Assignment 1

Jeffrey Middendorf – 27338096 || Abdullah Azad – 27565580 || Hermont - 27360261

Monash University – Faculty of Information Technology

FIT5148 – Distributed and big data processing

Heaven Hu

Wednesday, 5 April 2017

# 

**Table of Contents**

[Statement of Work 3](#_Toc479151367)

[1) Task 1 – Database 4](#_Toc479151368)

[a) Table Creation 4](#_Toc479151369)

[i) Hotel@FIT5148A 4](#_Toc479151370)

[ii) Room@FIT5148B 5](#_Toc479151371)

[iii) RoomFacility@FIT5148B 6](#_Toc479151372)

[iv) Customer@FIT5148B 7](#_Toc479151373)

[v) Guest@FIT5148B 8](#_Toc479151374)

[vi) Membership@FIT5148B 9](#_Toc479151375)

[vii) Booking@FIT5148B 10](#_Toc479151376)

[(1) Creation 10](#_Toc479151377)

[(1) Occupancy Constraint 11](#_Toc479151378)

[(2) Display Automatic Discount 12](#_Toc479151379)

[ii) Payment@FIT5148B 14](#_Toc479151380)

[(1) Creation 14](#_Toc479151381)

[(2) Payment Status 15](#_Toc479151382)

[b) Table Implementation 16](#_Toc479151383)

[c) ER Diagram 17](#_Toc479151384)

[d) Index Creation 19](#_Toc479151385)

[e) Populate Data 19](#_Toc479151386)

[2) Task 2 – Oracle Stored Procedure 22](#_Toc479151387)

[3) Task 3 – Triggers 24](#_Toc479151388)

[4) Task 4 – Fragmentation 26](#_Toc479151389)

[a) Horizontal Fragmentation 26](#_Toc479151390)

[b) Vertical Fragmentation 26](#_Toc479151391)

[c) Implementation 28](#_Toc479151392)

[i) Horizontal Fragmentation 28](#_Toc479151393)

[ii) Vertical Fragmentation 29](#_Toc479151394)

[5) Task 5 – User Interface 30](#_Toc479151395)

[a) Main Screen 32](#_Toc479151396)

[b) Hotel Frame 33](#_Toc479151397)

[c) Room Frame 34](#_Toc479151398)

[d) Customer Frame 35](#_Toc479151399)

[e) Guest Frame 36](#_Toc479151400)

[f) Membership Frame 37](#_Toc479151401)

[g) Booking Frame 38](#_Toc479151402)

[h) Payment Frame 39](#_Toc479151403)

[6) References 44](#_Toc479151404)

# **Statement of Work**

**Jeff**

* Task 1 – Final Version
* Task 3 – 1/2
* Task 4 - Horizontal
* Task 5
  + Main Menu
  + Hotel Frame
  + Room Frame
  + Membership Frame
  + Booking Frame
* This report

**Abdullah**

* Task 1 – Initial Version
* Task 4 – Vertical
* Task 5
  + Customer Frame
  + Payment Frame

**Hemant**

* Task 2
* Task 3
* Task 5
  + Guest Frame
  + Payment Frame

# **Task 1 – Database**

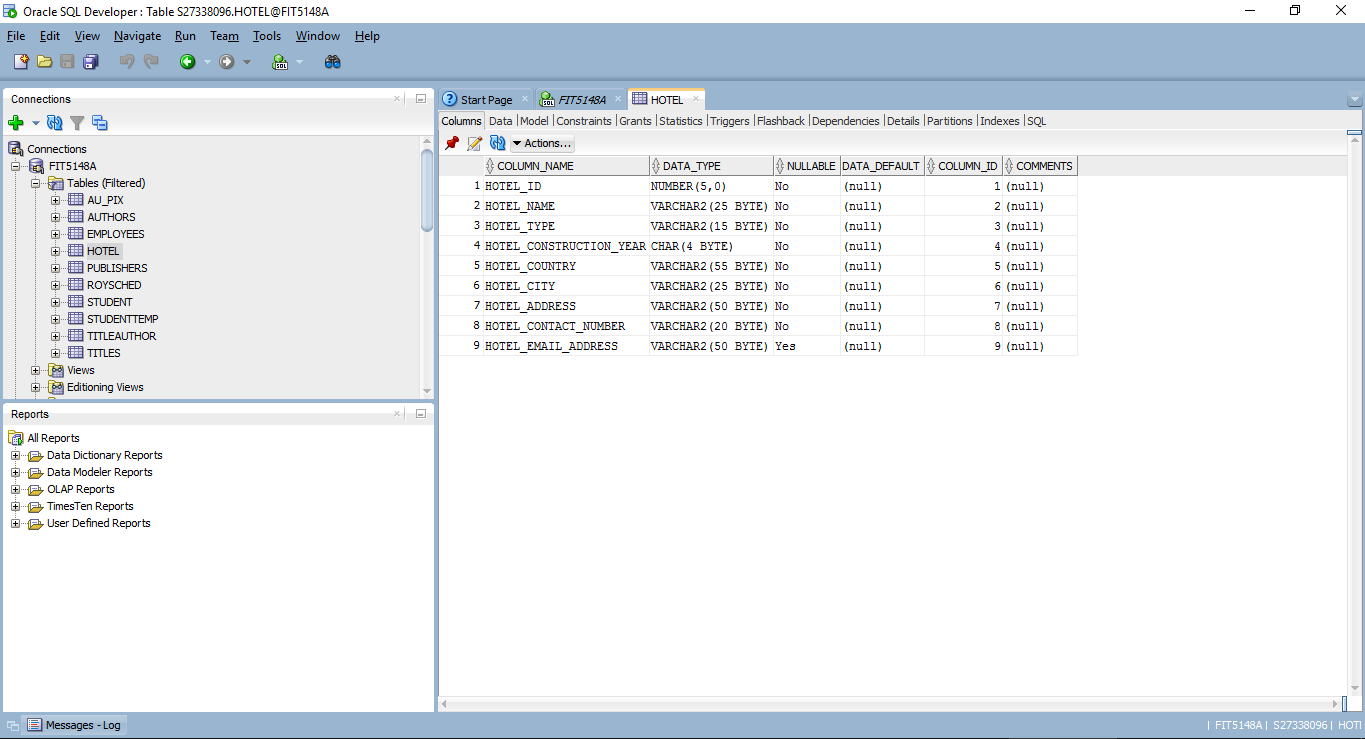
This system manages a hotel broker system for booking rooms. Please note that the creation scripts and the foreign key scripts are separated for readability. So, all the create scripts are executed and then the constraints and sequences are executed. There are also two associative (bridge) table that were not included in the specifications to handle many-to-many relationships between the tables.

# Table Creation

# *Hotel@FIT5148A*

The hotel table is the table that stores all the system’s hotel information. This hotel\_id was selected as the primary key.

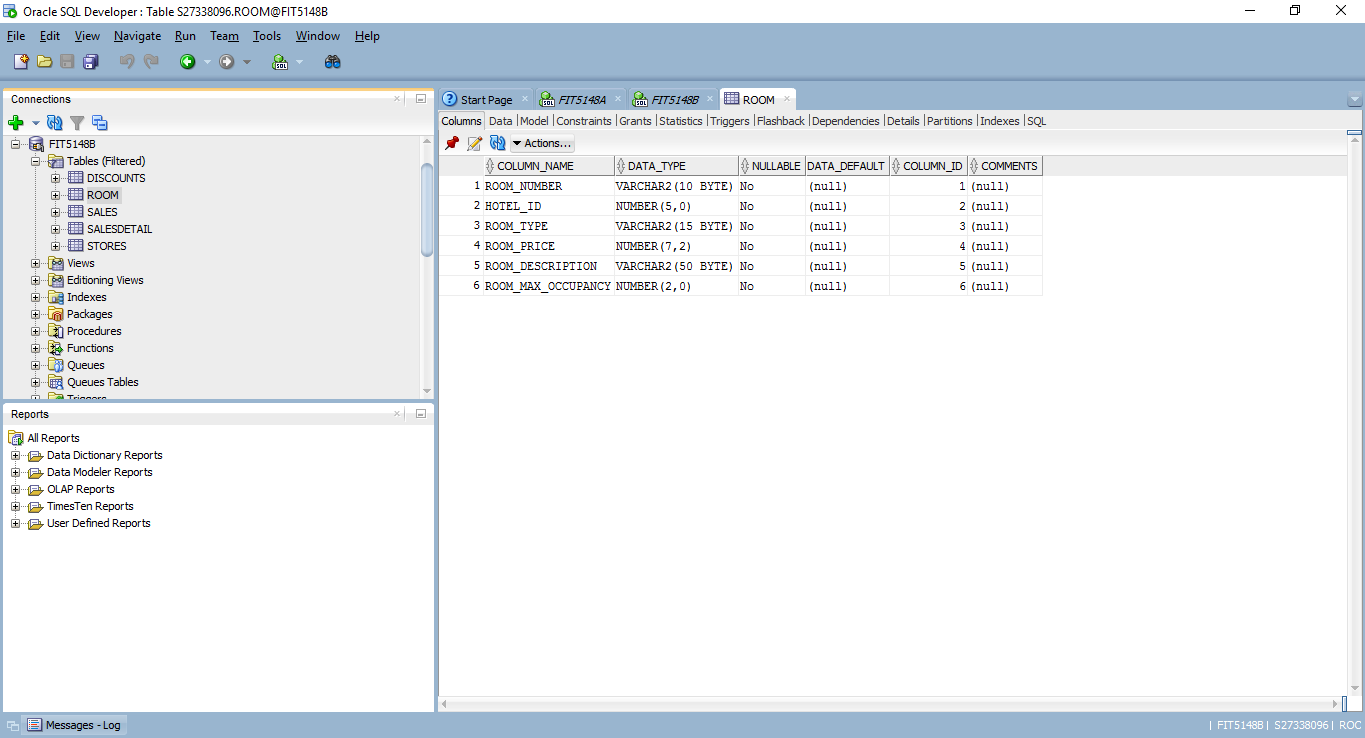




# *Room@FIT5148B*

This table stores all the room data for the system. It will relate back to the hotel table based on hotel\_id. Room number was chosen as the primary key.

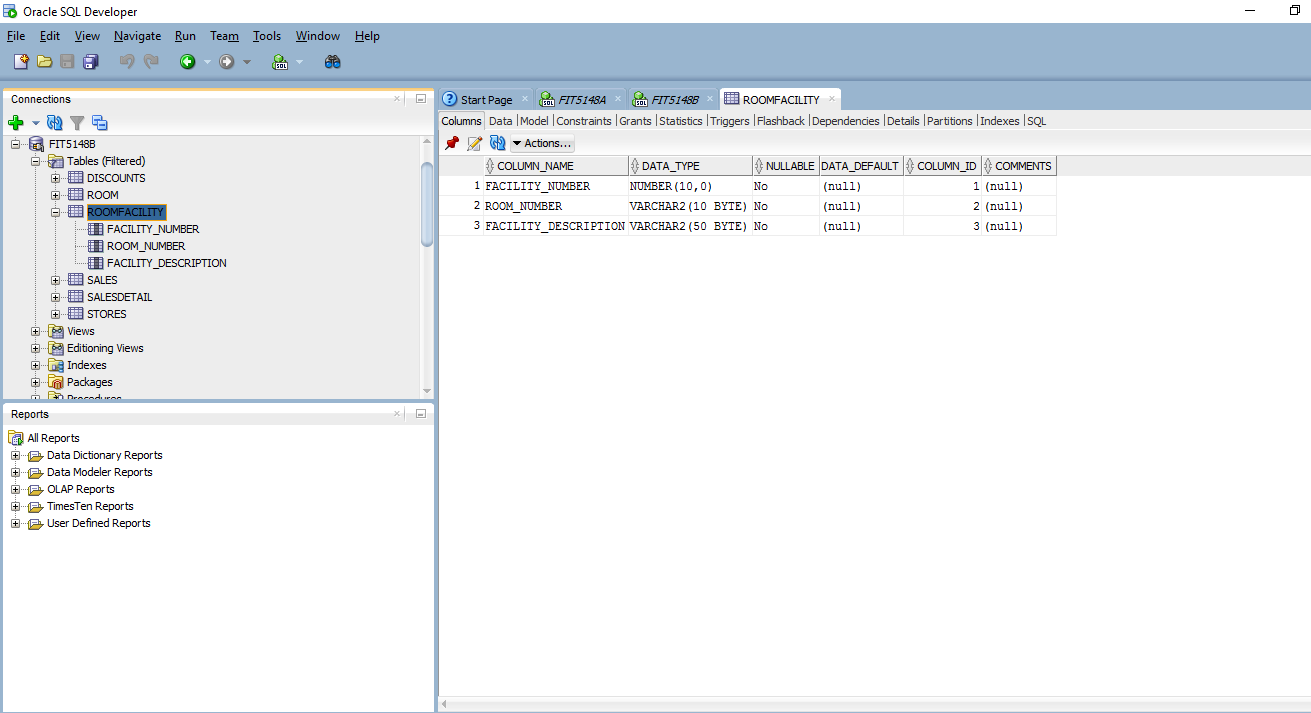




# *RoomFacility@FIT5148B*

This table store information about the facilities associated with each room. This relationship is established by creating a foreign key on room\_id.

