AirBnB-4

Course: CS6360 - Database Design
Section: 003
Team number: 4
Team members:
Raisaat Rashid
Hemanjeni Kundem
Taaha Kamal

Part 1: Project Requirements

- Each stay or **property** has a name, address, bedroom count, bathroom count, price per night, a maximum number of guests allowed, cleaning fee, available booking slots that show the dates when the property is available, check-in time, check-out time, an average rating, the total number of ratings for the property
- Each **property** can also belong to multiple categories (like cozy, entire house, apartment, etc.) and can have multiple bedrooms with each bedroom having the bed count and type of bed in each bedroom listed.
- Each **property** can have multiple house rules (like no-smoking, no pets allowed, etc.) and multiple amenities (like Wi-Fi, air conditioning, etc.)
- Each **property** can also have multiple photos and has a cancellation policy which lists whether the stay at the property is refundable if cancelled, the percentage of the price refunded and the time period within which it is refundable
- Each **property** must also have a unique ID
- There are two kinds of **users** - a **host** (who hosts the stay at a property) and a **guest** (who stays at a property)
- Each **user** account has an optional profile picture, about section, the user's first name, last name, date of birth, address, gender, phone number, details about one emergency contact (the contact's name, relationship to the user, preferred language, email address, country code and phone number), the user's email address and password
- Each **user** must have a unique user ID
- A **host** can be a super host and a host account can have a rating along with the total number of ratings for the host
- A **guest** account can also have a rating and the total number of ratings for the guest
- A **guest** can have a number of **wishlists**. Each **wishlist** has:
    - A name
    - A privacy
    - A list of properties in it (if any).
- Every **booking** of a stay made by a guest has a unique booking ID, booking date (when the booking was made), check-in date (the guest arrival date), check-out date (the date when the guest leaves), the number of guests for the stay, the date the booking was modified (if modified).
- Every **booking** lists the tax paid, the total price for the stay, the total price with taxes (and cleaning fee), the amount of the total price with taxes that has been paid, the amount due, as well as the promo code and the discount amount (if any promo was applied)
- Every **booking** also lists whether the booking has been cancelled, the cancellation date, the refund amount and the refund amount paid
- Each **guest** must have a **credit card** to pay for the stay. Each **credit card** must have the following:
    - A unique card number
    - A 3-digit CSV number
    - An expiration date

    - The name of the cardholder
    - The type of the card
    - Billing address
- A **credit card** can only belong to one **guest**
- Each host must have a **bank account** where the amount for a stay hosted by the host will be deposited. Each **bank account** must have the following details:
    - A unique account number
    - A routing number
    - The type of the account
- A **bank account** can only belong to one **host**

The following relationships were noted between the different components of the system:

- A host can send 0 or more messages to a guest and vice-versa, and each message should list who created it, whom it was sent to, the date of creation and the message body itself.
- A guest can review 0 or more properties and the property can contain 0 or more reviews from guests. Each review should have the following:
    - Multiple photos of the property (if any)
    - A comment
    - A cleanliness rating
    - A communication rating
    - A check-in rating
    - An accuracy rating
    - A location rating
    - A value rating
    - An overall rating
    - The date when the review was created
    - The date when the review was modified (if modified)
- A property listing can be included in 0 or more wishlists, and a wishlist can have 0 or more property listings. If a property is included in a wishlist, it may have the following information:
    - Check-in date (the date when the guest wishes to stay in the property)
    - Check-out date (the date when the guest wishes to leave the property)
- A guest can have 0 or more wishlists
- Each booking should have exactly one property listed, but a property can be in 0 or more bookings
- Each booking is made by exactly one guest but a guest can make 0 or more bookings
- A host can review 0 or more guests and a guest can review 0 or more hosts. Such a review relationship must have the following information:
    - The comment for the guest by the host
    - The rating given by the host to the guest
    - The date the review for the guest was created by the host
    - The date the review for the guest was modified by the host (if it was modified)

- The comment for the host by the guest
- The rating given by the guest to the host
- The date the review for the host was created by the guest
- The date the review for the host was modified by the guest (if it was modified)

