**CS157A Final Project Report**

**Project Title:** Hotel Reservation System

**Team members:** Frank Mock, Alex Preston, Joshua Urrea

**Team:** Team Alex-Frank-Joshua

**Database Schema**

Our database schema is shown below and is defined in a file named hotel.sql. Each table has an attribute that has ID included in it’s name. Attributes with ID in their name are keys. The primary key is, in most cases, the first tuple. The second tuple in most tables that include ID in their name are a foreign key.

The user of the program does not create the keys. They are automatically generated with insert operations. This prevents key constraint violations. The Occupied table has a foreign key named cID which references the Customer table cID. To prevent a foreign key violation when a customer is deleted, there is a trigger in place that automatically deletes the reference in the Occupied table when a customer is deleted from the Customer table.

**DROP** DATABASE **IF** **EXISTS** Hotel**;**

**CREATE** DATABASE Hotel**;**

**USE** Hotel**;**

**DROP** **TABLE** **IF** **EXISTS** Room**;**

**CREATE** **TABLE** Room

**(**

rID **INT** **PRIMARY** **Key** AUTO\_INCREMENT**,**

room\_type ENUM**(**'Basic'**,** 'Suite'**)DEFAULT** 'Basic' **NOT** **NULL,**

description **VARCHAR(**60**),**

price **DOUBLE** **NOT** **NULL**

**);**

**ALTER** **TABLE** Room AUTO\_INCREMENT **=** 101**;**

**DROP** **TABLE** **IF** **EXISTS** Customer**;**

**CREATE** **TABLE** Customer

**(**cID **INT** **PRIMARY** **Key** AUTO\_INCREMENT**,**

cNAME **VARCHAR(**40**),**

email **VARCHAR(**50**),**

phone **VARCHAR(**14**),**

num\_rooms **INT** **DEFAULT** 0**,**

updatedOn **TIMESTAMP** **NOT** **NULL** **ON** **UPDATE** **CURRENT\_TIMESTAMP,**

has\_spent **double** **DEFAULT** 0.00 **NOT** **NULL**

**);**

**DROP** **TABLE** **IF** **EXISTS** Occupied**;**

**CREATE** **TABLE** Occupied

**(**cID **INT,**

rID **INT,**

occupiedDate **TIMESTAMP** **NOT** **NULL** **DEFAULT** NOW**(),**

overStay **BOOLEAN** **DEFAULT** **FALSE,**

**PRIMARY** **KEY(**cID**,**rID**,**occupiedDate**),**

**FOREIGN** **KEY** **(**cID**)** **references** Customer **(**cID**)**

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE,**

**FOREIGN** **KEY** **(**rID**)** **references** Room**(**rID**)**

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**

**);**

**DROP** **TABLE** **IF** **EXISTS** VIP**;**

**CREATE** **TABLE** VIP

**(**cID **INT,**

notes **VARCHAR(**60**),**

**PRIMARY** **KEY(**cID**,**notes**),**

**FOREIGN** **KEY** **(**cID**)** **references** Customer**(**cID**)**

**);**

**DROP** **TABLE** **IF** **EXISTS** Ammenities**;**

**CREATE** **TABLE** Ammenities

**(**aID **INT** AUTO\_INCREMENT **NOT** **NULL,**

description **VARCHAR(**60**),**

price **DOUBLE,**

**PRIMARY** **KEY(**aID**)**

**);**

**DROP** **TABLE** **IF** **EXISTS** Ammenity\_orders**;**

**CREATE** **TABLE** Ammenity\_orders

**(**aID **INT,**

cID **INT,**

amount **INT,**

orderDate **TIMESTAMP** **NOT** **NULL** **DEFAULT** NOW**(),**

**PRIMARY** **KEY(**aID**,** cID**,** orderDate**),**

**FOREIGN** **KEY** **(**cID**)** **references** Customer**(**cID**)**

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**

**);**

**DROP** **TABLE** **IF** **EXISTS** Archive**;**

**CREATE** **TABLE** Archive

**(**archiveID **INT** **PRIMARY** **KEY** AUTO\_INCREMENT**,**

cID **INT,**

cNAME **VARCHAR(**40**),**

email **VARCHAR(**50**),**

phone **VARCHAR(**14**),**

num\_rooms **INT** **DEFAULT** 0**,**

updatedOn **TIMESTAMP** **NOT** **NULL,**

has\_spent **double** **DEFAULT** 0.00 **NOT** **NULL**

**);**

############### TRIGGERS ############################################

**DROP** **TRIGGER** **IF** **EXISTS** deleteCascadeTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** deleteCascadeTrigger

**AFTER** **DELETE** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**delete** **from** Occupied **where** cID **=** **OLD.**cID**;**

**delete** **from** VIP **WHERE** cID **=** **OLD.**cID**;**

**END;//**

delimiter **;**

**DROP** **TRIGGER** **IF** **EXISTS** insertVIPTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** insertVIPTrigger

**AFTER** **INSERT** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **NEW.**has\_spent **>** 3000.0 **THEN**

**INSERT** **INTO** VIP **(**cID**,** notes**)** **VALUES** **(NEW.**cID**,** ""**);**

**INSERT** **INTO** Ammenity\_orders**(**aID**,** cID**,** amount**)** **values(**1**,** **NEW.**cID**,** 1**);**

**END** **IF;**

**END;//**

delimiter **;**

**DROP** **TRIGGER** **IF** **EXISTS** updateVIPTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** updateVIPTrigger