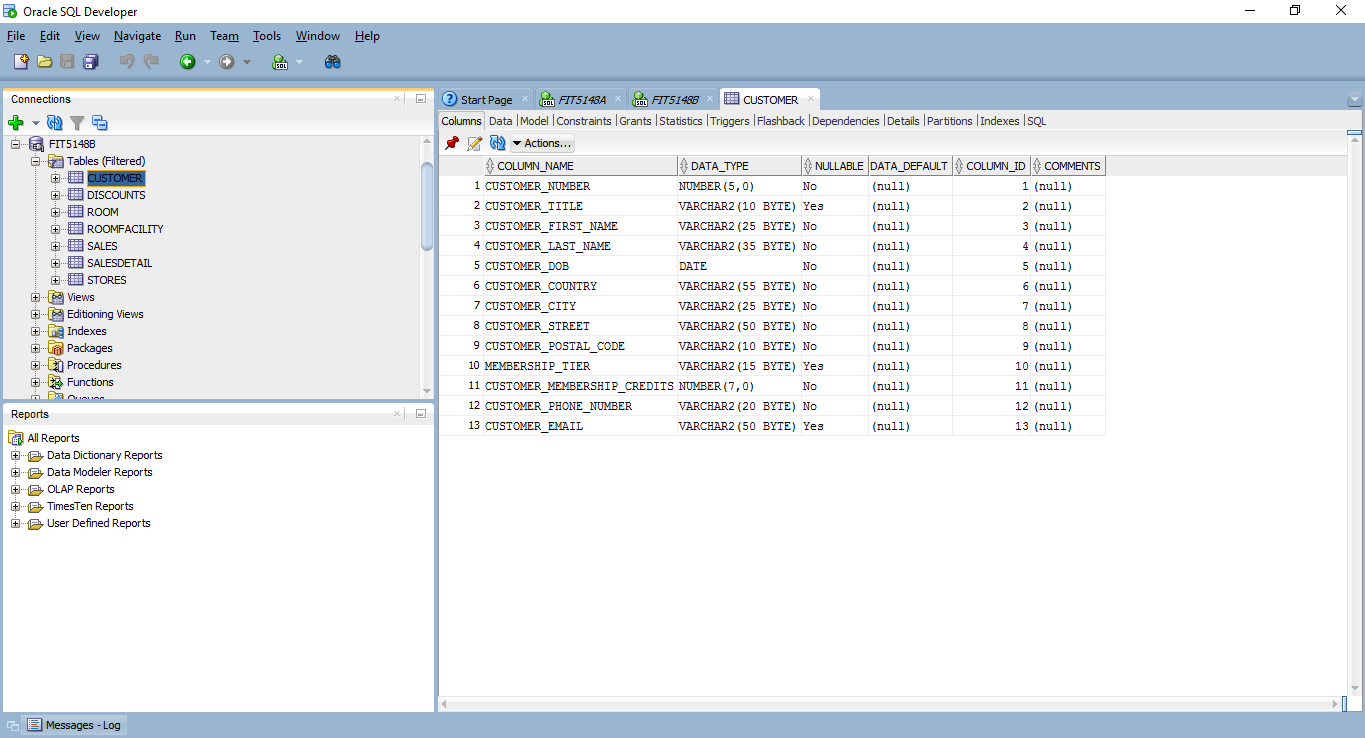




# *Customer@FIT5148B*

This table stores all the customer data on the HBS application. Customer\_number was chosen as the primary key. Membership\_tier is a foreign key referencing the membership attribute in the membership relation.

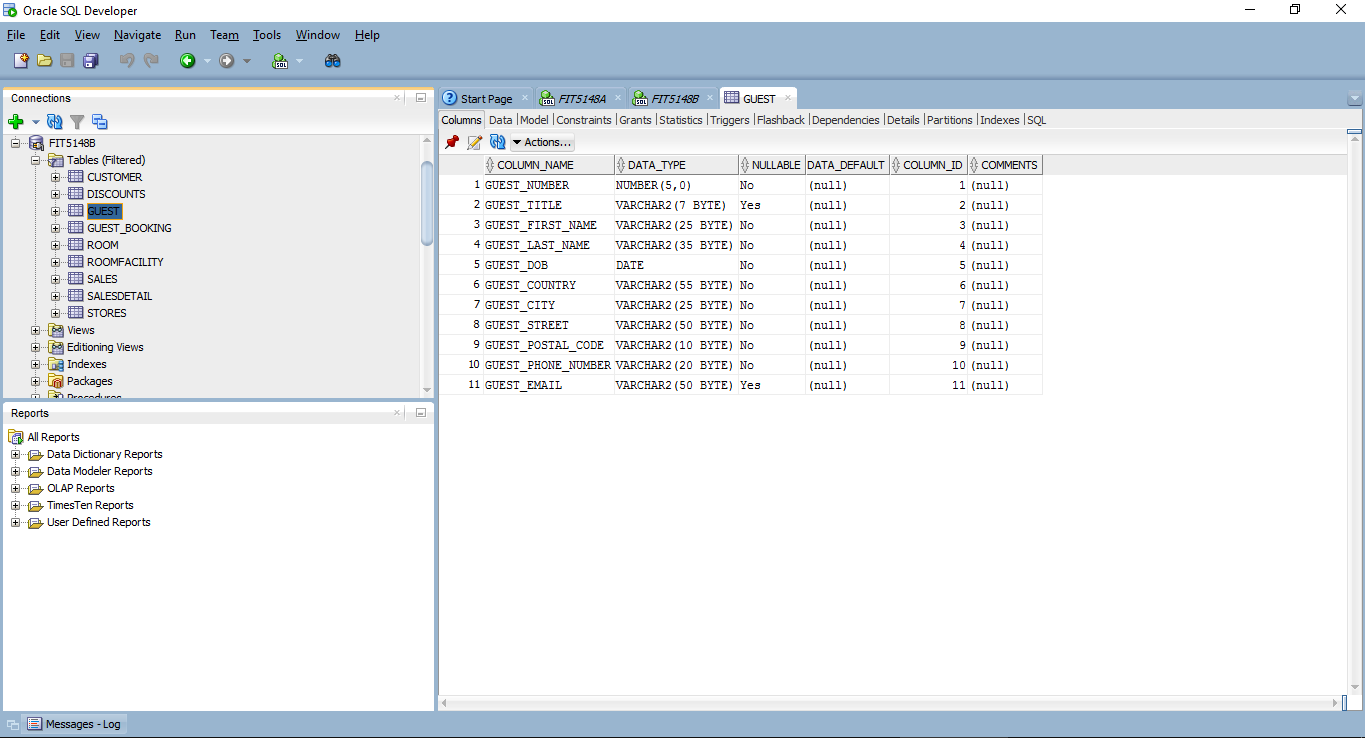




# *Guest@FIT5148B*

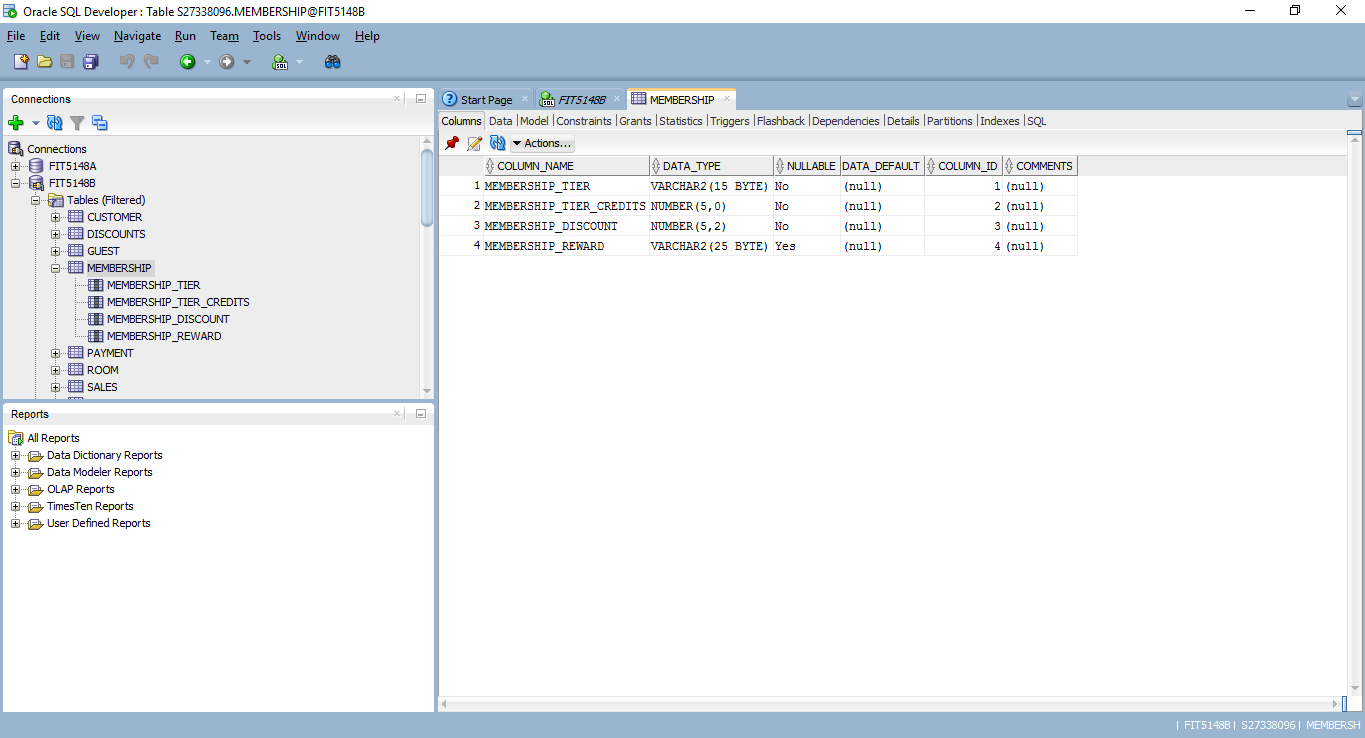
This table records all the guest data of the people who stay in the hotel. The guest may be different than the customer. Since one guest can have many booking and a single booking can have many guests, this creates a many-to many scenario. This will be handled by an associative table in the booking section of this report. Guest number was chosen as the primary key.





# *Membership@FIT5148B*





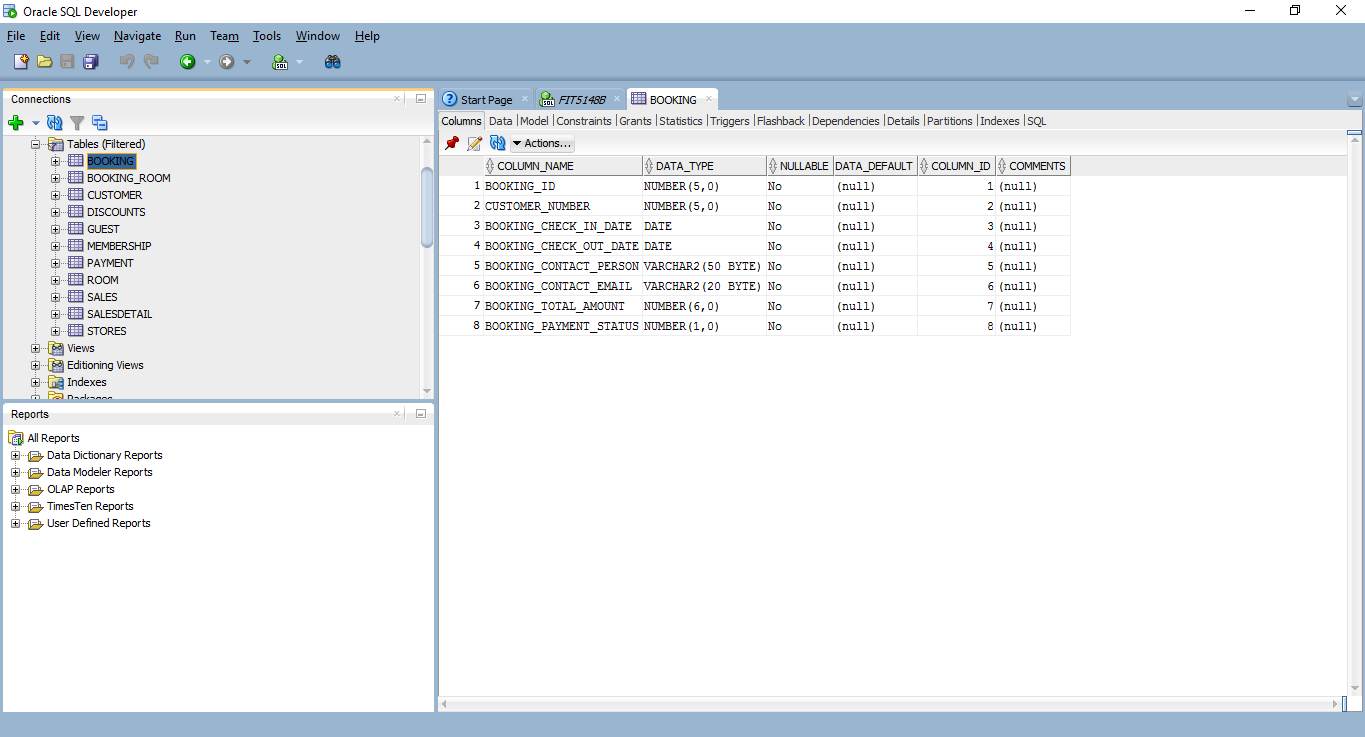
# *Booking@FIT5148B*

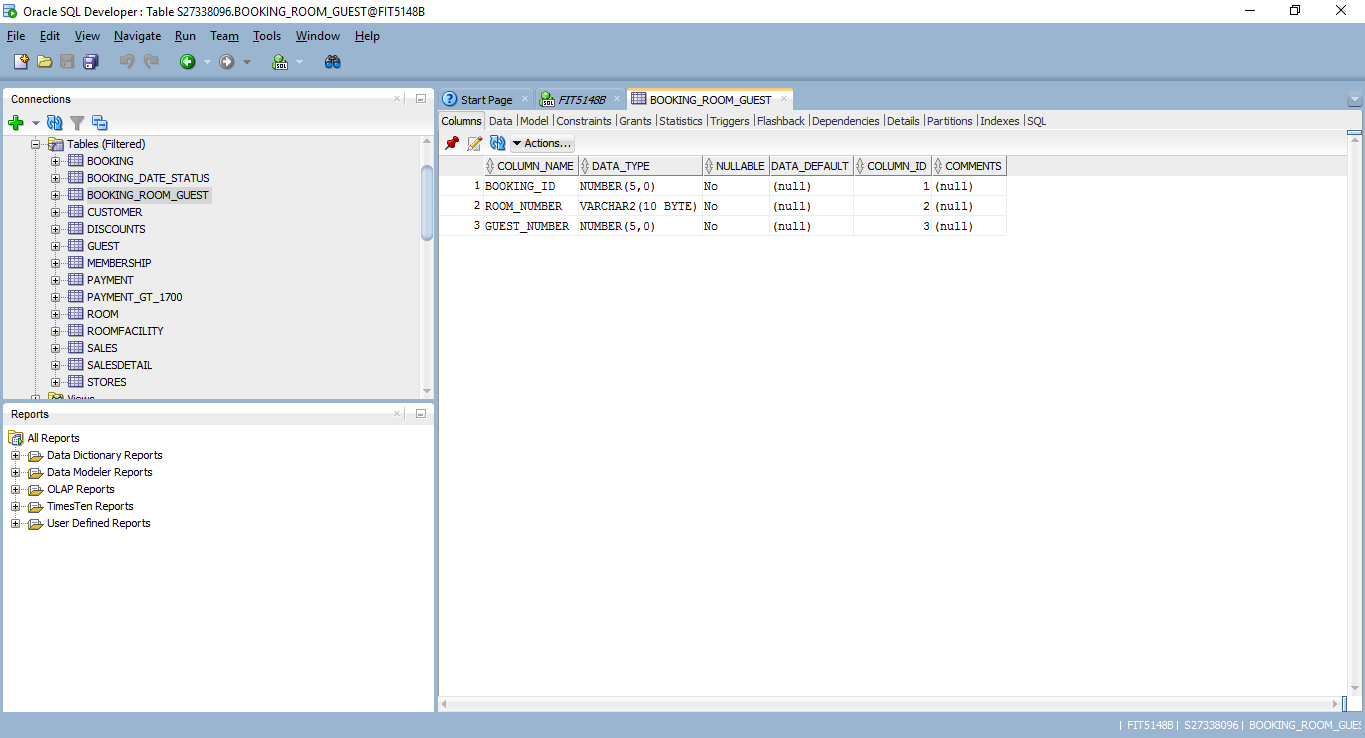
# *Creation*

This table records all the booking data of the rooms that were booked for the hotel. Since one booking can have many rooms and a single room can be in multiple bookings, this creates a many-to many scenario. There is a similar scenario between the guest and booking table. Since there is a constraint concerning max occupancy, a three-way associative table was created. This situation is resolve by creating an associative table to connect this table with the Room table. Booking\_ID was chosen as the primary key.