Part 2: EER Diagram

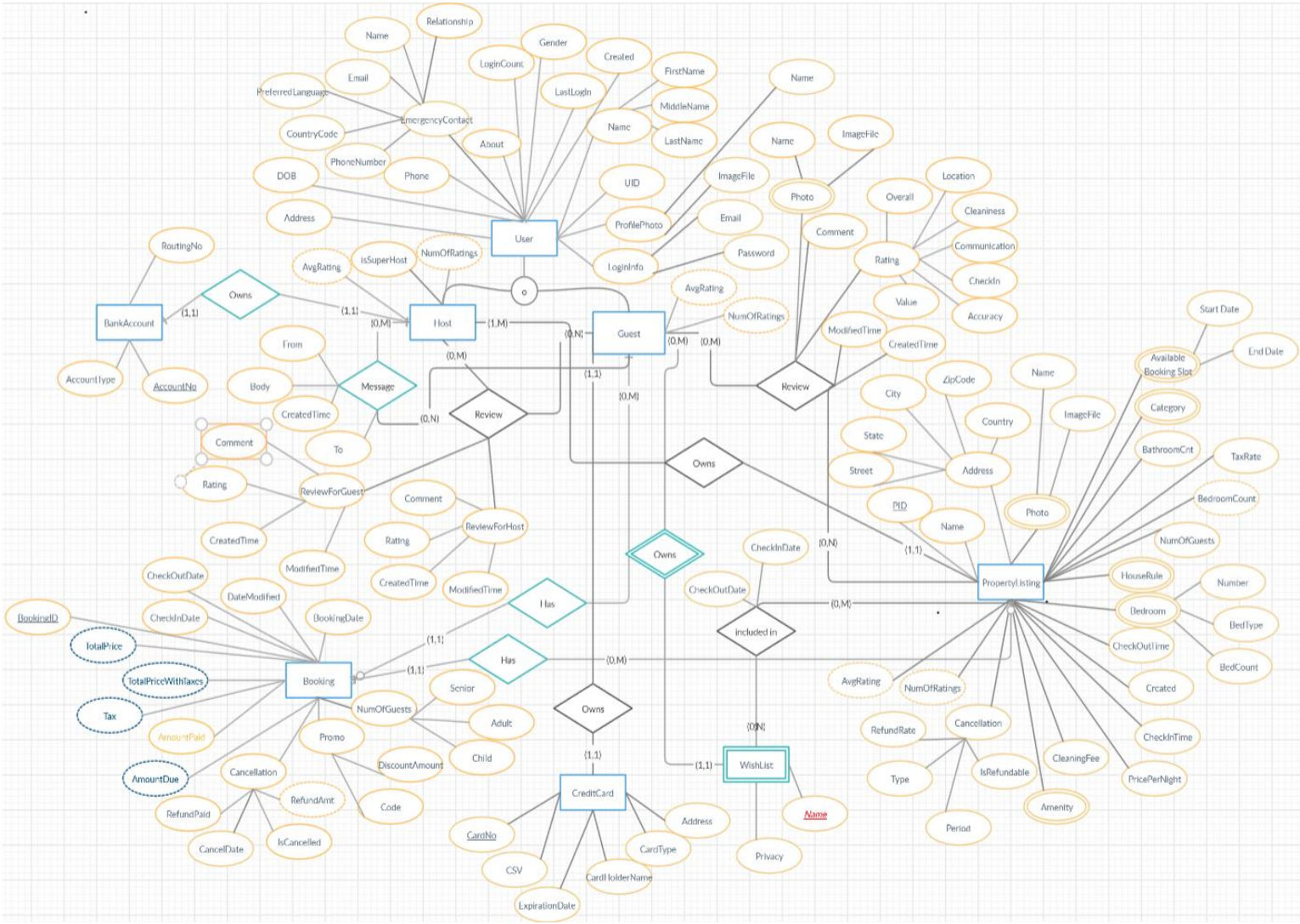
Assumptions made:

- A **host** can own only one **bank account**
- A **guest** can own only one **credit card**
- A **host** must own at least one **property listing**
- A **property listing** can only belong to one **host**
- A **wishlist** can be owned by only one guest

Additional notes about the EER diagram:

- TotalPrice in booking is derived from the corresponding property listing's pricePerNight and the total stay (the booking's check-out date - check-in date)
- Tax in booking is derived from the booking's TotalPrice and the corresponding property listing's TaxRate
- The TotalPriceWithTaxes in booking is derived from the booking's tax, TotalPrice, DiscountAmount from promo (if any) and the corresponding property listing's CleaningFee
- The booking's AmountDue is derived from the booking's TotalPriceWithTaxes and AmountPaid
- numOfRatings and avgRating of property listing is derived from the review relationship between guest and property listing
- avgRating and numOfRatings of host and guest are derived from the review relationship between them
- BedroomCnt in property listing can be derived from the number of values in the multivalued attribute, Bedroom, of property listing
- RefundAmt in booking is derived from the corresponding property listing's refund rate and the booking's TotalPriceWithTaxes
- The name attribute in wishlist is a partial key

The EER diagram is in the next page



<u>Relationships</u>

One-to-one: Host Owns BankAccount, Guest Owns CreditCard

One-to-many: Guest Has Booking, PropertyListing Has Booking

Many-to-many: Guest Review PropertyListing, Host Message Guest

There are are more relationships in the EER Diagram

Part 3: Relational Schema Mapping

PropertyListing

PID	Name	Street	City	State	Country	Zipcode	BathroomCnt	BedroomCnt	GuestNum	PricePerNight	TaxRate	CleaningFee	Created	CheckInTime	CheckOutTime	IsRefundable	CancellationPeriod	CancellationType	RefundRate	NumOfRatings	AvgRating	HID
																						F.K(Host.UID)

Category

PropertyID	Name
F.K. (PropertyListing.PID)	

Bedroom

PropertyID	Number	BedType	BedCnt
F.K. (PropertyListing.PID)			

AvailableBookingSlot

PropertyID	StartDate	EndDate
F.K. (PropertyListing.PID)		

HouseRule

PropertyID	RuleName
F.K. (PropertyListing.PID)	

PropertyPhoto

PropertyID	Name	ImageFile
F.K. (PropertyListing.PID)		

Amenity

PropertyID	Name
F.K. (PropertyListing.PID)	

User

UID	DOB	Email	Password	Gender	About	Phone	ProfilePhotoName	ProfilePhotoFile	Address	Fname	MInitial	LName	Created	LoginCnt	LastLogin	EmName	EmRelationship	EmPreferredLang	EmEmail	EmCountryCode	EmPhone

Host

UID	IsSuperHost	AvgRating	NumOfRatings	BankAccNum
F.K. (User.UID)				F.K. (BankAccount.Number)

Guest

UID	AvgRating	NumOfRatings	CreditCardNum
F.K. (User.UID)			F.K. (CreditCard.CardNum)

Message

HostUID	GuestUID	Created	To	From	Body
F.K. (Host.UID)	F.K. (Guest.UID)				

Booking

BID	CheckInDate	CheckOutDate	Tax	Total Price	Total Price WTaxes	AmountPaid	AmountDue	BookingDate	ModifiedDate	SeniorGuestNum	AdultGuestNum	ChildGuestNum	IsCancelled	RefundAmt	RefundPaid	CancelDate	PromoCode	GuestUID	PID
																	F.K. (Promo.Code)	F.K. (Guest.UID)	F.K. (PropertyListing.PID)

Promo

Code	DiscountAmt

CreditCard

CardNum	CSV	ExpirationDate	CardholderName	CardType	Address

ReviewForUsers

HostUID	GuestUID	GuestRating	HostRating	CommentForHost	CommentForGuest	ReviewForHostCreated	ReviewForGuestCreated	ReviewForHostModified	ReviewForGuestModified
F.K. (Host.UID)	F.K. (Guest.UID)								

WishList

UID	Name	Privacy
F.K. (Guest.UID)		

PropertyIncludedInWishlist

PID	UID	WishlistName	CheckInDate	CheckOutDate
F.K. (PropertyListing.PID)	F.K. (Host.UID)	F.K. (WishList.Name)		

BankAccount

Number	RoutingNum	AccountType

PhotoForPropertyReview

GuestPID	PID	Name	ImageFile
F.K. (Guest.UID)	F.K. (PropertyListing.PID)		

ReviewForProperty

GuestPID	PID	Created_Time	Modified_Time	Comment	Cleanliness_Rating	Communication_Rating	CheckIn_Rating	Accuracy_Rating	Location_Rating	Value_Rating	Overall_Rating
F.K. (Guest.UID)	F.K. (PropertyListing.PID)										

Part 4: Functional Dependencies and Normalization

Functional Dependencies:

PropertyListing
PID -> Name, Street, City, State, Country, Zipcode, BathroomCnt, BedroomCnt, GuestNum, PricePerNight, TaxRate, CleaningFee, Created, CheckInTime, CheckOutTime, IsRefundable, CancellationPeriod, CancellationType, RefundRate, CheckInTime, NumOfRatings, AvgRatings

Bedroom
PropertyID, Number -> BedType, BedCnt

AvailableBookingSlot
PID, StartDate -> EndDate

User
UID -> DOB, Email, Password, Gender, About, Phone, ProfilePhotName, ProfilePhotoFile, Address, Fname, MInitial, LName, Created, LoginCnt, LastLogin, EmName, EmRelationship, EmPreferredLang, EmEmail, EmCountryCode, EmPhone

Host
UID -> IsSuperHost, AvgRating, NumOfRatings

Guest
UID -> AvgRating, NumOfRatings

Message
HostUID, GuestID -> Created, To, From, Body

Booking
BID -> CheckInDate, CheckOutDate, Tax, TotalPrice, TotalPriceWTax, AmountPaid, AmountDue, BookingDate, ModifiedDate, SeniorGuestNum, AdultGuestNum, ChildGuestNum, IsCancelled, RefundAmt, RefundPaid, CancelDate

Promo
Code -> Discount_Amt

BankAccount
Number-> RoutingNum, AccountType

CreditCard
CardNum -> CSV, ExpirationDate,CardholderName, CardType, Address

ReviewForUsers

HostUID,GuestUID-> GuestRating, HostRating, CommentForHost, CommentForGuest, ReviewForHostCreated, ReviewForGuestCreated, ReviewForHostModified, ReviewForGuestModified,

WishList
UID, Name -> Privacy

PropertyIncludedInWishlist
PID, UID, WishlistName-> CheckInDate,CheckOutDate

ReviewForProperty
GuestID, PID -> Created_Time, Modified_Time, Comment, Cleanliness_Rating, Comment_Rating, Check-In_Rating, Accuracy_Rating, Location_Rating, Value_Rating, Overall_Rating

Normalization:

1NF

The relations are already in 1NF.

2NF

The relations are already in 2NF.

3NF

The relations are already in 3NF.

Therefore, the relations remain unchanged

Part 5: Final Relational Schema

Since the initial schema did not violate 1NF, 2NF and 3NF, the schema has not changed; the initial schema is the final schema **(reference part 3 to see it)**

