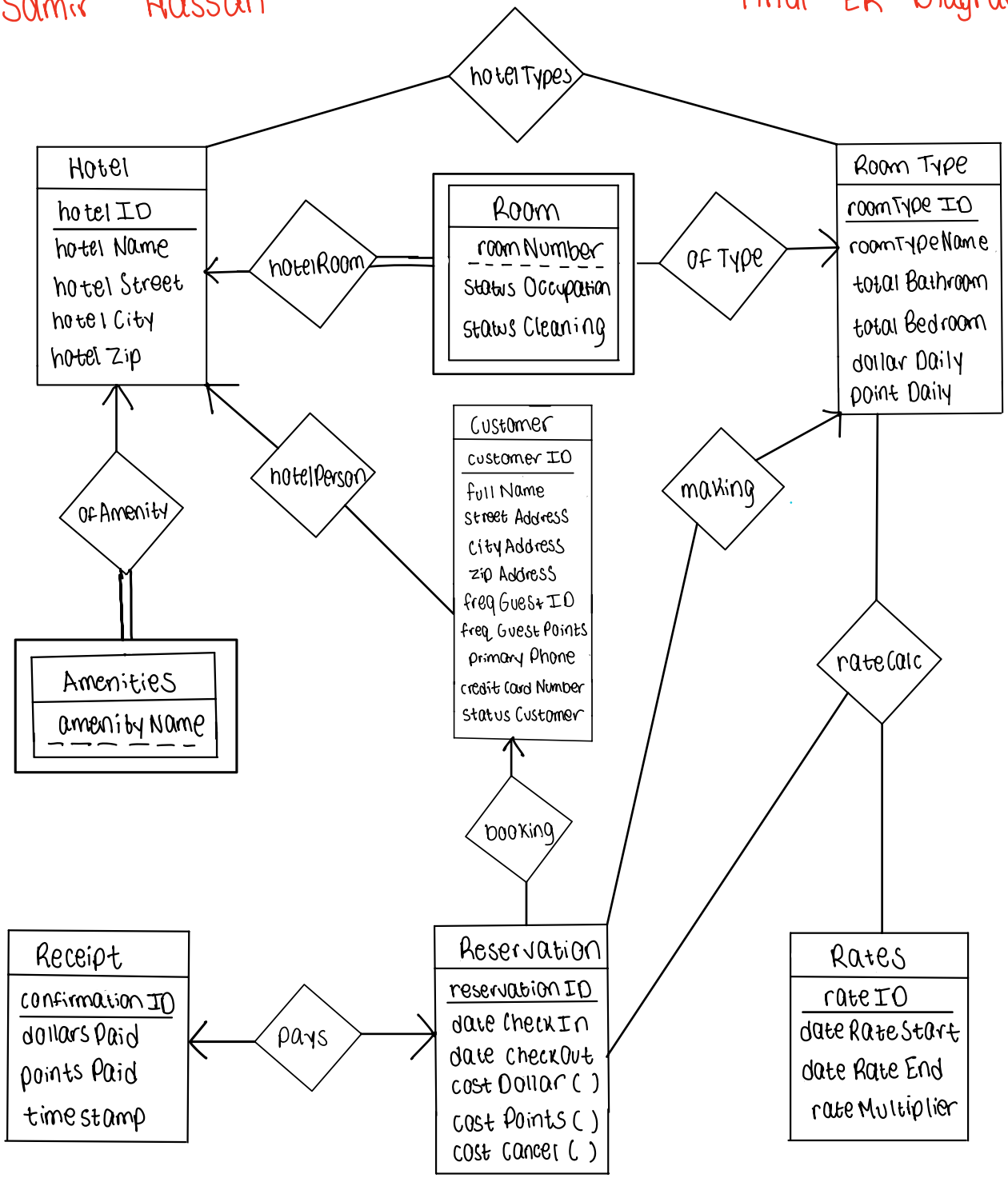
Samir Hassan

Final ER Diagram



Notes for final ER diagram submission

Entities

1. **Hotel**: Because we are assuming only US addresses there is no point in including the country of the hotel. The hotelCity is needed for our customer online reservation access interface.
2. **Room**: Room is a weak entity set dependent on the Hotel. Then, Room has a discriminating attribute of roomNumber. Combining the discriminating attribute with the Hotel's primary key, it has its own primary key. The statusOccupation and statusCleaning will be used for the housekeeping interface.
3. **Amenities**: Amenities is a weak entity set dependent on the Hotel. This is because Amenities' existence depends on Hotel. Then, Amenities have a discriminating attribute of amenityName. Combining the discriminating attribute with the Hotel's primary key, it has its own primary key.
4. **RoomType**: RoomType has the dollarDaily and pointDaily attributes to show how much a room of that type costs per daily depending on the payment method. Lastly, a roomID is used because roomType with the same name might have a different number of bedrooms and bathrooms.
5. **Customer**: I've excluded a creditCard or a paymentMethod entity because for this enterprise I'm using the assumption that a Customer has only one credit card as his payment method. From there, a creditCard expiration nor a security code is necessary. There is a statusCustomer attribute to keep track of whether the customer is currently occupying a room, future guest, or past guest, etc. I'll keep track of if he's joined the frequent guest program through the freqGuestID attribute (make it NULL).
6. **Rates**: I've included a rateMultiplier for the fact that we can multiply the base rate depending on whether it's peak hotel season (prices are expensive) or no one's booking rooms (prices are low).
7. **Reservation**: The dateCheckIn and dateCheckOut are NULL data till a Customer both checks in and checks out respectively. Since we are not allowing advance check-in, it is not necessary to add this data in advance. I've included costDollar and costPoints as derived attributes. Then, I use the dollarDaily/pointsDaily from RoomType and dateRateStart, dateRateEnd, and rateMultiplier from Rates to calculate a price for the room the