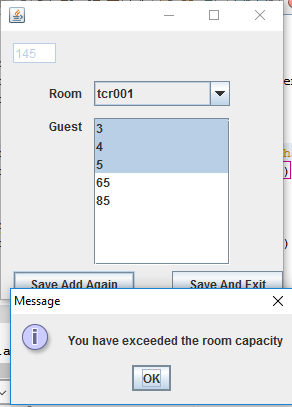






# *Occupancy Constraint*

The occupancy constraint is enforced through the UI. When a record is added through the booking frame, a second form appears asking to assign the guest with rooms. If you choose too many guests, an error message will display.



# *Display Automatic Discount*

The booking frame now calculates a members discount on every creation of a new booking the discount amount is then displayed on the screen.

**if** **(!**validateForm**()){**

**return;**

**}**

double discPerc **=** 0.0**;**

int discAmount **=** 0**;**

**try{**

ResultSet rset **=** getController**().**getModelBStatement**().**executeQuery**(**"Select Membership\_discount from membership m, customer c where c.membership\_tier = m.membership\_tier And c.Customer\_number = " **+** String**.**valueOf**(this.**cboCustomerNumber**.**getSelectedItem**()));**

**while** **(**rset**.**next**()){**

discPerc **=** Double**.**parseDouble**(**rset**.**getString**(**1**));**

**}**

**}**

**catch(**Exception e**){**

getController**().**showPopup**(**e**.**getMessage**());**

**}**

String strSQL **=** ""**;**

int totAmount **=** 0**;**

**try{**

totAmount **=** Integer**.**parseInt**(this.**txtBookingTotalAmount**.**getText**().**trim**());**

discAmount **=** **(**int**)(**discPerc **\*** totAmount**);**

totAmount **=** totAmount **-** discAmount**;**

**}**

**catch** **(**NumberFormatException nfe**){**

getController**().**showPopup**(**"Please enter a valid number"**);**

**this.**setSubFormVisibility**(false);**

**this.**setActionType**(**""**);**

**return;**

**}**

int statusValue **=** 0**;**

**if** **(**String**.**valueOf**(this.**cboPaymentStatus**.**getSelectedItem**()).**equals**(**"Paid"**)){**

statusValue **=** 1**;**

**}**

int custNum **=** 0**;**

custNum **=** Integer**.**parseInt**(**String**.**valueOf**(this.**cboCustomerNumber**.**getSelectedItem**()));**

**try{**

ResultSet rset **=** getController**().**getModelBStatement**().**executeQuery**(**"select booking\_seq.nextval from booking"**);**

int id **=** 0**;**

**while** **(**rset**.**next**()){**

id **=** rset**.**getInt**(**1**);**

**}**

strSQL **=** "INSERT INTO Booking Values (" **+** id **+** ", "**;**

strSQL **+=** custNum**;**

strSQL **+=** ", '" **+** **this.**txtBookingCheckInDate**.**getText**().**trim**();**

strSQL **+=** "', '" **+** **this.**txtBookingCheckOutDate**.**getText**().**trim**();**

strSQL **+=** "', '" **+** **this.**txtBookingContactPerson**.**getText**().**trim**();**

strSQL **+=** "', '" **+** **this.**txtBookingContactEmail**.**getText**().**trim**();**

strSQL **+=** "', " **+** totAmount**;**

strSQL **+=** ", " **+** statusValue **+** ")"**;**

rset **=** getController**().**getModelBStatement**().**executeQuery **(**strSQL**);**

getController**().**showPopup**(**"Your booking has been created. There was a discount applied of " **+** Double**.**toString**(**discAmount**));**

getController**().**setView**(new** BookingSubView**(**id**),** **this);**

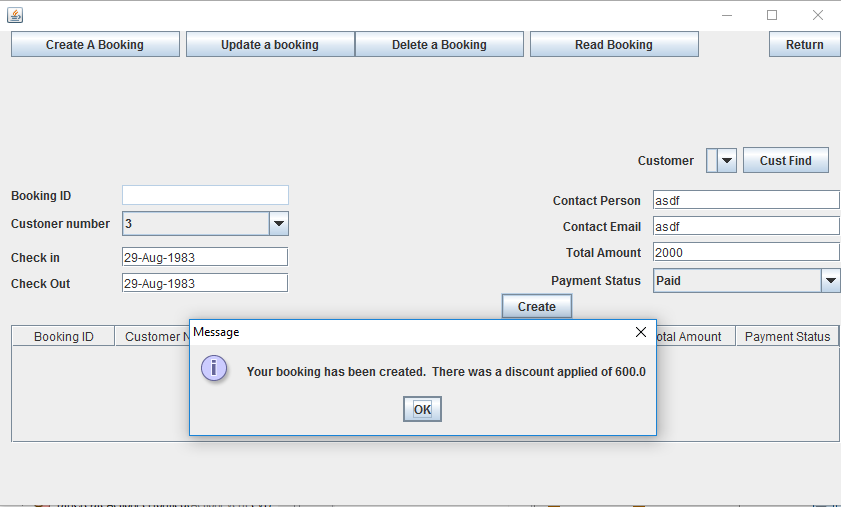
rset**.**close**();**

**}**

**catch(**Exception e**){**

getController**().**showPopup**(**e**.**getMessage**());**

**}**

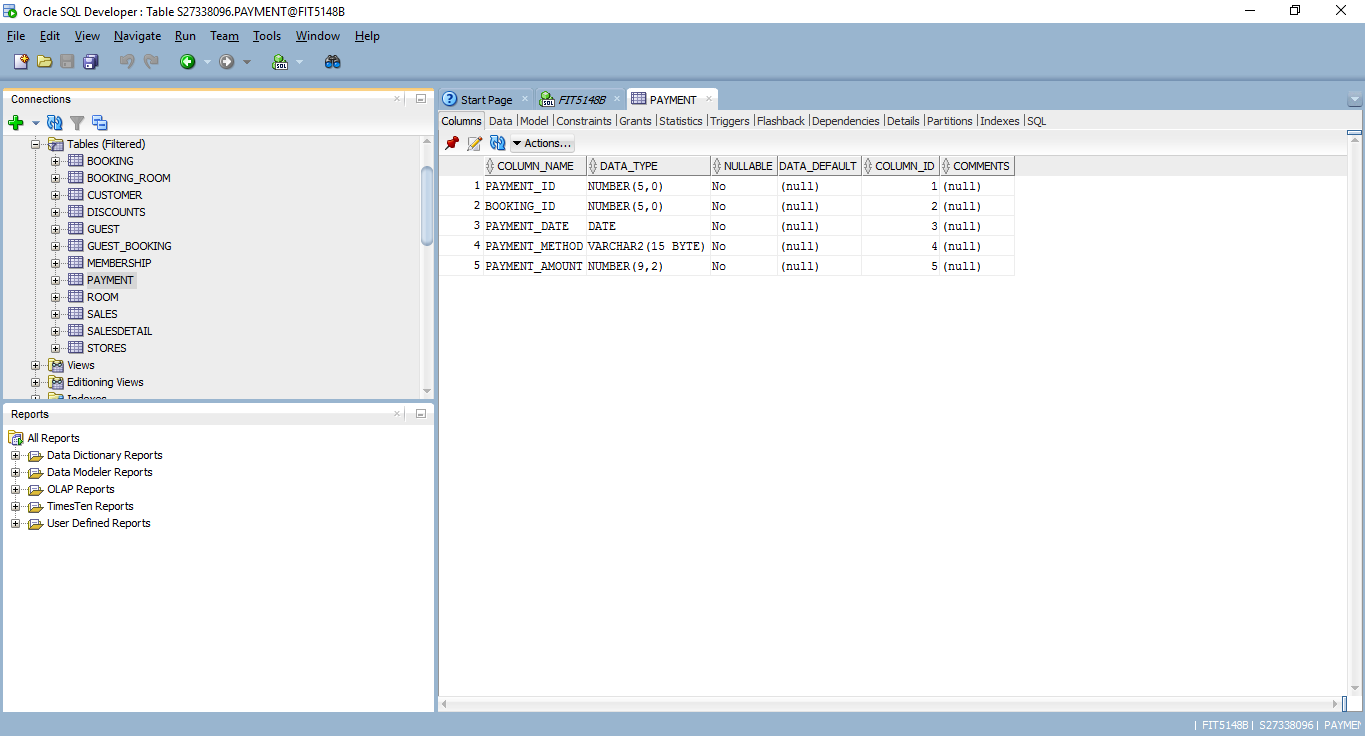


# *Payment@FIT5148B*

# *Creation*

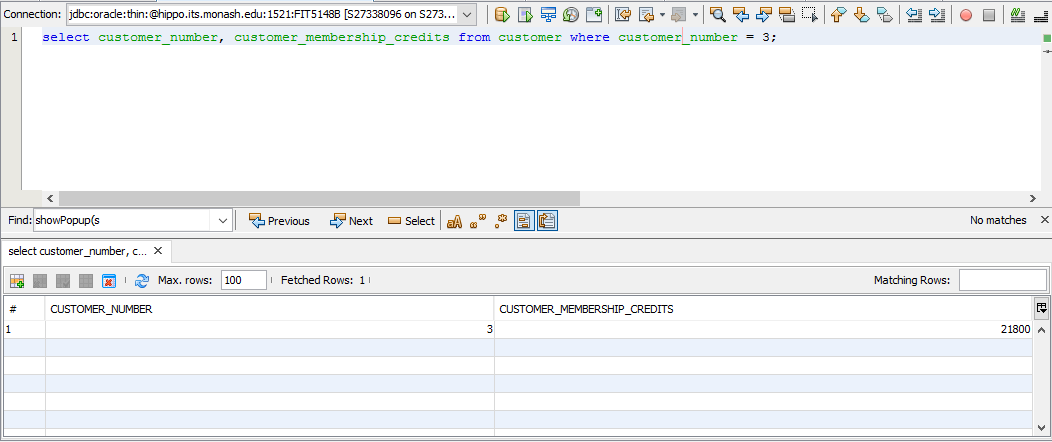
This table records all the payment information for the hotel system. The payment table has a foreign key of Booking\_ID to create the relationship back into the system. This payment\_id was chosen as the primary key.

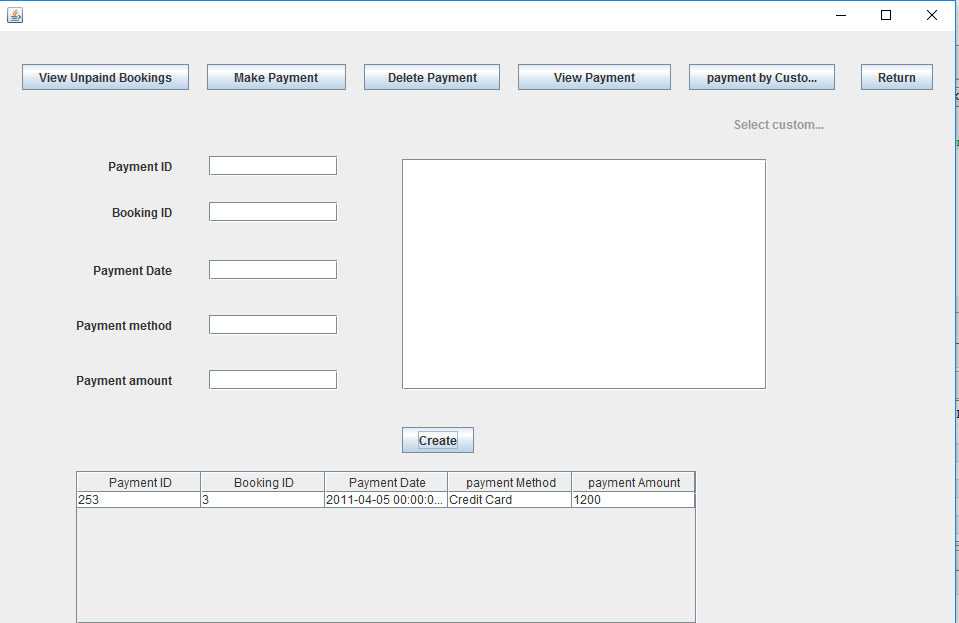


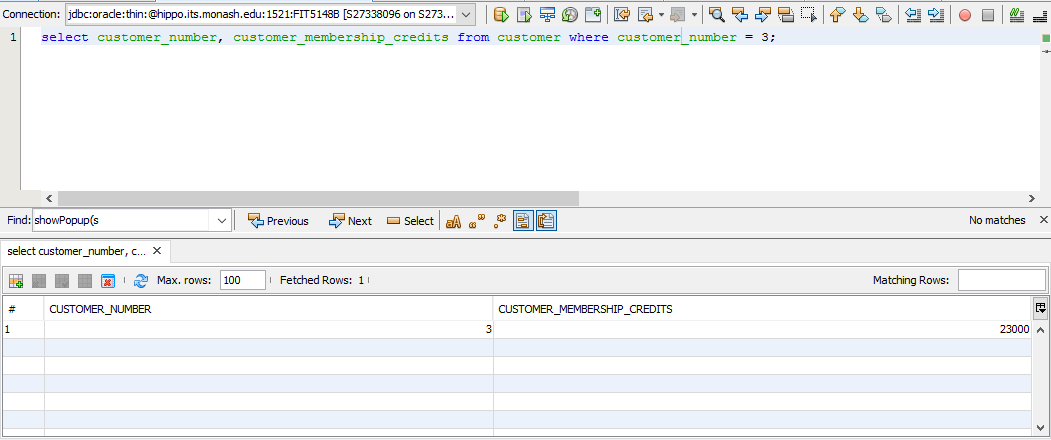


# *Payment Status*

There is also a payment status constraint that regarding the payment table. Whenever a payment gets added, the customer\_membership\_account should be update accordingly. This was accomplished via the Frame that is responsible for the payments.