**AFTER** **UPDATE** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **NEW.**has\_spent **>** 3000.0 **THEN**

**INSERT** **INTO** VIP **(**cID**,** notes**)** **VALUES** **(NEW.**cID**,** ""**);**

**END** **IF;**

**END;//**

delimiter **;**

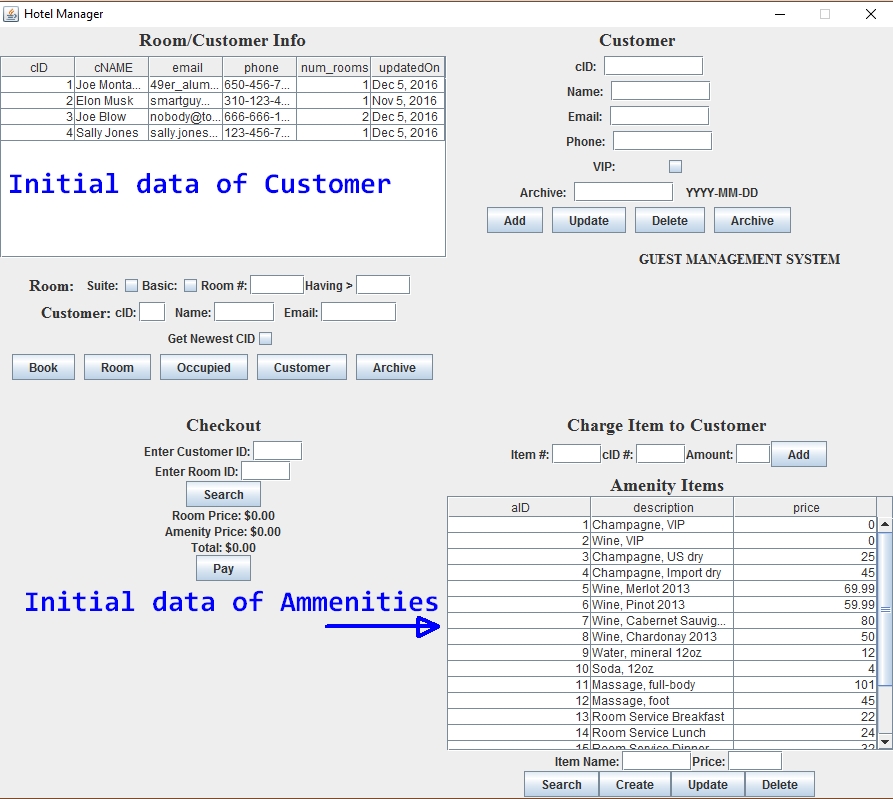
LOAD **DATA** **LOCAL** INFILE 'c:\\SQL\\rooms.txt' **INTO** **TABLE** Room**;**

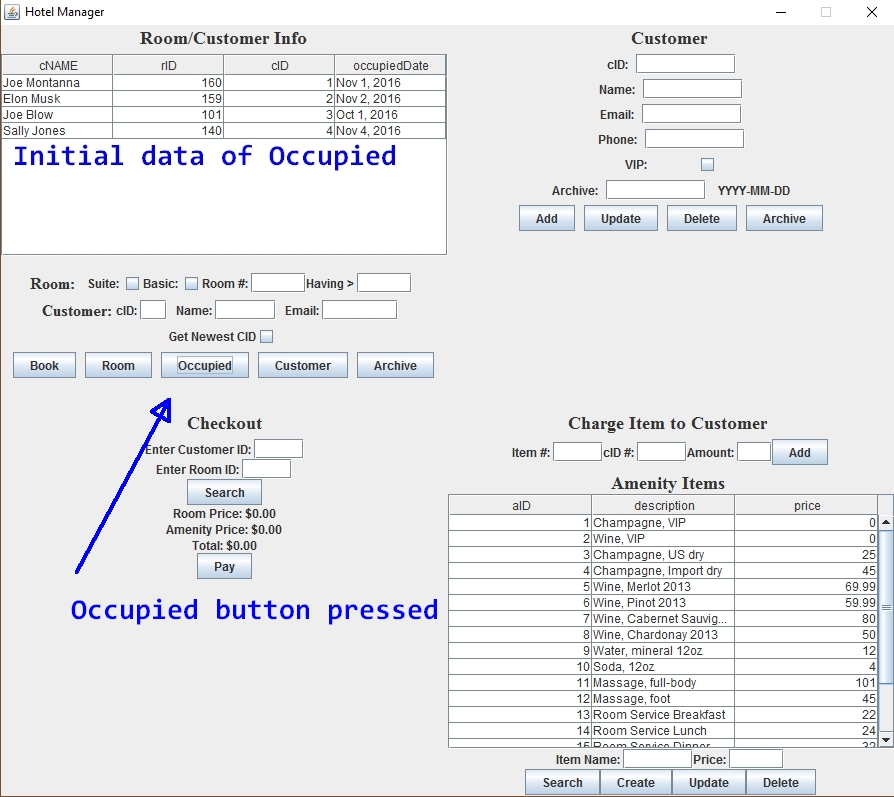
LOAD **DATA** **LOCAL** INFILE 'c:\\SQL\\customers.txt' **INTO** **TABLE** Customer**;**

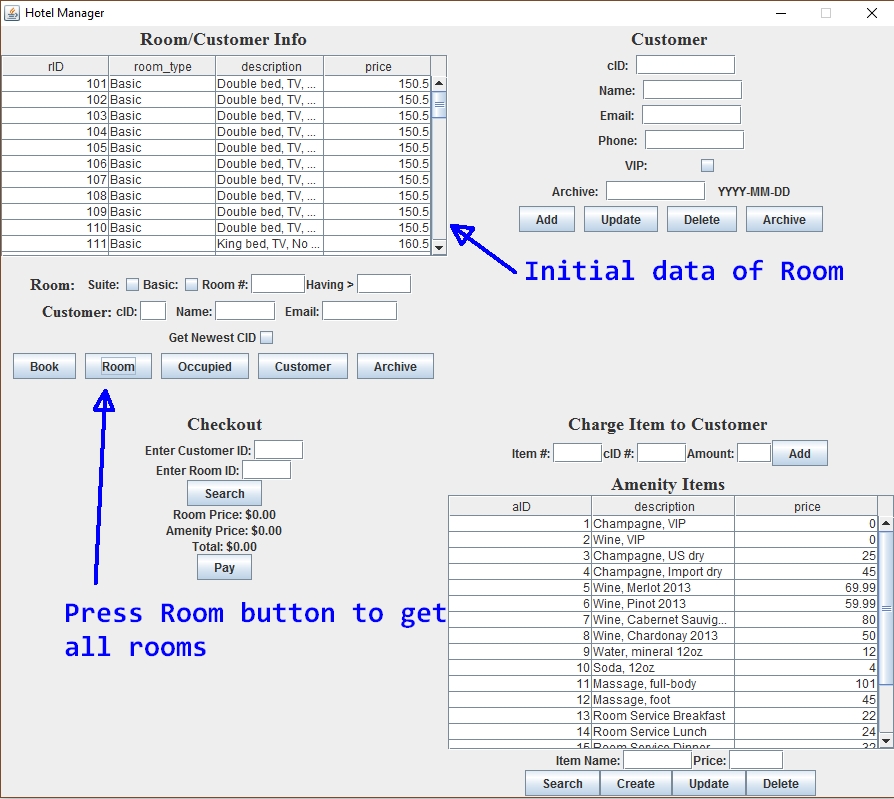
LOAD **DATA** **LOCAL** INFILE 'c:\\SQL\\occupied.txt' **INTO** **TABLE** Occupied**;**