Customer is in. costCancel is also a derived attribute because if the Customer cancels their Reservation, they are charged a fee in dollars. I didn't do a separate cancellation attribute for points and dollars because if a Customer used dollars to pay then they are charged a fee, but if they've used points to pay, as a member of the Frequent Guest Program, I don't charge them a cancellation fee.
8. **Receipt**: Receipt is used to confirm the dollar or points amount that has been paid. We also record the timestamp of the payment.

Relations

1. **hotelTypes**: relates Hotel and RoomType; Many to many because many RoomTypes can be associated with many Hotels. I assume that a room type isn't restricted to one specific hotel in our enterprise .
2. **hotelRoom**: relates Hotel and hotelRoom; self explanatory
3. **ofType**: relates Room and RoomType; self explanatory
4. **making**: relates Reservation and RoomType; One to many from RoomType to Reservation because a RoomType can be associated with many reservations. Remember, we are assigning the Customer a room of the reserved type therefore assuming that a type may have many rooms.
5. **booking**: relates Customer and Reservation; self explanatory
6. **rateCalc**: relates roomType, Rates, and Reservation. I need this ternary relation because we need attributes from all three entities to calculate the price, etc. It also allows for better access and expressive power of information because a roomType can have many Rates and many Reservations. A Reservation costDollar will depend on the Rates entity while it also depends on the RoomType. The Rates rateMultiplier is associated with many RoomTypes and it affects the Reservation.
7. **pays**: relates Reservation and Receipt; One to one because a reservation can only have one receipt and vice-versa.
8. **ofAmenity**: relates Hotel and Amenities; self explanatory
9. **hotelPerson**: relates Hotel and Customer; One to many from Hotel to Customer because I am assuming that a Hotel can be associated with many Customers but a Customer can only be associated with only one hotel. What happens then if the same customer wants to reserve a room at another hotel? Well, I can distinguish between this with customerID, primary key attribute, and using the statusCustomer, the repeat customer is now known as a new customer.