Part 6: Table Creation using SQL

Code for table creation:

```
DROP TABLE ReviewForProperty;
CREATE TABLE ReviewForProperty(
GuestID INT,
PID INT ,
Created_Time Date NOT NULL,
Modified_Time Date,
CommentInReview VARCHAR(1000) ,
Cleanliness_Rating NUMBER(2,1),
Communication_Rating NUMBER(2,1),
CheckIn_Rating NUMBER(2,1),
Accuracy_Rating NUMBER(2,1),
Location_Rating NUMBER(2,1),
Value_Rating NUMBER(2,1),
Overall_Rating NUMBER(2,1),
PRIMARY KEY(GuestID,PID));

DROP TABLE PhotoForPropertyReview;
CREATE TABLE PhotoForPropertyReview(
GuestID INT ,
PID INT ,
PhotoName VARCHAR(50) ,
ImageFile BLOB NOT NULL,
PRIMARY KEY(GuestID,PID,PhotoName));

DROP TABLE PropertyIncludedInWishlist;
CREATE TABLE PropertyIncludedInWishlist(
PID INT ,
AirBnBUID INT ,
WishlistName VARCHAR(50) ,
CheckInDate DATE,
CheckOutDate DATE,
PRIMARY KEY(PID,AirBnBUID,WishlistName));

DROP TABLE WishList;
CREATE TABLE WishList(
AirBnBUID INT ,
WishlistName VARCHAR(50) NOT NULL UNIQUE,
Privacy CHAR(1),
PRIMARY KEY(AirBnBUID,WishlistName));

DROP TABLE ReviewForUsers;
CREATE TABLE ReviewForUsers(
HostUID INT ,
GuestUID INT ,
GuestRating NUMBER(2,1),
```

```
HostRating  NUMBER(2,1),
CommentForHost VARCHAR(1000),
CommentForGuest VARCHAR(1000),
ReviewForHostCreated DATE,
ReviewForGuestCreated DATE,
ReviewForHostModified DATE,
ReviewForGuestModified DATE,
PRIMARY KEY(HostUID,GuestUID));

DROP TABLE CreditCard;
CREATE TABLE CreditCard(
CardNum INT ,
CSV INT NOT NULL,
ExpirationDate DATE  NOT NULL,
CardholderName VARCHAR(50) NOT NULL,
CardType CHAR(6) NOT NULL,
Address VARCHAR(100),
PRIMARY KEY(CardNum));

DROP TABLE BankAccount;
CREATE TABLE BankAccount(
AccountNUMBER INT ,
RoutingNum INT NOT NULL,
AccountType VARCHAR(20) NOT NULL,
PRIMARY KEY(AccountNUMBER));

DROP TABLE Promo;
CREATE TABLE Promo(
Code VARCHAR(10),
Discount_Amt NUMBER NOT NULL,
PRIMARY KEY(Code));

DROP TABLE Message;
CREATE TABLE Message(
HostUID INT ,
GuestID INT ,
Created DATE NOT NULL,
Message_To INT NOT NULL,
Message_From INT NOT NULL,
Body VARCHAR(1000),
PRIMARY KEY(HostUID,GuestID));

DROP TABLE Guest;
CREATE TABLE Guest(
AirBnBUID INT ,
AvgRating NUMBER(2,1) ,
NumOfRatings INT DEFAULT 0,
CreditCardNum INT NOT NULL UNIQUE,
PRIMARY KEY(AirBnBUID));
```

```
DROP TABLE Host;
CREATE TABLE Host(
AirBnBUID INT ,
IsSuperHost CHAR(1) ,
AvgRating NUMBER(2,1),
NumOfRatings INT,
BankAccountNumber INT NOT NULL UNIQUE,
PRIMARY KEY(AirBnBUID));

DROP TABLE AirBnBUser;
CREATE TABLE AirBnBUser(
AirBnBUID INT,
DOB DATE,
Email VARCHAR(20) NOT NULL,
UserPassword VARCHAR(20) NOT NULL,
Gender CHAR(1),
About VARCHAR(100),
Phone VARCHAR(15) NOT NULL,
ProfilePhotoName VARCHAR(20),
ProfilePhoto BLOB,
Address VARCHAR(100),
Fname VARCHAR(20) NOT NULL,
MInitial VARCHAR(20),
LName  VARCHAR(20),
Created TIMESTAMP,
LoginCnt INT,
LastLogin TIMESTAMP,
EmName VARCHAR(20),
EmRelationship VARCHAR(20),
EmPreferredLang VARCHAR(15),
EmEmail VARCHAR(20) NOT NULL,
EmCountryCode VARCHAR(3) NOT NULL,
EmPhone VARCHAR(15) NOT NULL,
PRIMARY KEY(AirBnBUID));

DROP TABLE Amenity;
CREATE TABLE Amenity(
PID INT ,
AmenityName VARCHAR(20),
PRIMARY KEY(PID,AmenityName));

DROP TABLE PropertyPhoto;
CREATE TABLE PropertyPhoto(
PID INT ,
PropertyName VARCHAR(20),
ImageFile BLOB NOT NULL,
PRIMARY KEY(PID,PropertyName));
```

```
DROP TABLE HouseRule;
CREATE TABLE HouseRule(
PID INT ,
RuleName VARCHAR(20),
PRIMARY KEY(PID,RuleName));

DROP TABLE AvailableBookingSlot;
CREATE TABLE AvailableBookingSlot(
PID INT ,
StartDate DATE,
EndDate DATE,
PRIMARY KEY(PID,StartDate,EndDate));

DROP TABLE Bedroom;
CREATE TABLE Bedroom(
PropertyID INT ,
BedroomNumber VARCHAR(5),
BedType VARCHAR(10),
BedCnt INT,
PRIMARY KEY(PropertyID,BedroomNumber,BedType,BedCnt));

DROP TABLE Category;
CREATE TABLE Category(
PID INT ,
categoryName VARCHAR(20),
PRIMARY KEY(PID,categoryName));

DROP TABLE PropertyListing;
CREATE TABLE PropertyListing(
PID INT ,
PropertyName VARCHAR(50) ,
Zipcode INT NOT NULL,
BathroomCnt INT,
BedroomCnt INT DEFAULT 0,
GuestNum INT,
PricePerNight NUMBER(6,2),
CleaningFee NUMBER(4,2),
Created Date,
CheckInTime TIMESTAMP,
CheckOutTime TIMESTAMP,
IsRefundable CHAR(1),
CancellationPeriod INT,
CancellationType VARCHAR(10),
RefundRate NUMBER(2,1),