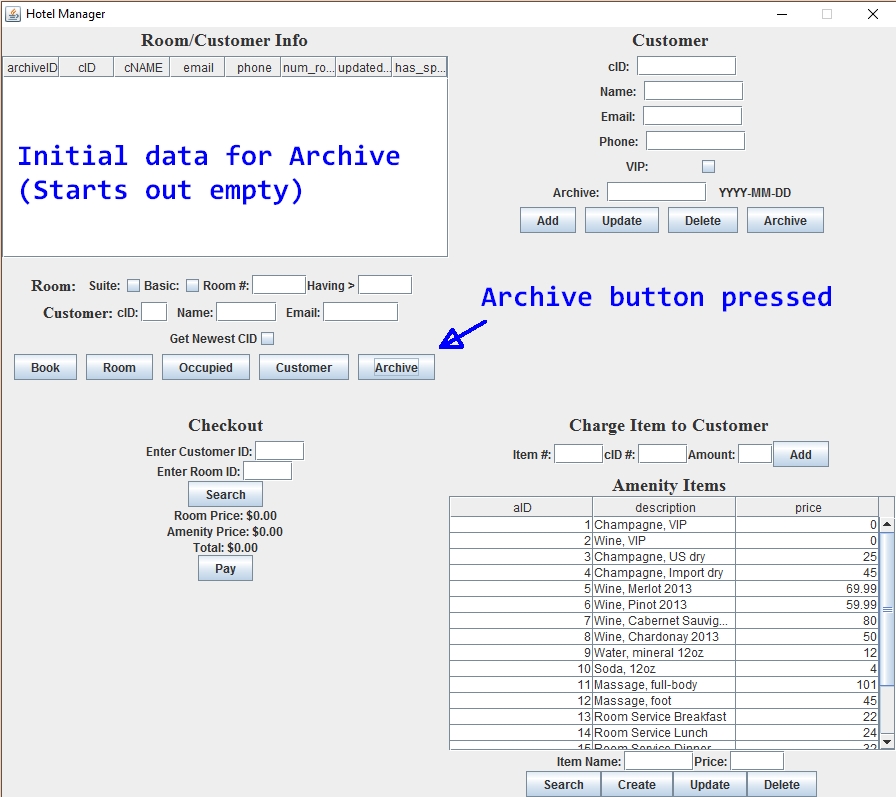
LOAD **DATA** **LOCAL** INFILE 'c:\\SQL\\ammenities.txt' **INTO** **TABLE** Ammenities**;**

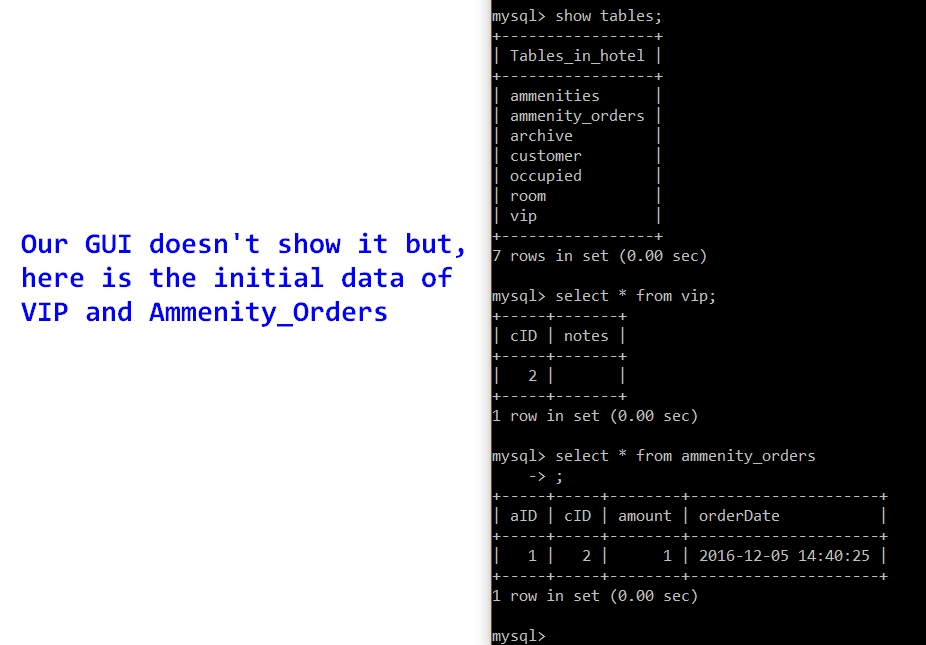
**Initial table data**











**15 Distinct User Functions**

1. View all rooms

2. View all rooms that are a suite

3. View room given a room number

4. View all customers

5. View all occupied rooms

6. View occupied rooms based on room number

7. View all archived

8. View all amenities

9. Insert new customer

10. Update a customer

11. Delete a customer

12. Archive customer base on a date

13. Book a room for a customer

14. Select customers having greater than n rooms

15. Get the newest cID entered

**All SQL SELECT statements**

All queries in the form of prepared statements are in the file named HotelQueries.java. The following are the SELECT queries from the prepared statements.

**SELECT** **\*** **FROM** archive**;**

**SELECT** cID**,** cNAME**,** email**,** phone**,** num\_rooms**,** updatedOn **FROM** customer;

**SELECT** **\*** **FROM** customer **WHERE** cName **=** **?;**

**SELECT** **\*** **FROM** room

**SELECT** **\*** **FROM** room **WHERE** rID**=** **?**

**Select** **distinct** r1**.**rID**,** r1**.**room\_type**,** r1**.**description**,** r1**.**price "

+ "**from** room r1**,** room r2 **where** r1**.**rID **=** r2**.**rID "

+ "**and** r1**.**room\_type **=** 'Suite'"

**SELECT** **\*** **FROM** ammenities

**select** cID**,** rID **as** Room**,** cName **as** Name**,** occupiedDate "

+ "**from** occupied **natural** **join** customer "

+ "**where** cID **in(select** cID **from** occupied **group** **by** cID **having** **count(\*)** **>** **?)**"

**select** cName**,** rID**,** cID**,** occupiedDate "

+ "**from** customer **natural** **join** occupied

**select** **max(**cID**)** **from** customer

**All SQL UPDATE statements**

All queries are in the form of prepared statements in the file named HotelQueries.java. The following are the UPDATE queries from the prepared statements.

**Update** customer **SET** cNAME**=** **?,** email**=** **?,** phone**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** cNAME**=** **?,** email**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** cNAME**=** **?,** phone**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** email**=** **?,** phone**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** cNAME**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** email**=** **?** **WHERE** cID **=** **?**

**Update** customer **SET** phone**=** **?** **WHERE** cID **=** **?**

**All SQL DELETE statements**

All queries are in the form of prepared statements in the file named HotelQueries.java. The following are the DELETE queries from the prepared statements.

**DELETE** **FROM** customer **WHERE** cID **=** **?**

**delete** **from** customer **where** updatedOn **<** **?**

**delete** **from** vip **where** cID **in(select** cID **from** customer **where** updatedOn **<** **?)**

**All SQL INSERT statements**

All queries are in the form of prepared statements in the file named HotelQueries.java. The following are the INSERT queries from the prepared statements.