# Table Implementation

This tables were altered to add the foreign key relationships that are illustrated in part c. Sequences were also created for six of the primary keys.

**Alter** **Table** RoomFacility **ADD** **Constraint** Room\_Facility\_fk **foreign** **key** **(**room\_number**)** **REFERENCES** Room**(**Room\_Number**)** **On** **DELETE** **CASCADE;**

**Alter** **Table** Payment **Add** **CONSTRAINT** Parment\_Booking\_fk **foreign** **key** **(**Booking\_ID**)** **REFERENCES** Booking**(**Booking\_ID**)** **On** **DELETE** **CASCADE;**

**Alter** **Table** Booking\_Room\_Guest **Add** **Constraint** BR\_Room\_fk **foreign** **key** **(**Room\_Number**)** **REFERENCES** Room**(**Room\_number**)** **On** **DELETE** **CASCADE;**

**ALTER** **Table** Booking\_Room\_Guest **Add** **Constraint** BR\_Booking\_fk **foreign** **key** **(**Booking\_ID**)** **REFERENCES** Booking**(**Booking\_ID**)** **On** **DELETE** **CASCADE;**

**ALTER** **Table** Booking\_Room\_Guest **Add** **Constraint** BR\_Guest\_fk **foreign** **key** **(**Guest\_Number**)** **REFERENCES** Guest**(**Guest\_Number**)** **On** **DELETE** **CASCADE;**

**Alter** **table** Booking **Add** **CONSTRAINT** Booking\_fk **foreign** **key** **(**customer\_number**)** **references** Customer**(**customer\_number**)** **On** **Delete** **Cascade;**

**alter** **table** Customer **add** **constraint** cust\_Mem\_fk **foreign** **key** **(**Membership\_Tier**)** **REFERENCES** Membership**(**Membership\_tier**)** **on** **delete** **cascade;**

**Alter** **table** Payment **add** **constraint** type\_chk **check(**Payment\_Method **In(**'Cash'**,**'Check'**,**'Credit Card'**));**

**CREATE** **SEQUENCE** payment\_seq MAXVALUE 99999 **START** **WITH** 1 **INCREMENT** **By** 1**;**

**CREATE** **SEQUENCE** customer\_seq MAXVALUE 99999 **START** **WITH** 1 **INCREMENT** **By** 1**;**

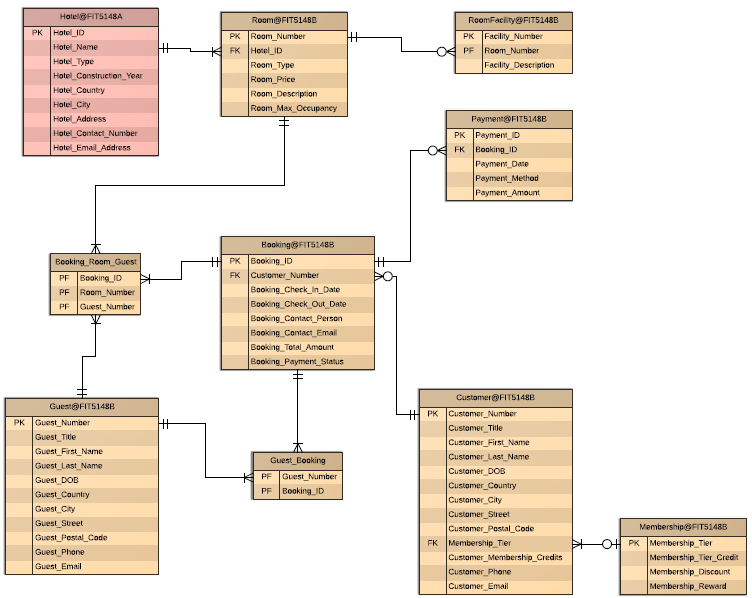
**CREATE** **SEQUENCE** guest\_seq MAXVALUE 99999 **START** **WITH** 1 **INCREMENT** **By** 1**;**

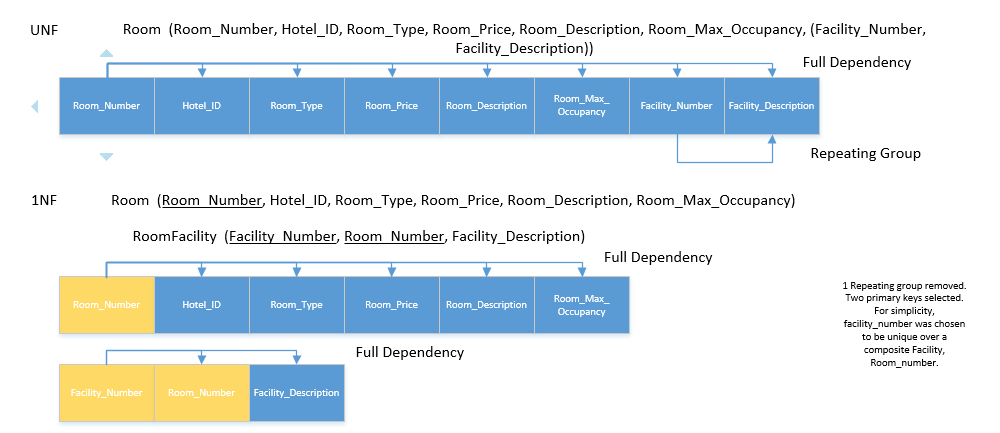
**CREATE** **SEQUENCE** booking\_seq MAXVALUE 99999 **START** **WITH** 1 **INCREMENT** **By** 1**;**

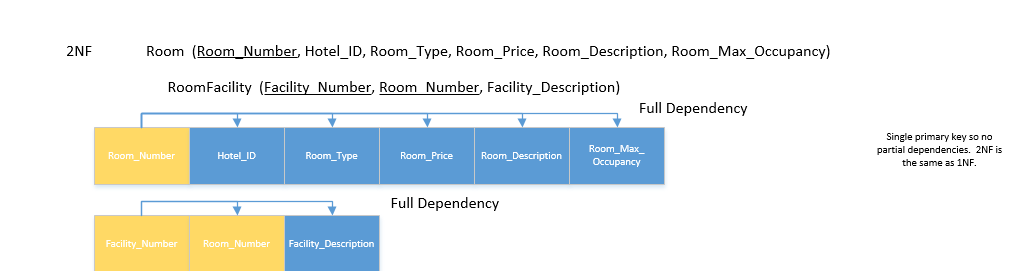
**create** **index** idx\_customer **on** booking**(**customer\_number**);**

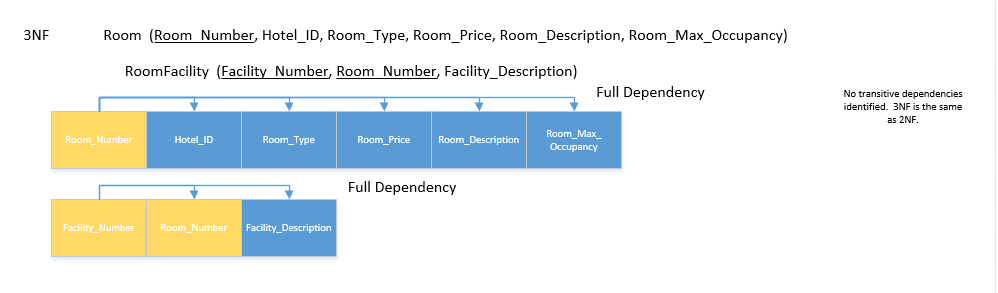
# ER Diagram

This is the ER diagram for the hotel booking system. It largely follows the design specified in the requirements. Two tables were added that were not specified in the report to handle many-to-many relationships (refer to part A). Room Facility did not require an associative table because it was a repeating group of Room. For the primary key, a composite key consisting of Facility\_number and room number was selected. This is different than the other associative tables as guest/room would be repeating groups of each other separating into the associative tables. The diagram for room and room facility was included in this report.









# Index Creation

An index was created on the of the Booking table. This attribute was chosen because the customer booking relationship will be a frequently searched field that doesn’t have an index. This filed is the link between the hotel and the payment information.



# Populate Data

A series of insert into statements were then executed to populate the tables that were just created. Sequences, nextval and currval, from part d was used where applicated to populate the primary keys.

FIT5148A





FIT5148B

**INSERT** **INTO** Room **Values** **(**'tcr001'**,** 1**,** 'single'**,** 60**,** 'A room with two single beds'**,**2**);**

**INSERT** **INTO** Room **Values** **(**'ns001'**,** 2**,** 'twin'**,** 240**,** 'A room with two single beds'**,**2**);**

**INSERT** **INTO** Room **Values** **(**'FS001'**,** 3**,** 'double'**,** 570**,** 'A room with Double bed'**,**2**);**

**INSERT** **INTO** Room **Values** **(**'sol001'**,** 4**,** 'twin double'**,** 1370**,** 'Two double beds'**,**4**);**

**INSERT** **INTO** Room **Values** **(**'ohl001'**,** 5**,** 'hollywood twin'**,** 3000**,** 'Two single beds with a single head board'**,**4**);**

**insert** **into** roomfacility **values(**'HT01'**,**'tcr001'**,**'Hot Tub'**);**

**insert** **into** roomfacility **values(**'Sau01'**,**'ns001'**,**'Sauna'**);**

**insert** **into** roomfacility **values(**'Hrt01'**,**'FS001'**,**'Bed shaped like a heart'**);**

**insert** **into** roomfacility **values(**'Fri01'**,**'sol001'**,**'Mini-Fridge'**);**

**insert** **into** roomfacility **values(**'Bid01'**,**'ohl001'**,**'Bidet'**);**

**insert** **into** membership **values(**'Blue'**,**1000**,**0.01**,**'Car'**);**

**insert** **into** membership **values(**'Silver'**,**3000**,**0.02**,**'Bus'**);**

**insert** **into** membership **values(**'Gold'**,**5000**,**0.05**,**'Boat'**);**

**insert** **into** membership **values(**'Diamond'**,**10000**,**0.15**,**'Sports Car'**);**

**insert** **into** membership **values(**'Platinum'**,**20000**,**0.30**,**'Helicopter'**);**

**insert** **into** customer **values(**customer\_seq**.nextval,**'MR'**,**'James'**,**'Bond'**,**'19-AUG-1971'**,**'UK'**,**'London'**,**'Bond Street'**,**'er4 6yt'**,**'Blue'**,**2900**,**'+4475968473'**,**'JBond@gmail.com'**);**

**insert** **into** guest **values(**guest\_seq**.nextval,**'Mr'**,**'stewart'**,**'Little'**,**'13-JUN-1980'**,**'Australia'**,**'Sydney'**,**'little island'**,**'4322'**,**'+61876435443'**,**'slitt@hotmail.com'**);**

**insert** **into** booking **values(**booking\_seq**.nextval,**customer\_seq**.currval,**'18-MAY-2014'**,**'30-MAY-2014'**,**'Alexander the great'**,**'Alex@yahoo.com'**,**3400**,**1**);**

**insert** **into** booking\_room\_guest **values** **(**booking\_seq**.currval,** 'tcr001'**,** 1**);**

**insert** **into** payment **values(**payment\_seq**.nextval,**booking\_seq**.currval,**'06-JAN-2001'**,**'Credit Card'**,**900**);**

**insert** **into** customer **values(**customer\_seq**.nextval,**'MR'**,**'Thomas'**,**'Gomez'**,**'23-FEB-1921'**,**'Australia'**,**'Perth'**,**'green Street'**,**'6152'**,**'Silver'**,**4999**,**'+61475968473'**,**'TGoz@gmail.com'**);**

**insert** **into** guest **values(**guest\_seq**.nextval,**'Mr'**,**'Steve'**,**'jonson'**,**'01-JUN-1980'**,**'Australia'**,**'Sydney'**,**'Morisson'**,**'4533'**,**'+61875400093'**,**'SJv@hotmail.com'**);**

**insert** **into** booking **values(**booking\_seq**.nextval,**customer\_seq**.currval,**'29-NOV-2015'**,**'30-NOV-2015'**,**'Jerry anderson'**,**'jad@yahoo.com'**,**900**,**1**);**

**insert** **into** booking\_room\_guest **values** **(**booking\_seq**.currval,** 'ns001'**,** 2**);**

**insert** **into** payment **values(**payment\_seq**.nextval,**booking\_seq**.currval,**'17-APR-2011'**,**'Credit Card'**,**3200**);**

**insert** **into** customer **values(**customer\_seq**.nextval,**'DR'**,**'George'**,**'catton'**,**'20-AUG-1981'**,**'Canada'**,**'Ontario'**,**'Little india'**,**'rt5 56'**,**'Platinum'**,**23000**,**'+5475968473'**,**'GCat@gmail.com'**);**

**insert** **into** guest **values(**guest\_seq**.nextval,**'Mr'**,**'Angela'**,**'Lee'**,**'15-FEB-1990'**,**'Australia'**,**'Perth'**,**'river street'**,**'3133'**,**'+61874343393'**,**'ale@hotmail.com'**);**

**insert** **into** booking **values(**booking\_seq**.nextval,**customer\_seq**.currval,**'04-DEC-2015'**,**'11-JAN-2016'**,**'Andy Cooper'**,**'ACP@yahoo.com'**,**1900**,**1**);**

**insert** **into** booking\_room\_guest **values** **(**booking\_seq**.currval,** 'ns001'**,** 3**);**

**insert** **into** payment **values(**payment\_seq**.nextval,**booking\_seq**.currval,**'06-APR-2011'**,**'Credit Card'**,**1200**);**

**insert** **into** customer **values(**customer\_seq**.nextval,**'MR'**,**'Hemant'**,**'Bachhan'**,**'11-JUL-1969'**,**'USA'**,**'Florida'**,**'Olivia street'**,**'94532'**,**'Gold'**,**7000**,**'+0175968473'**,**'HBH@gmail.com'**);**

