACE TRIGGER insertPayment
AFTER INSERT ON Books_11
FOR EACH ROW

DECLARE

amt Payment.amount%TYPE;

BEGIN

        -- Calculate the payment amount using the previously defined procedure.

        calculateAmount(:new.Property_id, :new.Booking_id, amt);

        -- Insert a new transaction record with the respective booking_id, amount and payment status
        as "Pending".

        INSERT INTO Payment_12
          ( Booking_id,
                amt)
          VALUES
          ( :new.Booking_id,
                amt);
        INSERT INTO Payment_11
          ( booking_id,
                booking_status)
          VALUES
          ( :new.Booking_id,
                "Pending");
END;
```

---

2. ***Trigger to check if number of rooms booked is less than rooms available for the property. If it is, decrement available rooms for the property by number of rooms booked.***

```
CREATE OR REPLACE TRIGGER checkRooms
BEFORE INSERT OR UPDATE ON Books_11
FOR EACH ROW

DECLARE
avl_rooms Property.available_rooms%TYPE;
num_rooms Booking.no_rooms%TYPE;
```

```
BEGIN


        -- Fetch total number of rooms on the property available

        SELECT available_rooms INTO avl_rooms
        FROM Property
        WHERE Property_id = :new.Property_id;


        -- Fetch number of rooms booked

        SELECT no_rooms INTO num_rooms
        FROM Booking
        WHERE Booking_id = :new.Booking_id;


        -- Check if the number of rooms being booked is greater than the available_rooms

        IF avl_rooms >= num_rooms THEN

                -- If yes, then reduce the available_rooms by the number of rooms booked.

                UPDATE Property
                SET available_rooms = available_rooms - num_rooms
                WHERE Property_id = :new.Property_id;
        ELSE

                Raise_Application_Error(-20000, 'Not enough rooms available!');

        END IF;

END;
```

****