**insert** **into** archive **(**cID**,** cName**,** email**,** phone**,** num\_rooms**,** updatedOn**,** has\_spent**)** "

+ "**select** **\*** **from** customer **where** updatedOn **<** **?**"

**INSERT** **INTO** customer **(**cName**,** email**,** phone**,** updatedOn**)** **VALUES** **(?,** **?,** **?,** NOW**()),** **Statement.**RETURN\_GENERATED\_KEYS

**INSERT** **INTO** occupied **(**cID**,** rID**)** **VALUES** **(?,** **?)**

**All SQL TRIGGER statements**

The triggers are defined with the database schema in hotel.sql. Here are the triggers defined in that file. As mentioned above, there is a trigger in place to prevent a foreign key violation if a tuple in the Occupied table is referring to a cID in the Customer table and a customer is deleted. The trigger is supposed to delete any references in the VIP table also, but for some reason it only deletes the reference in the occupied table. It was necessary to write a prepared statement query to prevent a foreign key violation with regards to the VIP table. There is also another trigger in place that inserts customers into the VIP table if the customer has spent greater than $3000. There is a corresponding trigger for update.

**DROP** **TRIGGER** **IF** **EXISTS** deleteCascadeTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** deleteCascadeTrigger

**AFTER** **DELETE** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**delete** **from** Occupied **where** cID **=** **OLD.**cID**;**

**delete** **from** VIP **WHERE** cID **=** **OLD.**cID**;**

**END;//**

delimiter **;**

**DROP** **TRIGGER** **IF** **EXISTS** insertVIPTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** insertVIPTrigger

**AFTER** **INSERT** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **NEW.**has\_spent **>** 3000.0 **THEN**

**INSERT** **INTO** VIP **(**cID**,** notes**)** **VALUES** **(NEW.**cID**,** ""**);**

**INSERT** **INTO** Ammenity\_orders**(**aID**,** cID**,** amount**)** **values(**1**,** **NEW.**cID**,** 1**);**

**END** **IF;**

**END;//**

delimiter **;**

**DROP** **TRIGGER** **IF** **EXISTS** updateVIPTrigger**;**

delimiter **//**

**CREATE** **TRIGGER** updateVIPTrigger

**AFTER** **UPDATE** **ON** Customer

**FOR** **EACH** **ROW**

**BEGIN**

**IF** **NEW.**has\_spent **>** 3000.0 **THEN**

**INSERT** **INTO** VIP **(**cID**,** notes**)** **VALUES** **(NEW.**cID**,** ""**);**

**END** **IF;**

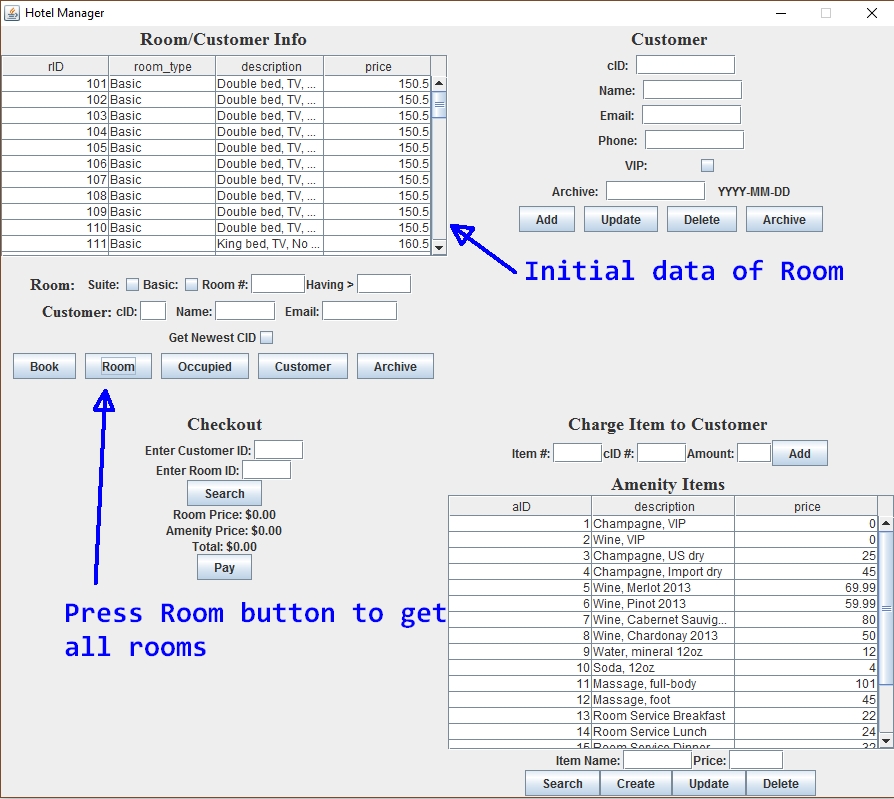
**END;//**

delimiter **;**

**Screen Shots Showing Each of the 15 Functions**

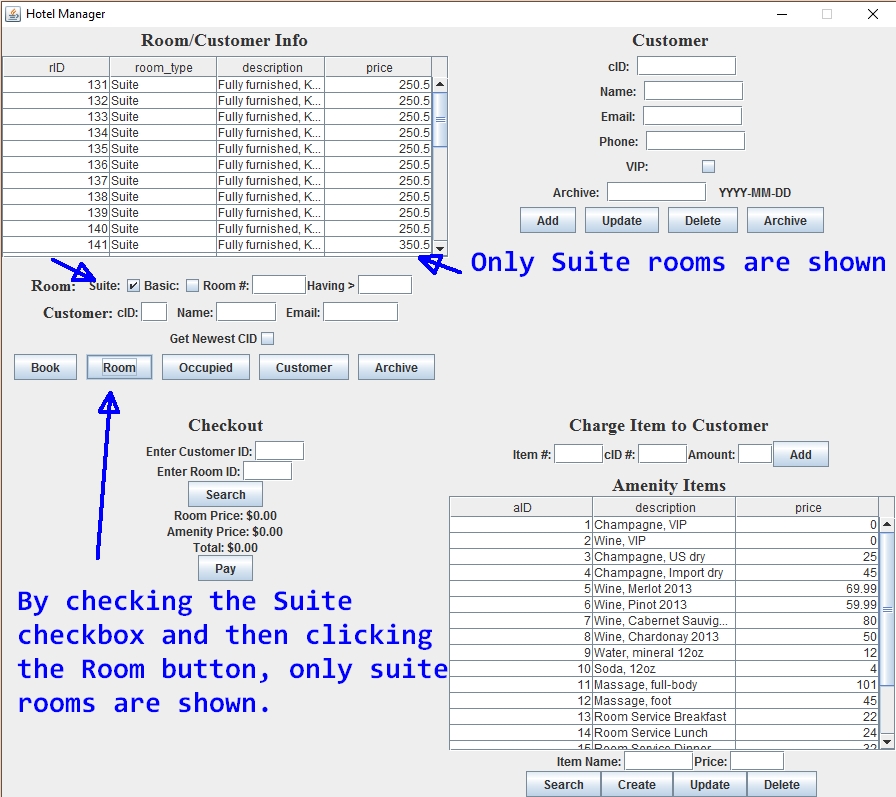
1. **View all Rooms**

By clicking the Room button, all the rooms in the room relation are shown.