**insert** **into** guest **values(**guest\_seq**.nextval,**'Mrs'**,**'Xenia'**,**'Mann'**,**'15-NOV-1990'**,**'Kenya'**,**'Nairobi'**,**'nabel street'**,**'439 iu8'**,**'+54874343393'**,**'manx@hotmail.com'**);**

**insert** **into** booking **values(**booking\_seq**.nextval,**customer\_seq**.currval,**'21-DEC-2015'**,**'31-DEC-2015'**,**'Selena Gomez'**,**'sel@yahoo.com'**,**9500**,**1**);**

**insert** **into** booking\_room\_guest **values** **(**booking\_seq**.currval,** 'FS001'**,**4**);**

**insert** **into** payment **values(**payment\_seq**.nextval,**booking\_seq**.currval,**'09-JUN-2005'**,**'Cash'**,**1100**);**

**insert** **into** customer **values(**customer\_seq**.nextval,**'MR'**,**'Rachel'**,**'King'**,**'17-DEC-1992'**,**'Poland'**,**'Eliota'**,**'Polish street'**,**'ri 888'**,**'Diamond'**,**12000**,**'+0354468473'**,**'Rkg@gmail.com'**);**

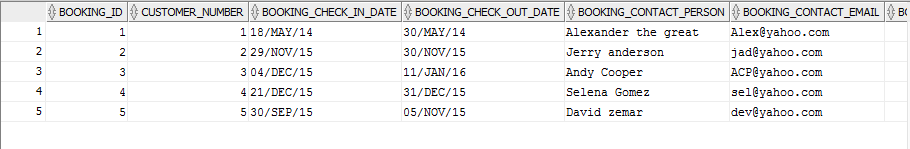
**insert** **into** guest **values(**guest\_seq**.nextval,**'Mr'**,**'Abdul'**,**'Aziz'**,**'15-JAN-1989'**,**'Qatar'**,**'Doha'**,**'Persian way'**,**'54q 3'**,**'+987874343393'**,**'Azz@hotmail.com'**);**

**insert** **into** booking **values(**booking\_seq**.nextval,**customer\_seq**.currval,**'30-SEP-2015'**,**'05-NOV-2015'**,**'David zemar'**,**'dev@yahoo.com'**,**17500**,**1**);**

**insert** **into** booking\_room\_guest **values** **(**booking\_seq**.currval,** 'sol001'**,**5**);**

**insert** **into** payment **values(**payment\_seq**.nextval,**booking\_seq**.currval,**'08-JUL-2006'**,**'Check'**,**2132**);**

**commit;**



# **Task 2 – Oracle Stored Procedure**



/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

**package** storedproc**;**

import **java.sql.**CallableStatement**;**

import **java.sql.**DriverManager**;**

import **java.sql.Connection;**

import **java.sql.SQLException;**

**public** **class** Storedproc**{**

**private** **static** final String DB\_DRIVER **=** "oracle.jdbc.driver.OracleDriver"**;**

**private** **static** final String DB\_CONNECTION **=** "jdbc:oracle:thin:@hippo.its.monash.edu.au:1521:FIT5148B"**;**

**private** **static** final String DB\_USER **=** "S27338096"**;**

**private** **static** final String DB\_PASSWORD **=** "student"**;**

**public** **static** void main**(**String**[]** argv**)** **{**

try **{**

callOracleStoredProcOUTParameter**();**

**}**

catch **(SQLException** e**)** **{**

System**.out.**println**(**e**.**getMessage**());**

**}**

**}**

**private** **static** void callOracleStoredProcOUTParameter**()** throws **SQLException** **{**

**Connection** dbConnection **=** **null;**

CallableStatement callableStatement **=** **null;**

String getEMAILbycustidSql **=** "{call getEMAILbycustid(?,?)}"**;**

try **{**

dbConnection **=** getDBConnection**();**

callableStatement **=** dbConnection**.**prepareCall**(**getEMAILbycustidSql**);**

callableStatement**.**setInt**(**1**,**2**);**

callableStatement**.**setString**(**2**,**"hsdeo1@monash.edu"**);**

**//**callableStatement**.**registerOutParameter**(**2**,** **java.sql.**Types**.VARCHAR);**

**//** **execute** getDBUSERByUserId stored **procedure**

callableStatement**.**executeUpdate**();**

**//**String userName **=** callableStatement**.**getString**(**2**);**

**//**System**.out.**println**(**"UserName : " **+** userName**);**

**}**

catch **(SQLException** e**)** **{**

System**.out.**println**(**e**.**getMessage**());**

**}**

finally **{**

**if** **(**callableStatement **!=** **null)** **{**

callableStatement**.close();**

**}**

**if** **(**dbConnection **!=** **null)** **{**

dbConnection**.close();**

**}**

**}**

**}**

**private** **static** **Connection** getDBConnection**()** **{**

**Connection** dbConnection **=** **null;**

try **{**

**Class.**forName**(**DB\_DRIVER**);**

**}**

catch **(**ClassNotFoundException e**)** **{**

System**.out.**println**(**e**.**getMessage**());**

**}**

try **{**

dbConnection **=** DriverManager**.**getConnection**(**DB\_CONNECTION**,** DB\_USER**,**DB\_PASSWORD**);**

**return** dbConnection**;**

**}**

catch **(SQLException** e**)** **{**

System**.out.**println**(**e**.**getMessage**());**

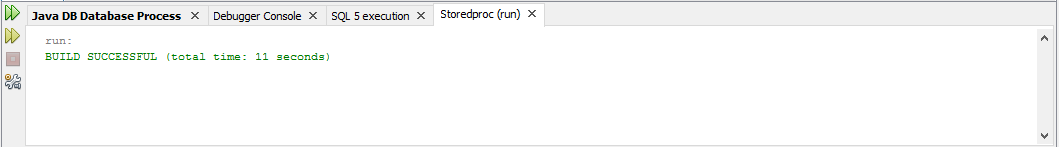
**}**

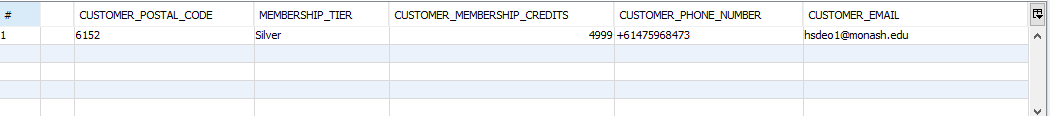
**return** dbConnection**;**

**}**

**}**







# **Task 3 – Triggers**

There was a trigger created on FIT5148A for when there is an insert into the hotel table. This trigger will delete the child entities and update the room entity with a new hotel id on updates. There was also a trigger on FIT5148B to check if a hotel exists to prevent orphan records.

--Trigger to manage integrity of the hotel booking system

**CREATE** **OR** **REPLACE** **TRIGGER** hotel\_integrity

**AFTER** **DELETE** **OR** **INSERT** **OR** **UPDATE** **OF** hotel\_id **on** hotel

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

**BEGIN**

**if** deleting **then**

**DELETE**

**FROM** ROOM@FIT5148B

**WHERE** hotel\_id **=** **:old.**hotel\_id**;**

**else**

**if** updating **then**

**update** room@FIT5148B

**set** hotel\_id **=** **:new.**hotel\_id

**where** hotel\_id **=** **:old.**hotel\_id**;**

**else**

**if** inserting **then**

**insert** **into** room@FIT5148B

**values** **((select** dbms\_random**.**string**(**'U'**,** 5**)** str **from** dual**),:new.**hotel\_id**,**'Single'**,**60**,**'Default Showroom'**,**0**);**

**End** **if;**

**End** **If;**

**End** **if;**

**END;**

--Trigger to manage integrity of the hotel booking system

**CREATE** **OR** **REPLACE** **TRIGGER** hotel\_integrity\_child

**BEFORE** **INSERT** **OR** **UPDATE** **OF** room\_number **on** room

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

**Declare**

hotelCount **number(**6**,**0**);**

**BEGIN**

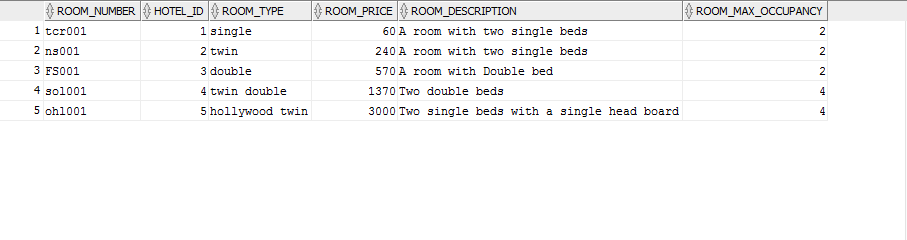
**Select** **count(**hotel\_id**)** **into** hotelCount **From** Hotel@FIT5148A **Where** hotel\_id **=** **:new.**hotel\_id**;**

**if** **(**hotelCount **=** 0**)** **THEN**

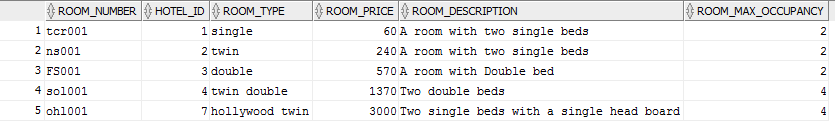
RAISE\_APPLICATION\_ERROR **(**-20099**,**'No hotel exists for this entity.'**);**

