Implementation:

viewHotels: This function requests from the user their current latitude and longitude. These two attributes become converted into a double since they're taken in as a string. We then query for a List of every hotel with their latitudes and longitudes. We then for loop to calculate the euclidean distance of every hotel in that generated List and check whether or not it is less than 30 and return the hotels that have a distance of less than 30.

viewRooms: We request a hotelID and a date from the user. We then query all available rooms for that hotel and that date. This also removes previously booked rooms from the query with the NOT IN keyword before the subquery that searches for booked rooms on the given date that the given hotelID.

bookRooms: We request from the user their userID, hotelID, roomNumber and date. We then query whether or not the room is already booked on that date and if it is not then we continue to the next query to insert their booking. After booking their room, the room price is returned to the user. The user is also given prompts about whether or not their booking was successful and if it was successful, they are also given the price for that room.

viewRecentBookingsfromCustomer: This function asks for the userID and queries for all the bookings under this userID and within the SELECT of the main query, we have a subquery that returns the price of the rooms that the user has booked. In the end, the user gets a result of their booking information along with the price of the room.

updateRoomInfo: When prompted, we check whether the user is a manager. If they are, we ask for the room number, the price to update, and the image url to update. We then do an update query where we update the tuple that the given info matches in the Rooms table. We then do an INSERT into RoomUpdatesLog to add a tuple that tells us about the update that we performed.

viewRecentUpdates: User is prompted for their manager info and hotelID. If they are confirmed as a manager, the 5 most recent room updates are returned from the given hotel of that manager in descending order by the updated date from RoomUpdatesLog.

viewBookingHistoryofHotel: For this, we check if the managerUserID is valid with the hotelID. If they are a manager, they are asked if they want a specific date range. If they say yes, they enter the date and all the records of bookings from that date range are displayed. If they say no then every record of all bookings of that hotel are displayed.If the managerID does not match, an error message is displayed.

viewRegularCustomers: This query asks the user for managerUserID and hotelID. The query searches for the customer that appears the most in the booking history of the given hotel. It returns the 5 most frequent matches in descending order.

placeRoomRepairRequests: This query requests the user's managerID and checks if it is valid. If the managerID exists then we continue and ask for all the required information to book a room repair. After that we insert all of the requested information into RoomRepairs and RoomRepairRequests.

viewRoomRepairHistory: The user is prompted to enter their manager userID. If the manager userID is found matching a managerID we prompt them for their hotel ID. We then select from RoomRepairs the information about the room repairs for that hotel.

Problems/Findings:

Prior to this project, I was not familiar with Java syntax since I had never used Java in any other courses before. After working on this project, I am now more familiar with Java and its syntax where I have basic fluency in Java. Also after working on this project I discovered more keywords within the PostgreSQL language that I had not previously used for other assignments in this course thus expanding my knowledge on PostgreSQL.