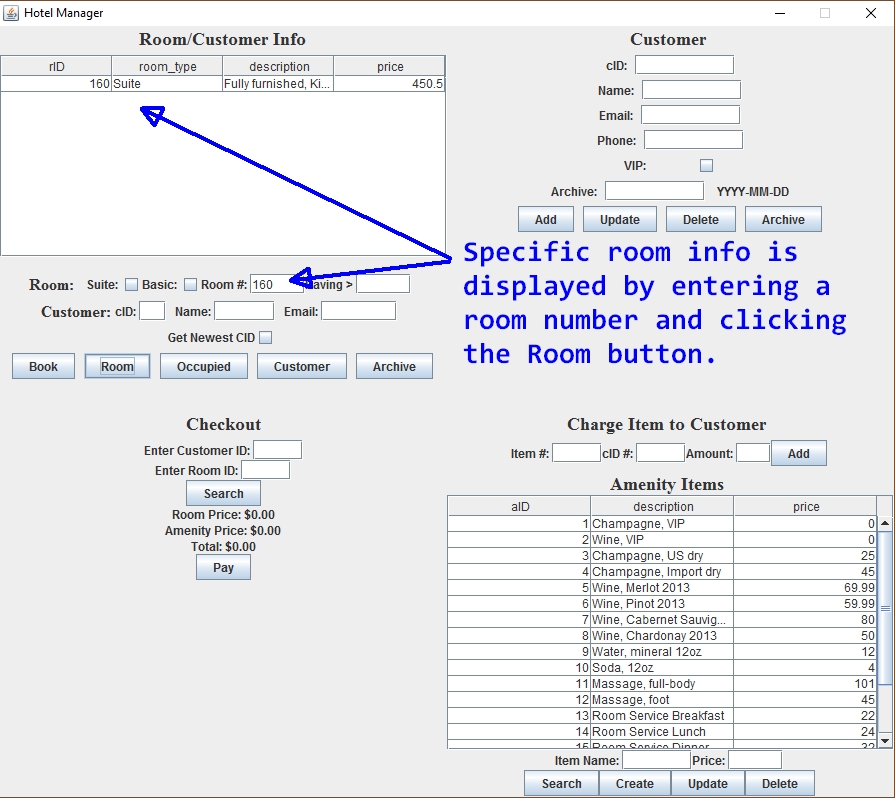
1. **View all Rooms that are suites**

By clicking the Room button and selecting the checkbox for suite, all the rooms in the room relation that are suites are shown.



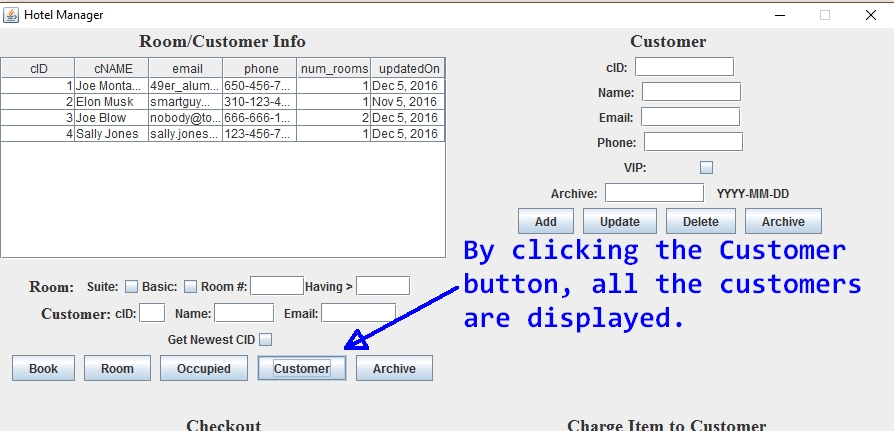
1. **View specific Room by room number**

By entering a room number and clicking the Room button the details of a specific room are displayed.



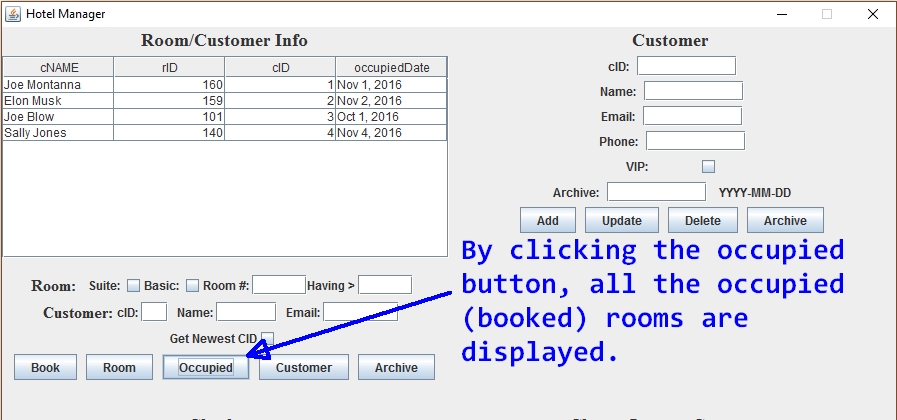
1. **View all Customers**

By Clicking the Customer Button, all the customers are displayed.



1. **View all occupied rooms**

By Clicking the Occupied Button, all the occupied (booked) rooms are displayed.