NumOfRatings INT DEFAULT 0,
AvgRatings NUMBER(2,1) DEFAULT 0,
HID INT,
Street VARCHAR(20),
City VARCHAR(20),
```

```
StateofResidence VARCHAR(20),
Country VARCHAR(20),
TaxRate NUMBER(2,1),
PRIMARY KEY(PID));

DROP TABLE Booking;
CREATE TABLE Booking(
BID INT,
CheckInDate DATE NOT NULL,
CheckOutDate DATE NOT NULL,
AmountPaid NUMBER(6,2),
BookingDate DATE NOT NULL,
ModifiedDate DATE,
SeniorGuestNum INT DEFAULT 0,
AdultGuestNum INT DEFAULT 0,
ChildGuestNum INT DEFAULT 0,
IsCancelled CHAR(1),
RefundPaid CHAR(1),
CancelDate DATE,
PromoCode  VARCHAR(10),
GuestUID INT,
PID INT,
TotalPrice NUMBER(6,2),
Tax NUMBER(4,2),
TotalPriceWTax NUMBER(6,2),
AmountDue NUMBER(4,2),
RefundAmt NUMBER(4,2),
PRIMARY KEY(BID));

ALTER TABLE Guest ADD CONSTRAINT GuestFK_1 FOREIGN KEY(AirBnBUID) REFERENCES
AirBnBUser(AirBnBUID) ON DELETE  CASCADE;
ALTER TABLE Guest ADD CONSTRAINT GuestFK_2 FOREIGN KEY(Creditcardnum)
REFERENCES CREDITCARD(CARDNUM) ON DELETE  CASCADE;
ALTER TABLE Host ADD CONSTRAINT HostFK_1 FOREIGN KEY(AirBnBUID) REFERENCES
AirBnBUser(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE Host ADD CONSTRAINT HostFK_2 FOREIGN KEY(BANKACCOUNTNUMBER)
REFERENCES  BANKACCOUNT(ACCOUNTNUMBER) ON DELETE  CASCADE ;
ALTER TABLE ReviewForProperty ADD CONSTRAINT ReviewForPropertyFK_1 FOREIGN
KEY(GuestID) REFERENCES Guest(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE ReviewForProperty ADD CONSTRAINT ReviewForPropertyFK_2 FOREIGN
KEY(PID) REFERENCES  PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE PhotoForPropertyReview ADD CONSTRAINT PhotoForPropertyReviewFK_1
FOREIGN KEY(GuestID) REFERENCES Guest(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE PhotoForPropertyReview ADD CONSTRAINT PhotoForPropertyReviewFK_2
FOREIGN KEY(PID) REFERENCES  PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE PropertyIncludedInWishlist ADD CONSTRAINT
PropertyIncludedInWishlistFK_1 FOREIGN KEY(AirBnBUID) REFERENCES
Guest(AirBnBUID) ON DELETE  CASCADE ;
```

```
ALTER TABLE PropertyIncludedInWishlist ADD CONSTRAINT
PropertyIncludedInWishlistFK_2 FOREIGN KEY(PID) REFERENCES
PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE PropertyIncludedInWishlist ADD CONSTRAINT
PropertyIncludedInWishlistFK_3 FOREIGN KEY(WishlistName) REFERENCES
WishList(WishlistName) ON DELETE  CASCADE;
ALTER TABLE WishList ADD CONSTRAINT WishListFK_1 FOREIGN KEY(AirBnBUID)
REFERENCES Guest(AirBnBUID) ON DELETE  CASCADE;
ALTER TABLE ReviewForUsers ADD CONSTRAINT ReviewForUsersFK_1 FOREIGN
KEY(HostUID) REFERENCES Host(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE ReviewForUsers ADD CONSTRAINT ReviewForUsersFK_2 FOREIGN
KEY(GuestUID) REFERENCES  Guest(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE Message ADD CONSTRAINT MessageFK_1 FOREIGN KEY(HostUID)
REFERENCES Host(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE Message ADD CONSTRAINT MessageFK_2 FOREIGN KEY(GuestID)
REFERENCES  Guest(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE Amenity ADD CONSTRAINT AmenityFK_1 FOREIGN KEY(PID) REFERENCES
PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE PropertyPhoto ADD CONSTRAINT PropertyPhotoFK_1 FOREIGN KEY(PID)
REFERENCES PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE HouseRule ADD CONSTRAINT HouseRuleFK_1 FOREIGN KEY(PID)
REFERENCES PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE AvailableBookingSlot ADD CONSTRAINT AvailableBookingSlotFK_1
FOREIGN KEY(PID) REFERENCES PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE Bedroom ADD CONSTRAINT BedroomFK_1 FOREIGN KEY(PropertyID)
REFERENCES PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE Category ADD CONSTRAINT CategoryFK_1 FOREIGN KEY(PID) REFERENCES
PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE PROPERTYLISTING ADD CONSTRAINT propertyFK_1 FOREIGN KEY(HID)
REFERENCES Host(AIRBNBUID) ON DELETE  CASCADE ;
ALTER TABLE Booking ADD CONSTRAINT BookingFK_1 FOREIGN KEY(GuestUID)
REFERENCES Guest(AirBnBUID) ON DELETE  CASCADE ;
ALTER TABLE Booking ADD CONSTRAINT BookingFK_2 FOREIGN KEY(PID) REFERENCES
PropertyListing(PID) ON DELETE  CASCADE ;
ALTER TABLE Booking ADD CONSTRAINT BookingFK_3 FOREIGN KEY(PROMOCODE)
REFERENCES  Promo(Code) ON DELETE  CASCADE ;
```