**END** **IF;**

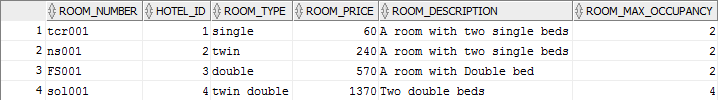
**END;**













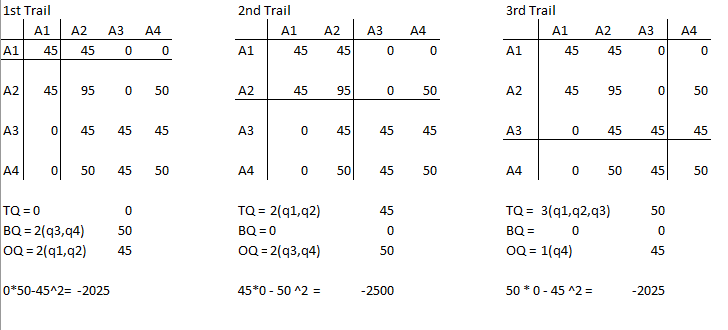
# **Task 4 – Fragmentation**

# Horizontal Fragmentation



# Vertical Fragmentation



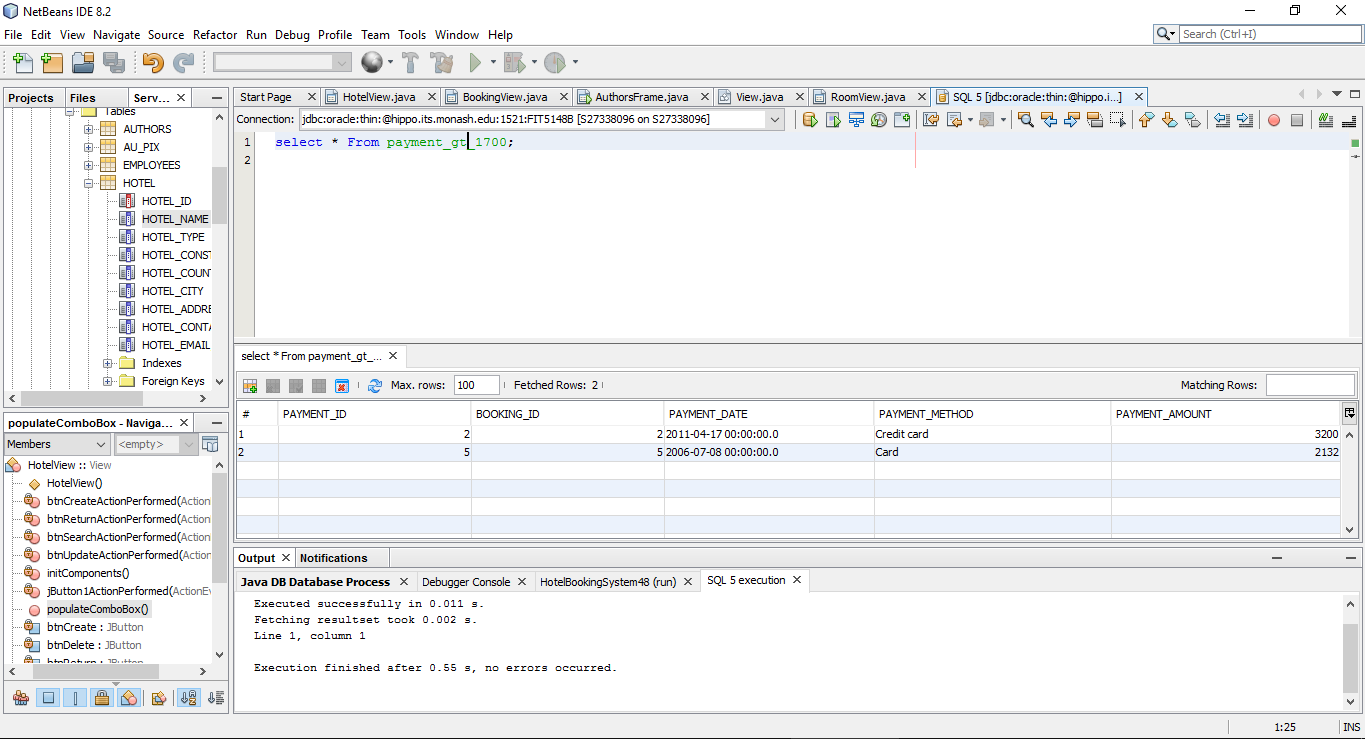


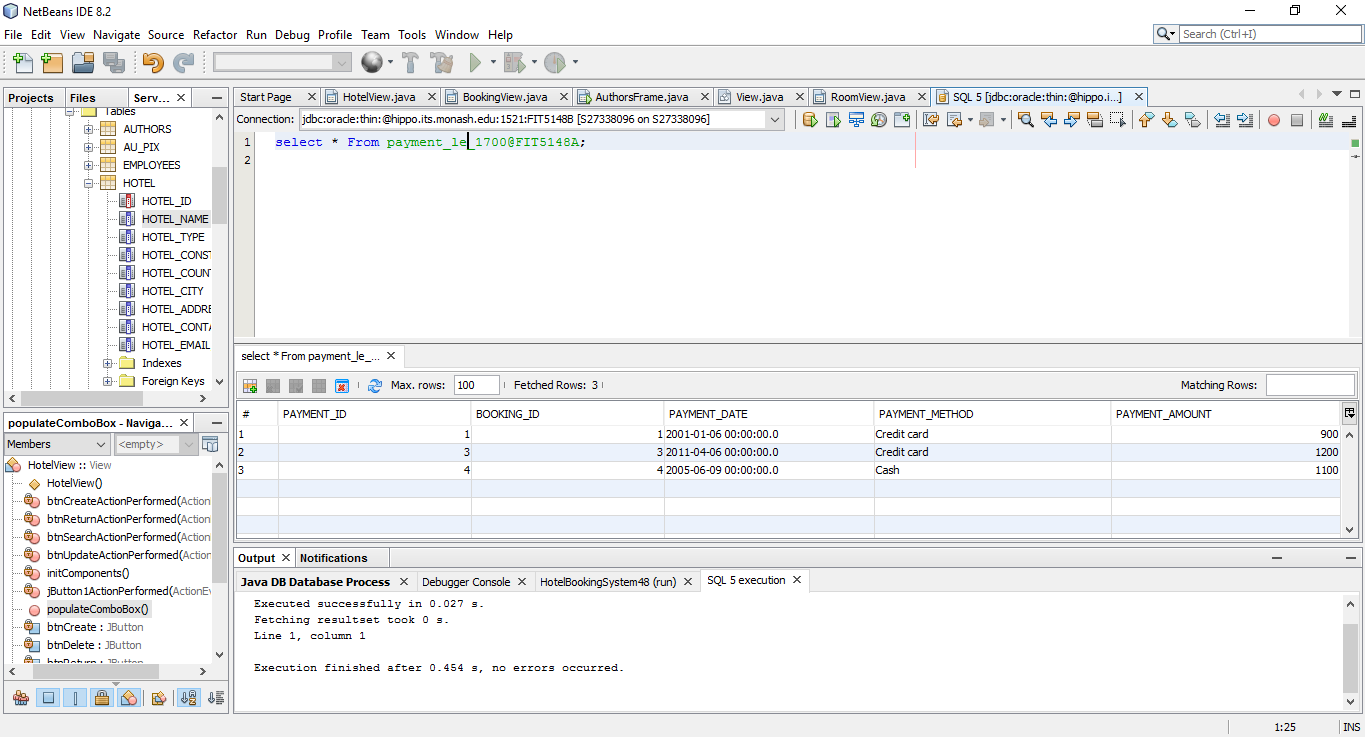
# Implementation

# Horizontal Fragmentation

This query shows the fragmentation implementation for the horizontal fragmentation as described with the methodology above. This should easily be able to achieve the assumed user hypothetical.



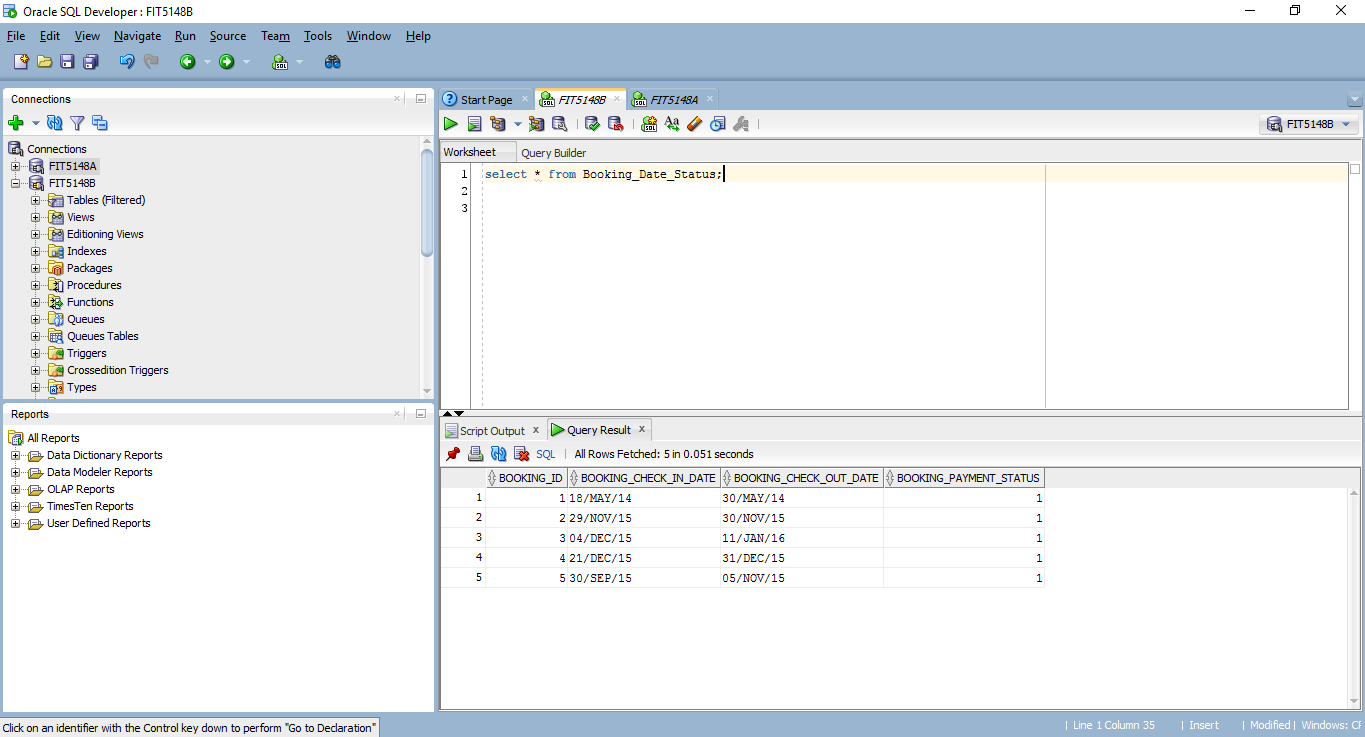


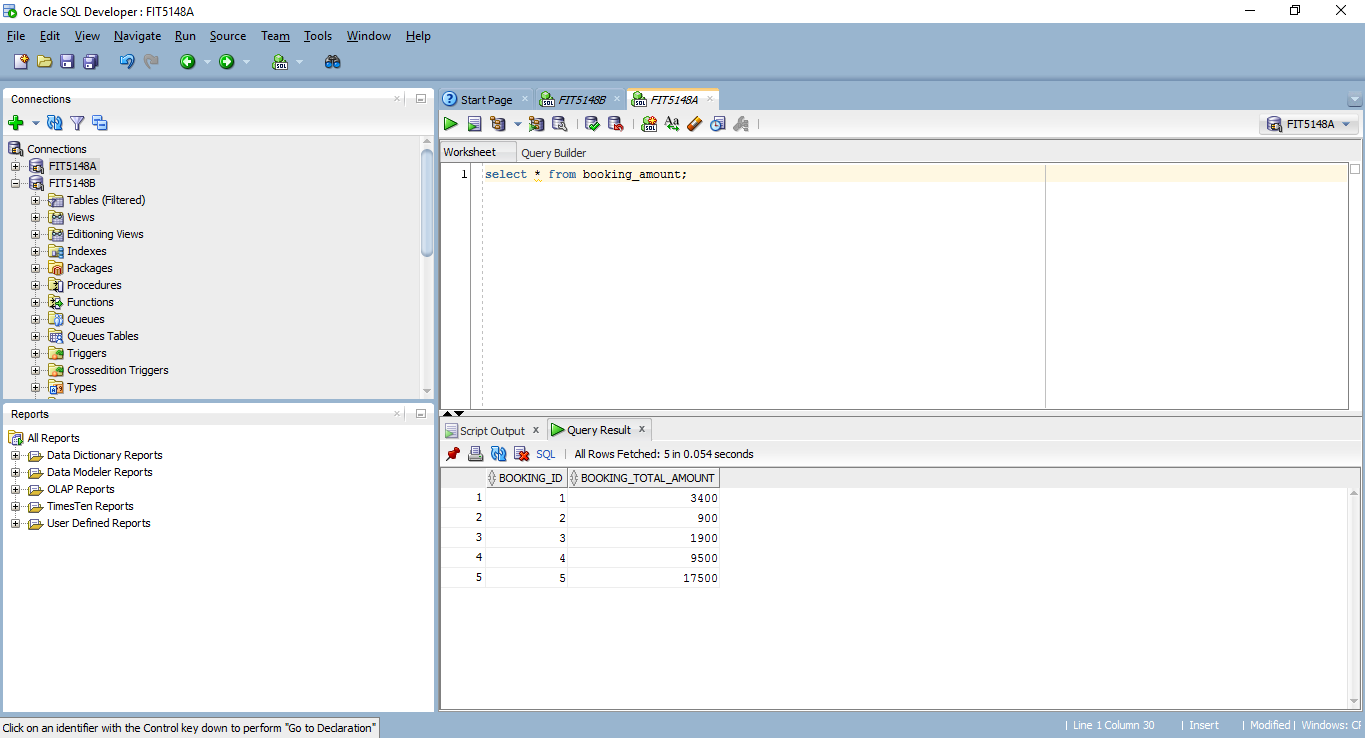


# Vertical Fragmentation

This query shows the fragmentation implementation for the vertical fragmentation as described with the methodology above. First trail had a top (tied) score of -2025 so that is the implementation with attribute A1 in one query and the other three in the other query. This should easily be able to achieve the assumed user hypothetical.







# **Task 5 – User Interface**

This system uses an MVC model to maintain the perpetual connection to the database. The main method creates a model object, which creates the single connection, and instantiate a controller class with the attribute of the model. There is also an abstract View which has a controller attribute. Then, when the model serves up the views, the child view has access to the controller that has the model.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package hotelbookingsystem48**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** java**.**sql**.**Statement**;**

/\*\*

\* This is the controller class for the HBS system.

\*

\* @author Jeffrey Middendorf

\* @version 1.0

\* @since 2/4/2017

\*/

public class HBSController **{**

private HBSModel sysModel**;**

public HBSController**(**HBSModel newMod**){**

sysModel **=** newMod**;**

**}**

public HBSModel getSysModel**(){**

**return** sysModel**;**

**}**

public void inititialize**(){**

MainView frmMainMenu **=** **new** MainView**();**

setView**(**frmMainMenu**);**

**}**

public Statement getModelAStatement**(){**

**return** sysModel**.**getStmtA**();**

**}**

public Statement getModelBStatement**(){**

**return** sysModel**.**getStmtB**();**

**}**

public void setModel**(**HBSModel newMod**){**

sysModel **=** newMod**;**

**}**

public void setView**(**View view**){**

view**.**setController**(this);**

view**.**setVisible**(true);**

**}**

public void setView**(**View view**,** View oldView**){**

view**.**setController**(this);**

view**.**setVisible**(true);**

oldView**.**setVisible**(false);**

**}**

public void showPopup**(**String message**){**

JOptionPane**.**showMessageDialog**(new** JFrame**(),** message**);**

**}**

**}**

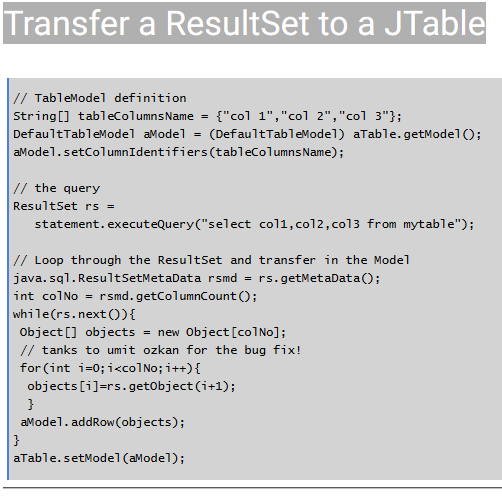


Figure 1- Sample Code Provided By How-To (Gagnon, 2017)

# Main Screen

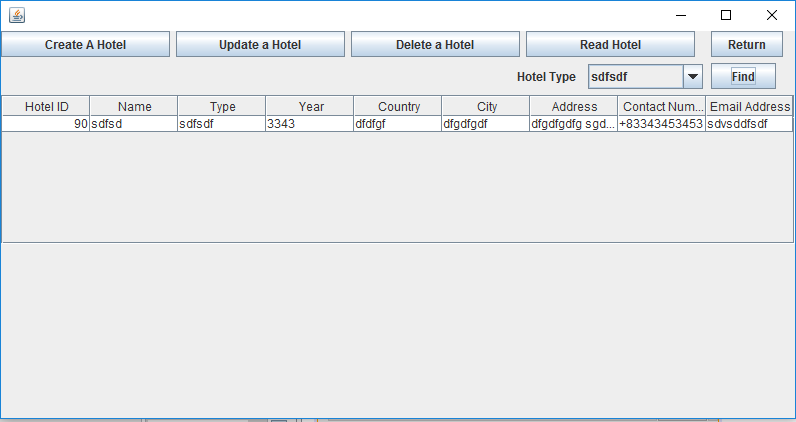
The main form consists of seven buttons that navigate the different forms. It also has one button to exit the system and close the single database connection.



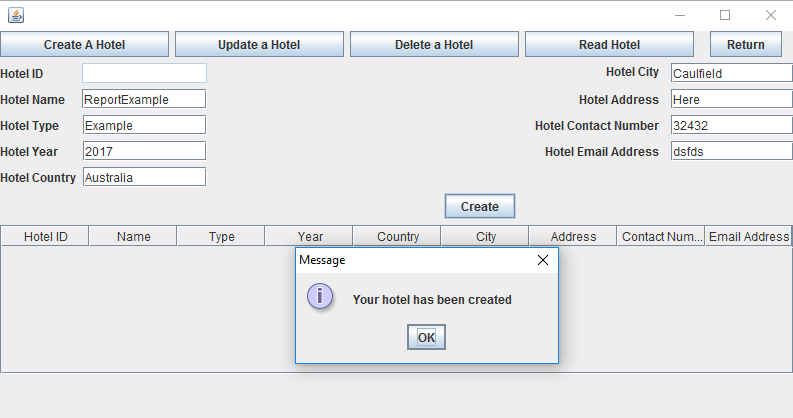
# Hotel Frame

The hotel Frame performs all create, read, update, delete functionality for the hotel table. It also searches by room type.