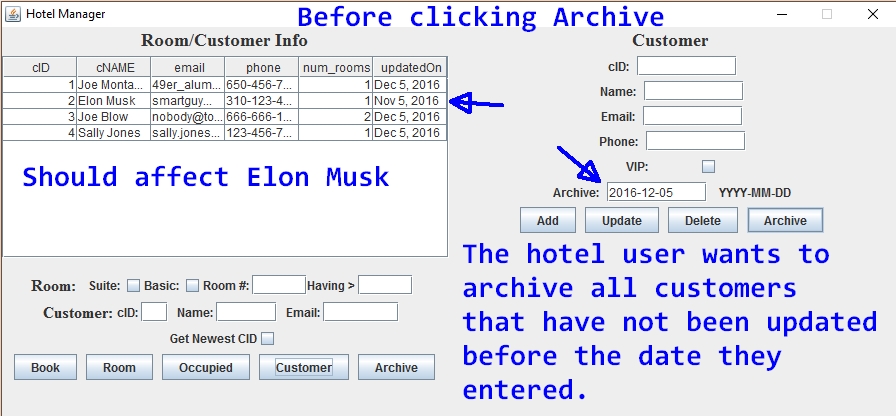
1. **View occupied room by entering a room number.**

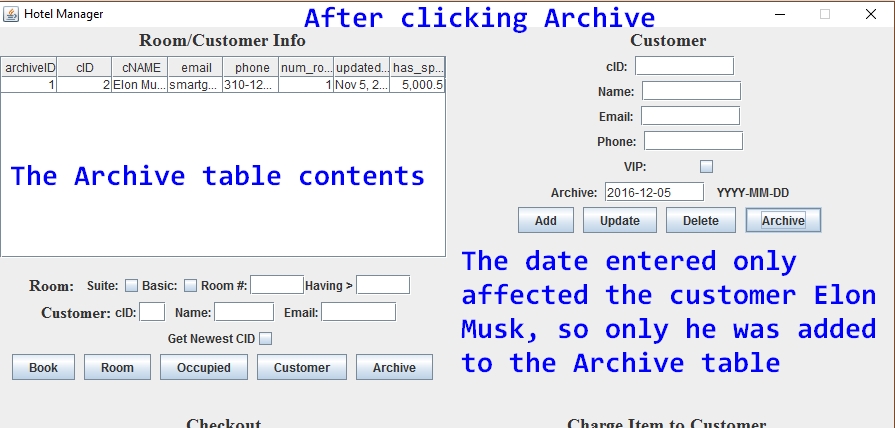
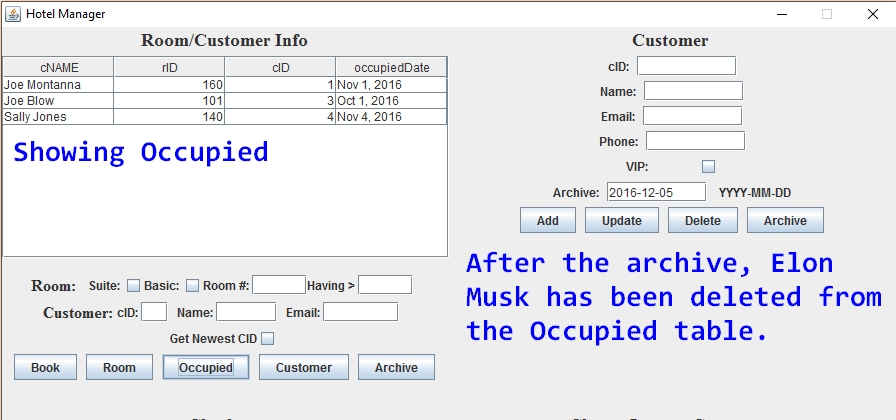
A hotel user can find out who is occupying a room by entering a room number and clicking the Occupied button.

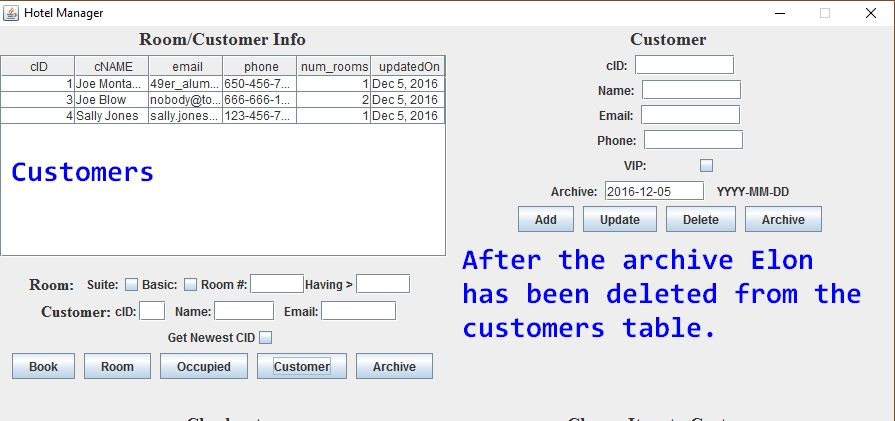


1. **Archive all customers that have not been updated before a date.**

By entering a date all customers that have not been updated before that date are archived. They are automatically deleted from the customer table and any table that references their cID. This prevents dangling pointers.