Part 7: Stored Procedures and Triggers

Stored Procedure #1:

The following procedure is used to fetch surrounding properties of given a property

```
CREATE OR REPLACE PROCEDURE GetSurroundingProperties(PropertyID IN INT) IS

zip INT;
recordproperty PROPERTYLISTING%rowtype;
 propid INT;
 propname varchar(20);
 CURSOR cprop IS
    SELECT PID,PROPERTYNAME
    FROM PROPERTYLISTING
    WHERE ZIPCODE =zip;
 BEGIN
  SELECT ZIPCODE into zip  FROM PROPERTYLISTING  WHERE PID=PropertyID;
  open cprop;
   LOOP
        FETCH cprop INTO propid,propname;
        EXIT WHEN cprop%NOTFOUND;
        dbms_output.put_line(propid||propname);
    end loop;
  close cprop;
 END;

 set serveroutput on size 30000;
exec GetSurroundingProperties(1);
```

Stored Procedure #2:

The following procedure is used to get properties which have greater than X bedrooms, where X is determined by the user

```
SET SERVEROUTPUT ON;
create or replace
PROCEDURE GetMinimumBedroomNumber(bedCnt IN INT) AS beds INT;

recordproperty PROPERTYLISTING%rowtype;
 propId INT;
 propName varchar(20);

CURSOR properties IS
  SELECT PID, PROPERTYNAME
  FROM PROPERTYLISTING
  WHERE BEDROOMCNT >= bedCnt;
```

```
BEGIN

  open properties;
  LOOP
      FETCH properties INTO propId, propName;
      EXIT WHEN properties%NOTFOUND;
      dbms_output.put_line(propId||propName);
  end loop;
close properties;
END;
```

Trigger #1:
The following trigger increments the number of ratings of a property and updates the average rating of the property when a new review for the property is added.

```
create or replace TRIGGER Update_Property_Rating
AFTER INSERT ON ReviewForProperty
FOR EACH ROW
BEGIN
      UPDATE PropertyListing
      SET AvgRatings = (AvgRatings * NumOfRatings +
:new.Overall_Rating)/(NumOfRatings + 1)
      WHERE PID = :new.PID;
      UPDATE PropertyListing
      SET NumOfRatings = NumOfRatings + 1
      WHERE PID = :new.PID;
END;
```

Trigger #2:
The following trigger increments the bedroom count of a property when a new bedroom tuple is added for the property in the bedroom table.

```
create or replace TRIGGER Update_Bedroom_Count
AFTER INSERT ON Bedroom
FOR EACH ROW
BEGIN
      UPDATE PropertyListing
      SET BedroomCnt = BedroomCnt + 1
      WHERE PID = :new.PropertyID;
END;
```