Search:



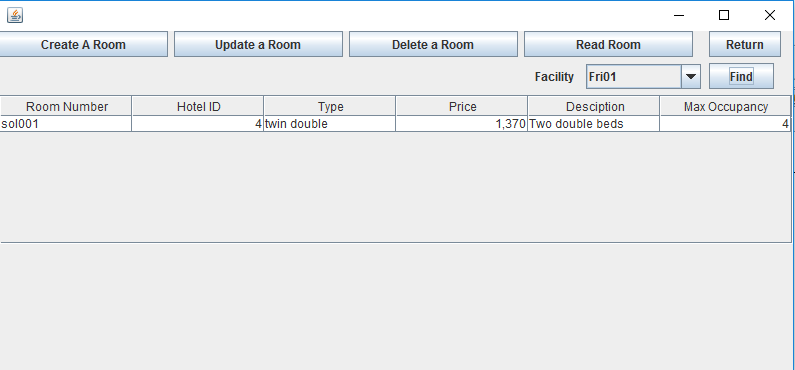
Create:



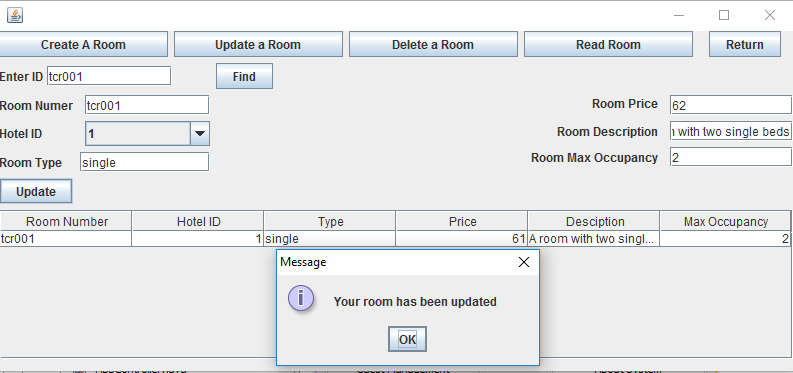
# Room Frame

The Room Frame performs all create, read, update, delete functionality for the Room table. It also searches by Facility.

Search:



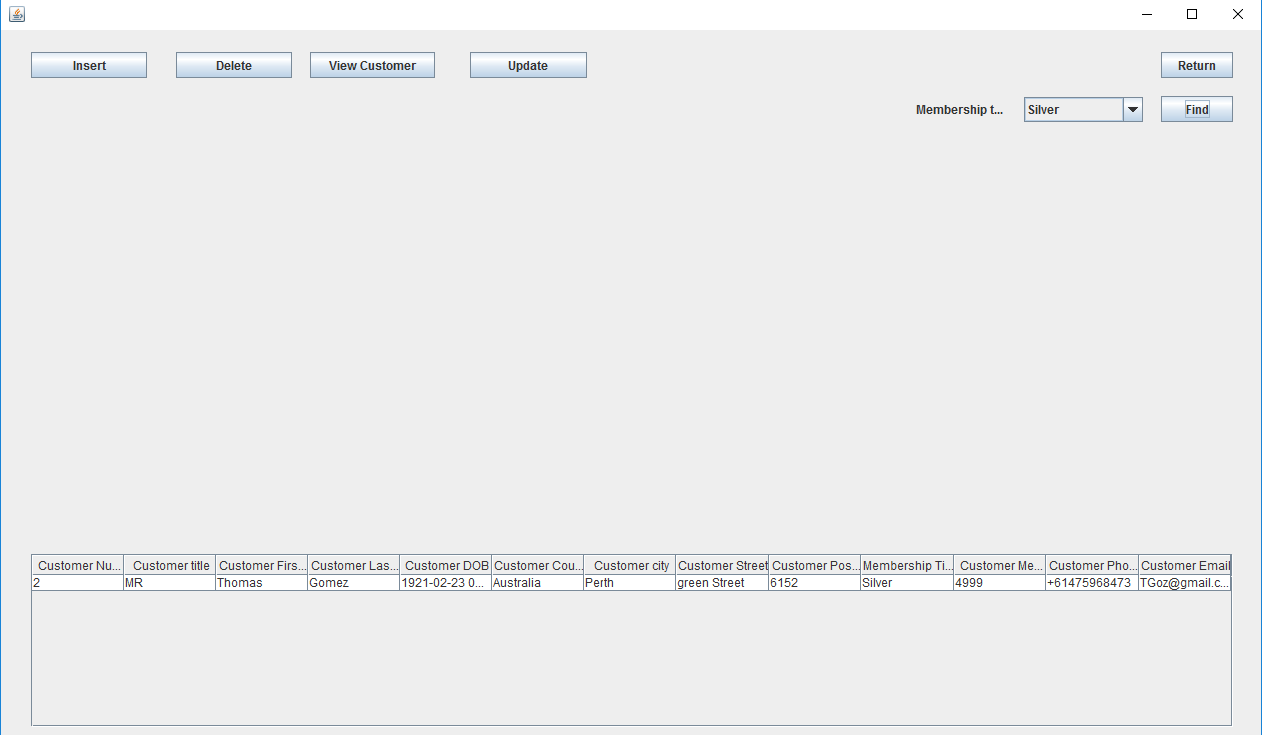
Update:

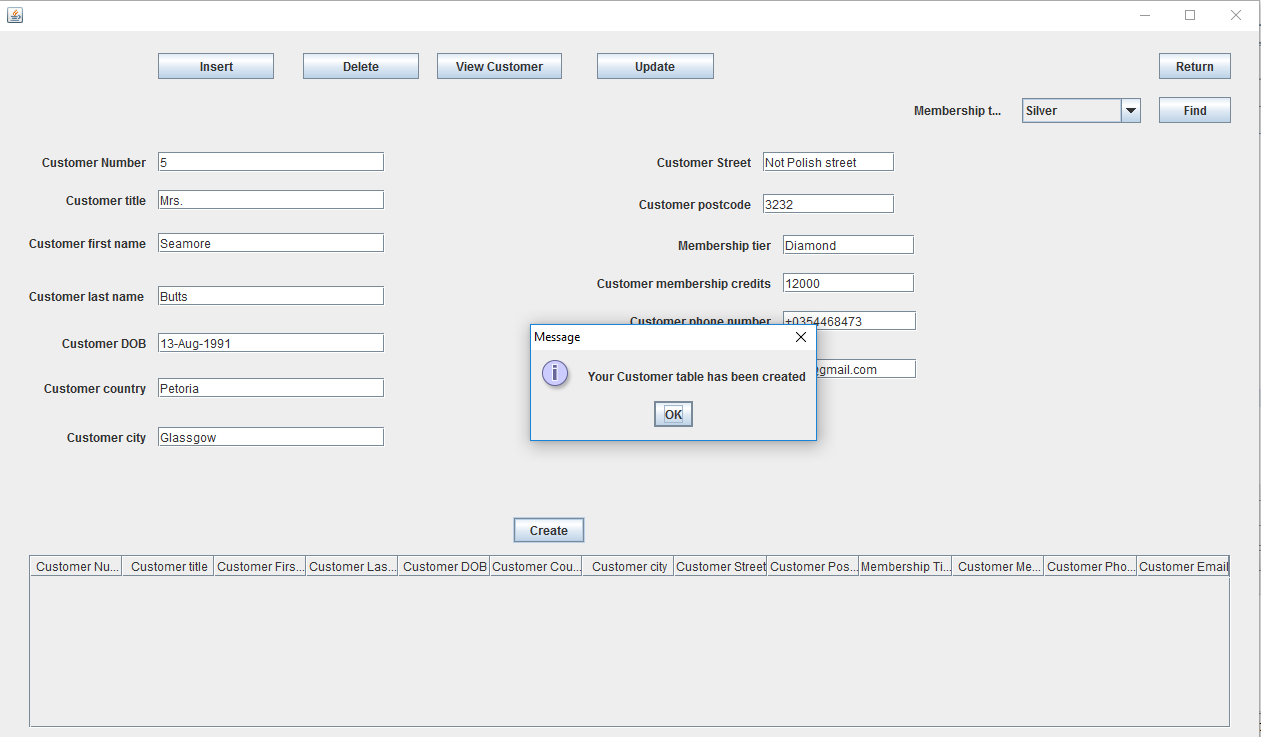


# Customer Frame

The Customer Frame performs all create, read, update, delete functionality for the Customer table. It also searches by Membership Tier.

Search:

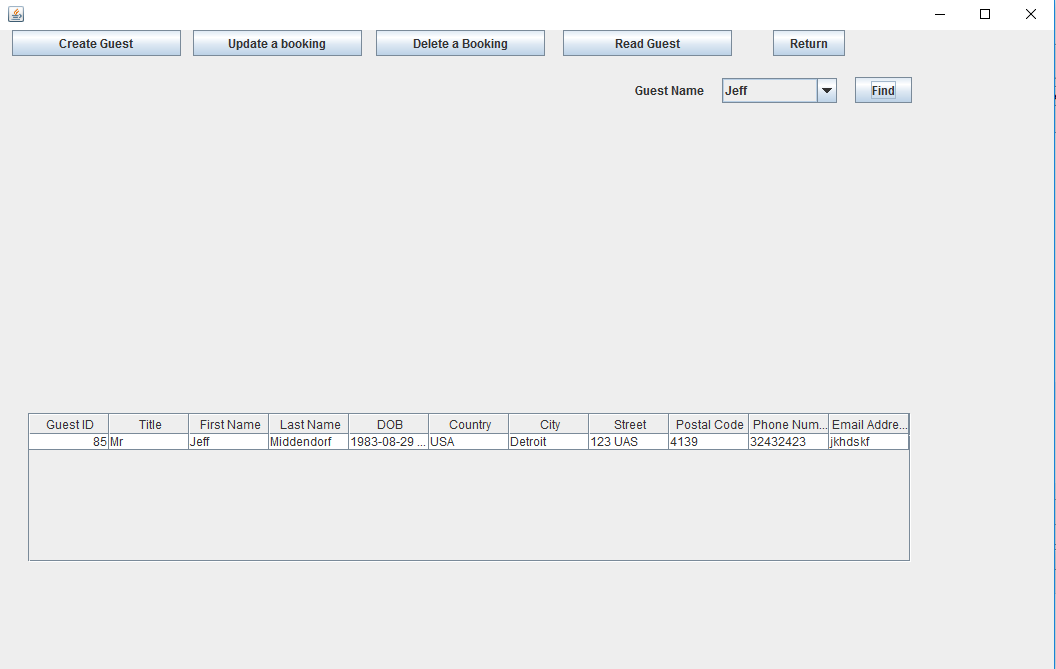




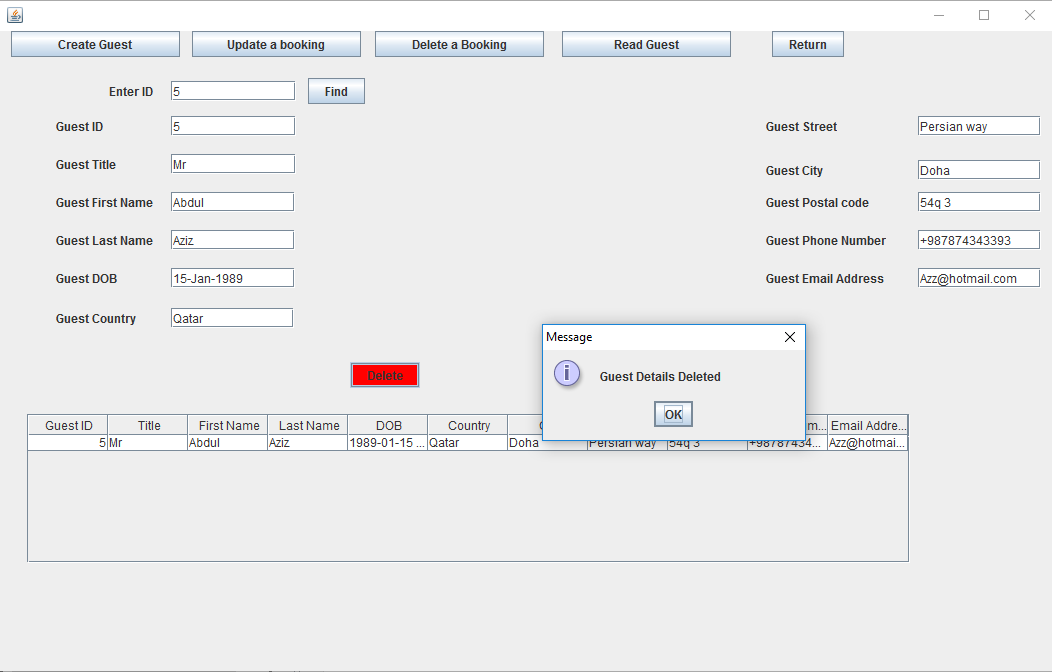
# Guest Frame

The Guest Frame performs all create, read, update, delete functionality for the Guest table. It also searches by guest first name.

Search:



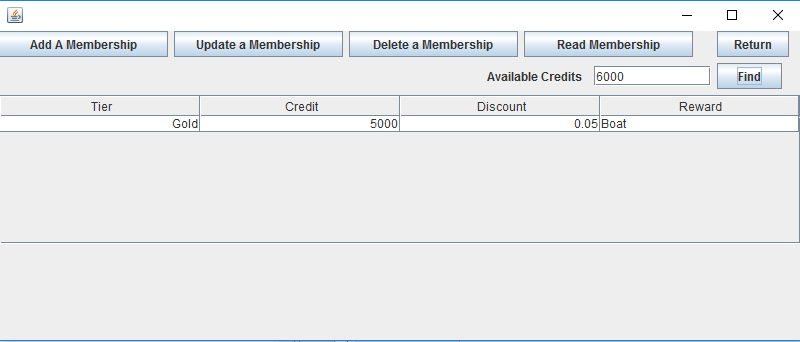
Delete:



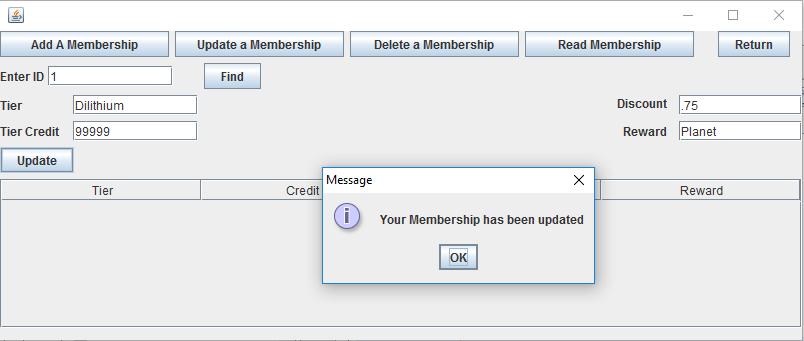
# Membership Frame

The Membership Frame performs all create, read, update, delete functionality for the Membership table. It also for the eligible membership tier based on an input amount.