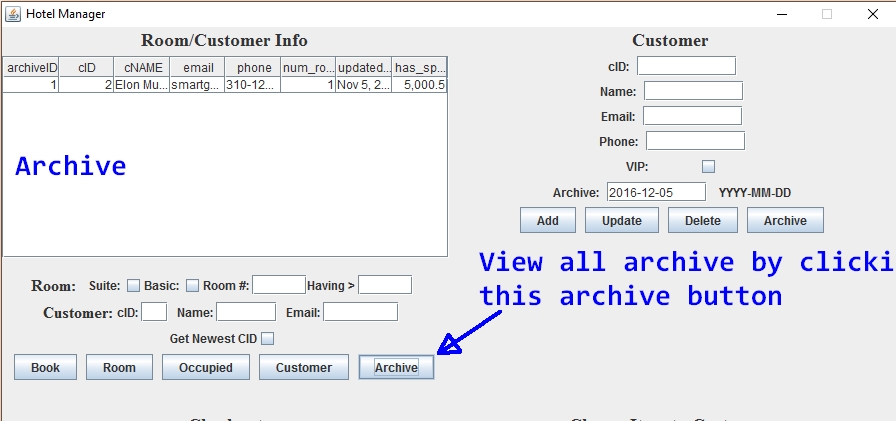




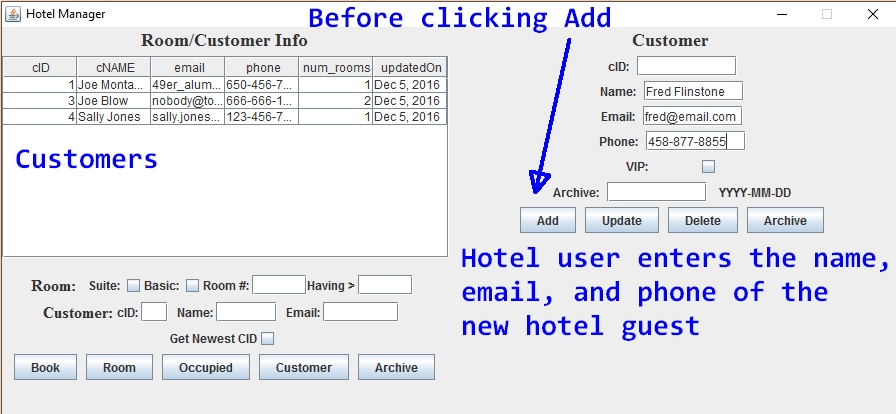
1. **View all archived**

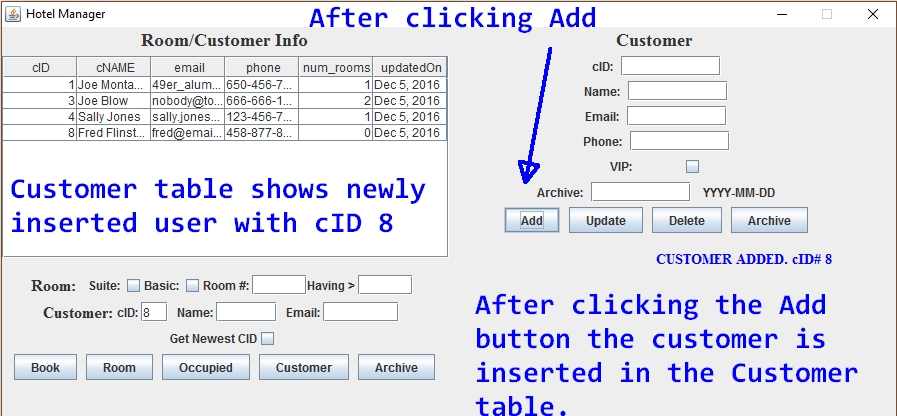
Click the left side Archive button to view all archive



1. **Insert new Customer**

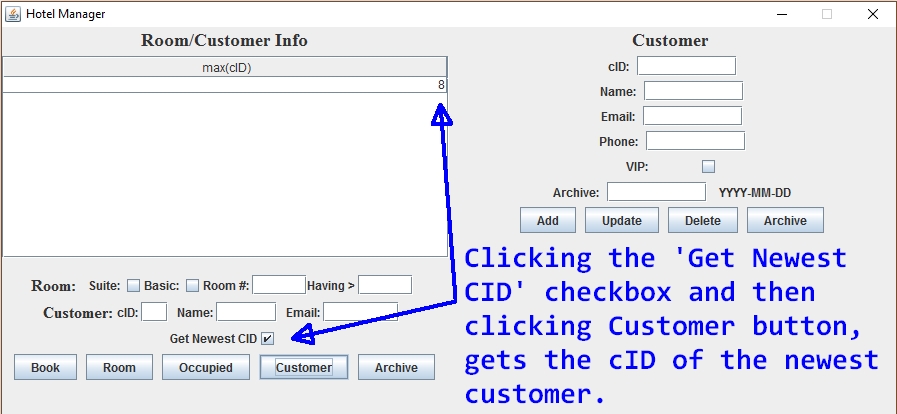
Hotel user enters the name, email, and phone of the new customer. Then by clicking the Add button the person is inserted in the Customer table.



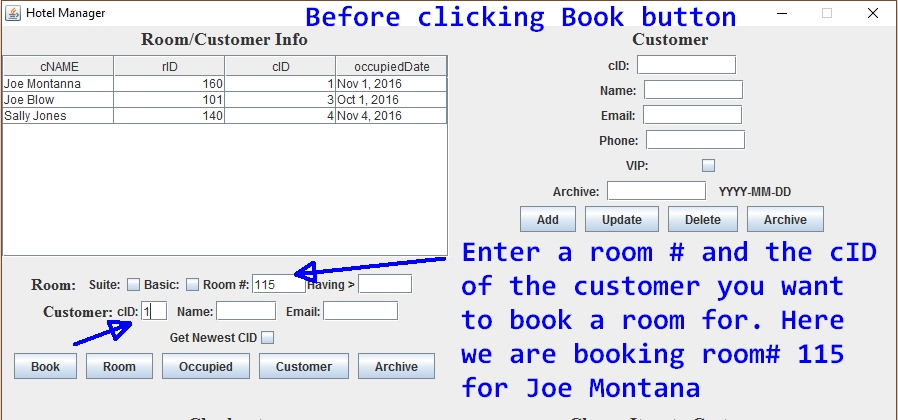


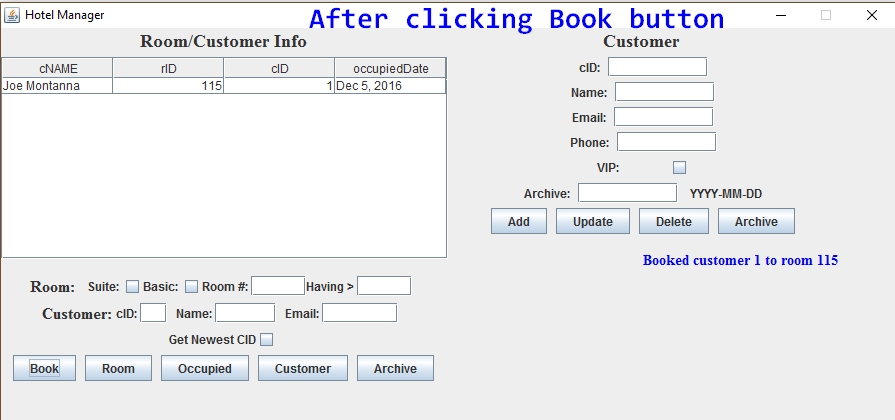
1. **Get cID of most recently added customer**

Select the ‘Get the Newest CID’ checkbox and click the Customer button



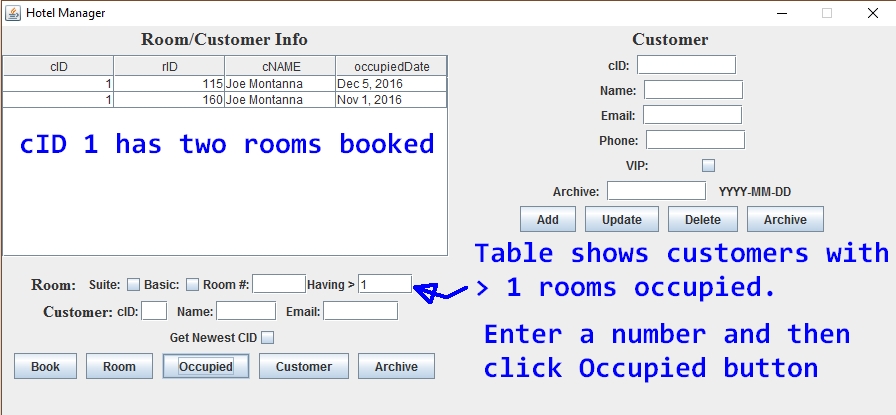
1. **Book a room for a customer**





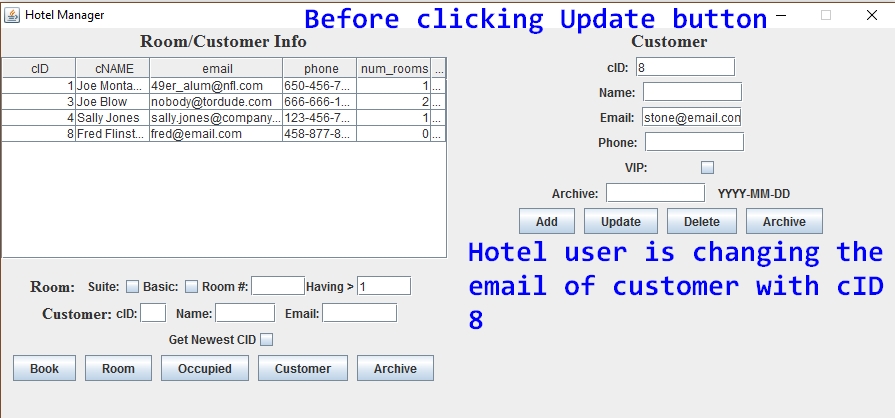
1. **Show customers having greater than n rooms**

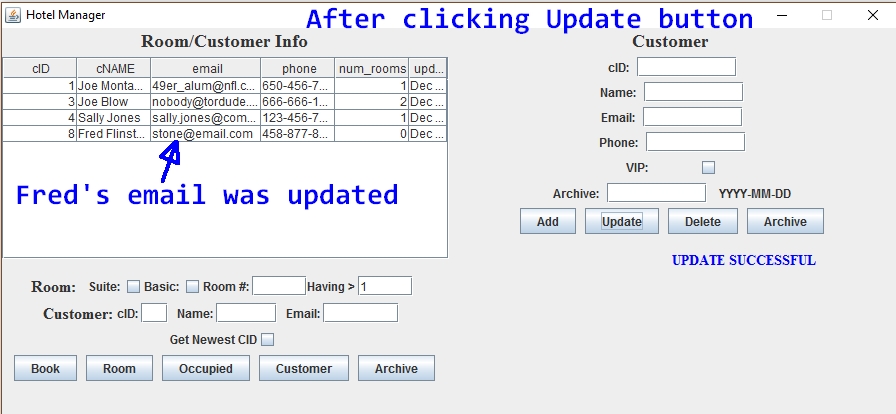
By entering a number in the ‘Having >’ textbox and clicking the occupied button, The table will show customers that have more than n number of rooms booked.



1. **Update Customer info**

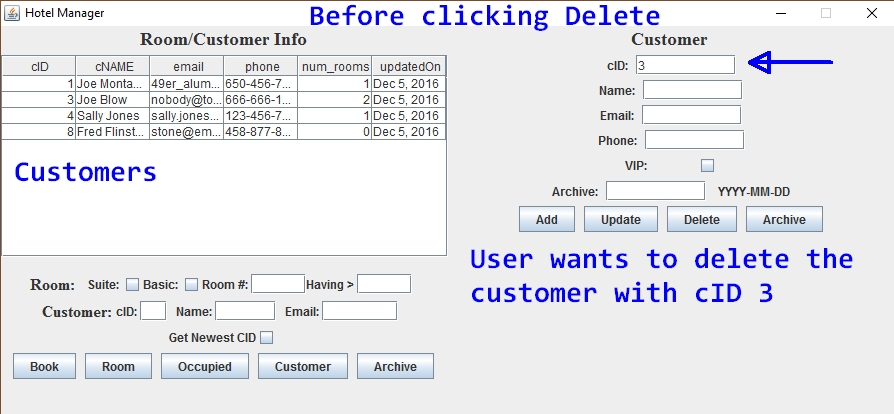
By entering either a name, email, and phone (or any combination of these) and a customer ID (cID) and then clicking Update – that users information is updated.

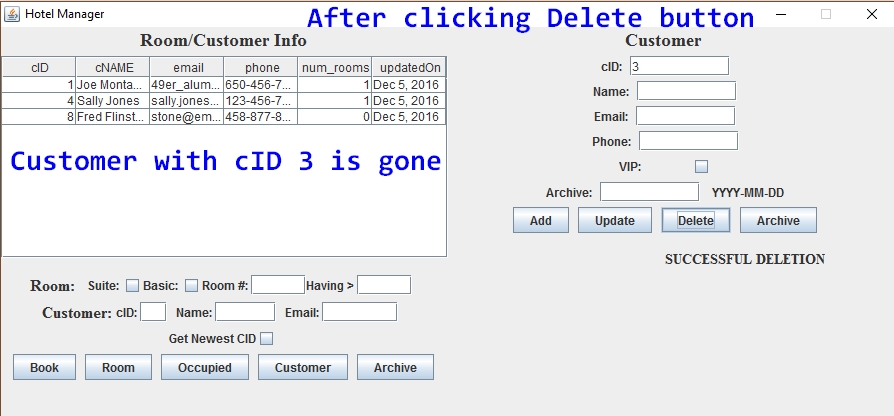




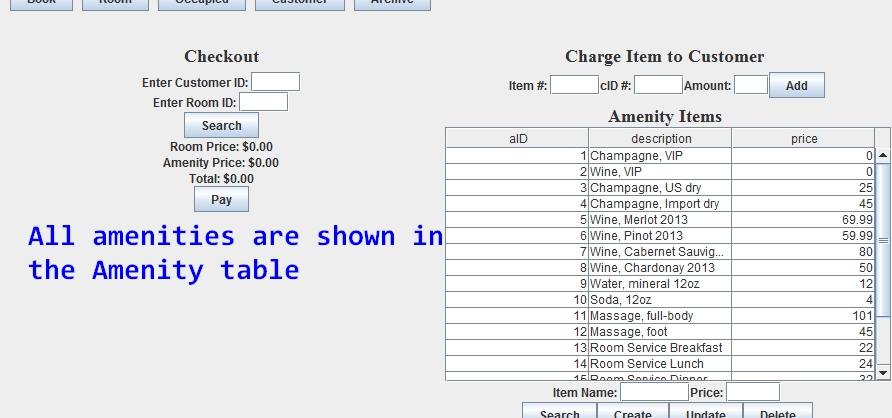
1. **Delete a Customer**

By entering a customer ID (cID) and clicking Delete a Customer is deleted. There are triggers and prepared statements in place to prevent referential integrity constrain errors.





1. **View all Amenities**



**Final notes**

The trigger defined in the schema was not sufficient to mitigate referential integrity problems when deleting customers. This is because even though the trigger deleted from the Occupied table when a customer was deleted from the Customer table, it did not delete from the VIP table when the customer was referenced there. To ensure that this did happen, it was necessary to execute a prepared statement that first deleted the customer from VIP table. Then a customer deletion, by the archive feature for example, could delete from customer and the trigger would take care of any references in the Occupied table.