Search:



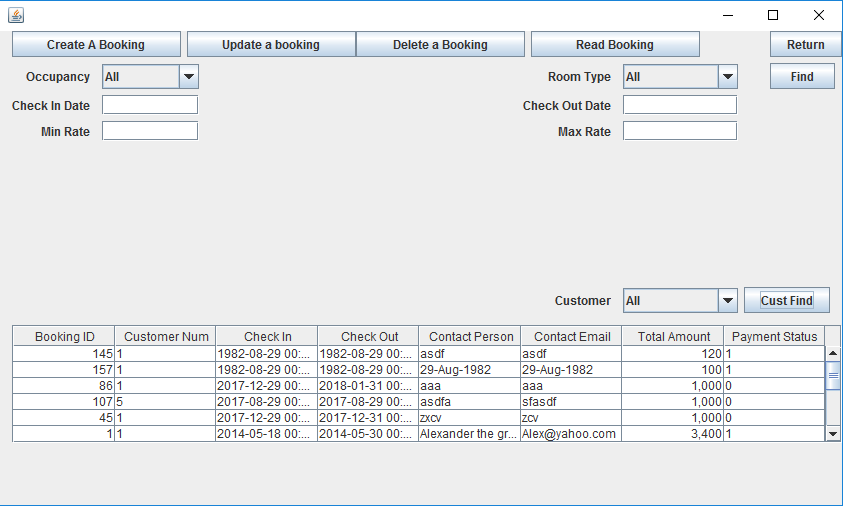
Update:



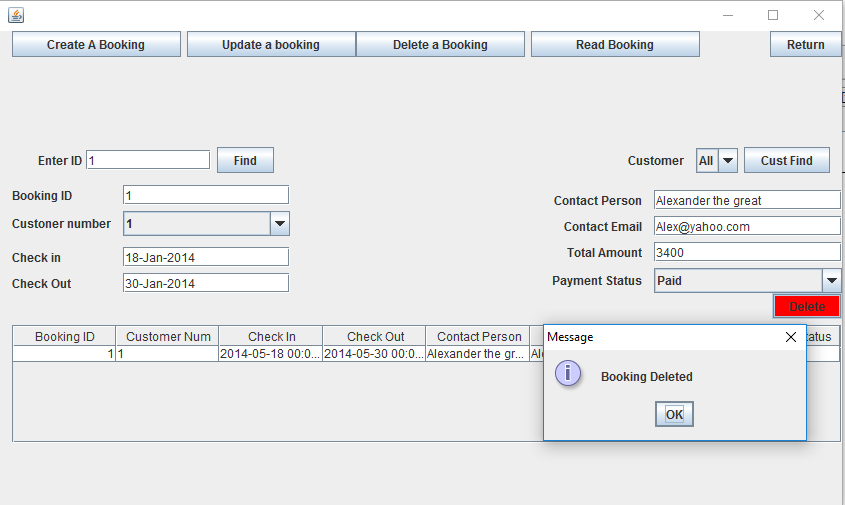
# Booking Frame

The Booking Frame performs all create, read, update, delete functionality for the Booking table. It also searches by customer id, occupancy, room type, price range, and the booked period.

Search:



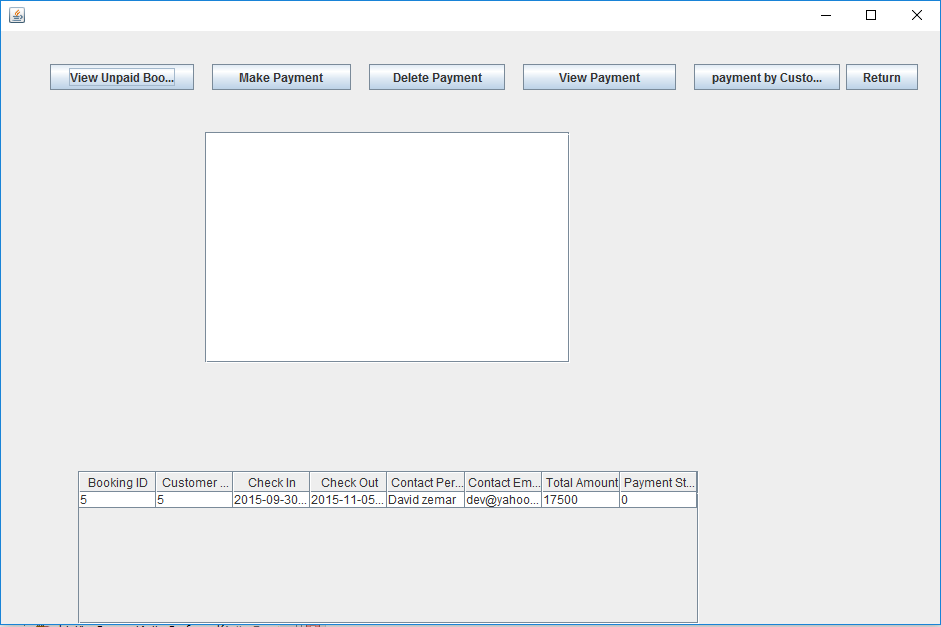
Delete:



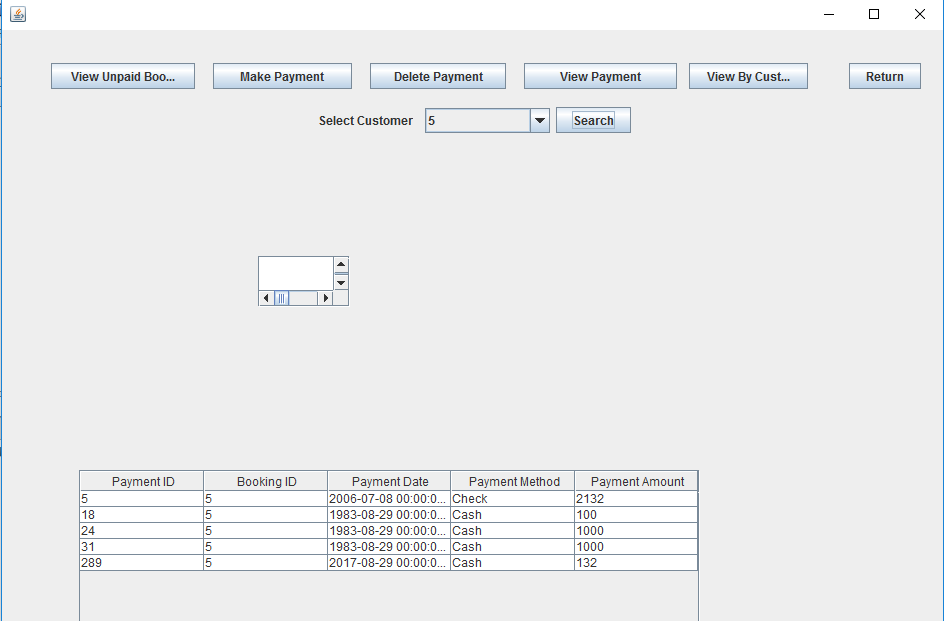
# Payment Frame

The Payment Frame performs all create, read, update, delete functionality for the Payment table. It also searches by unpaid bookings and customer number. There are also two check to perform through this UI, updating the customer’s membership credits and the booking’s payment status.

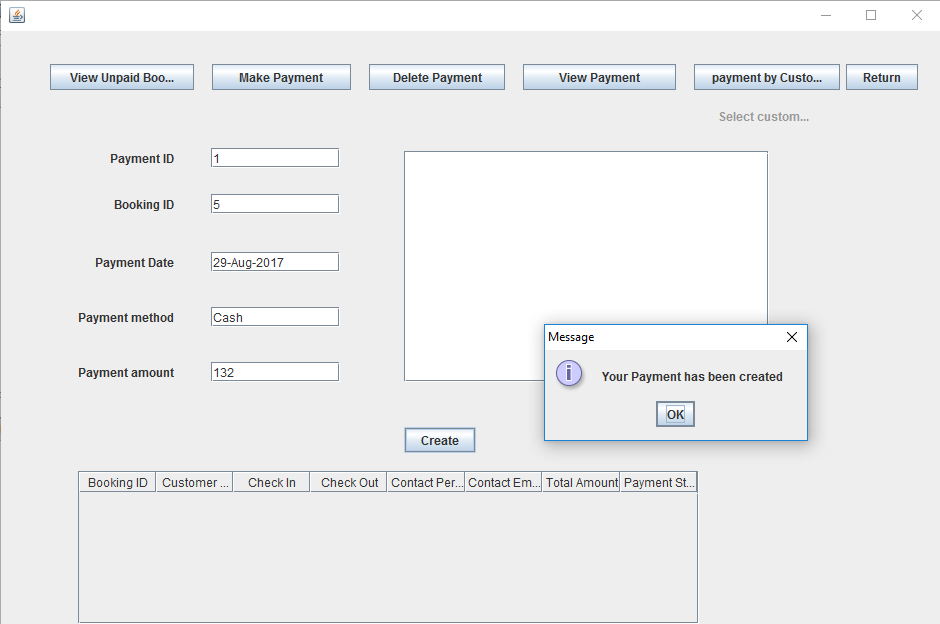
Search:



Search By Customer:

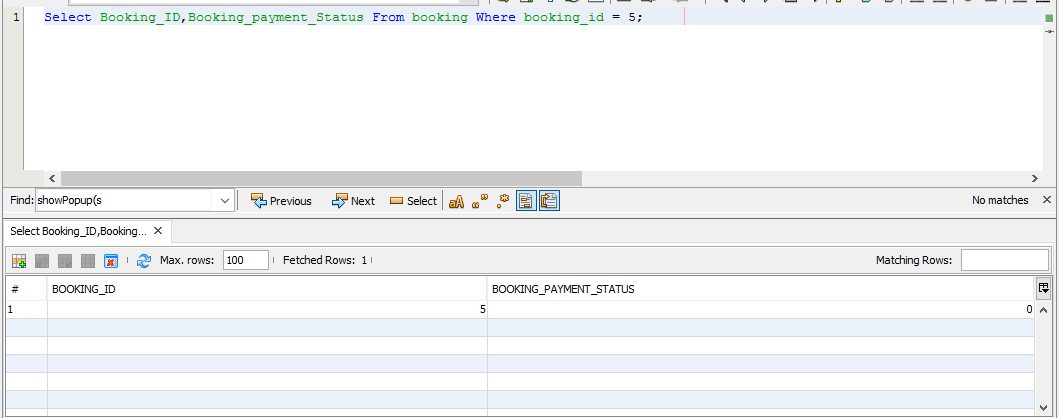


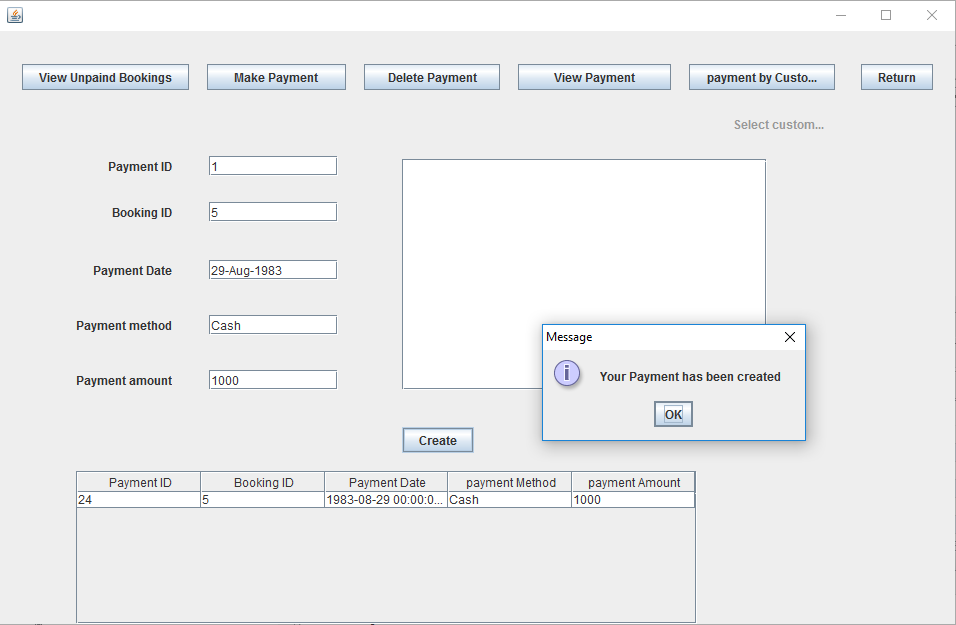
Make a payment:

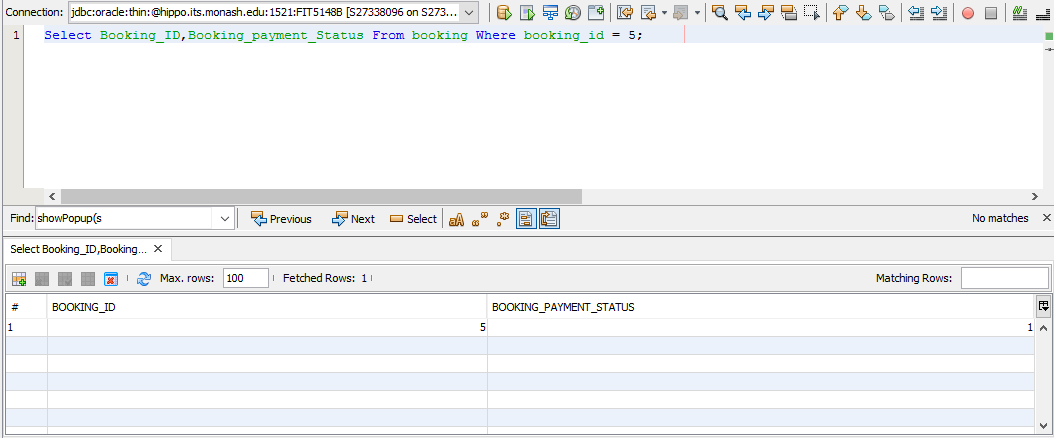


Update booking payment status

The example below is for the insert. The following Customer\_membership\_credits will show the delete.

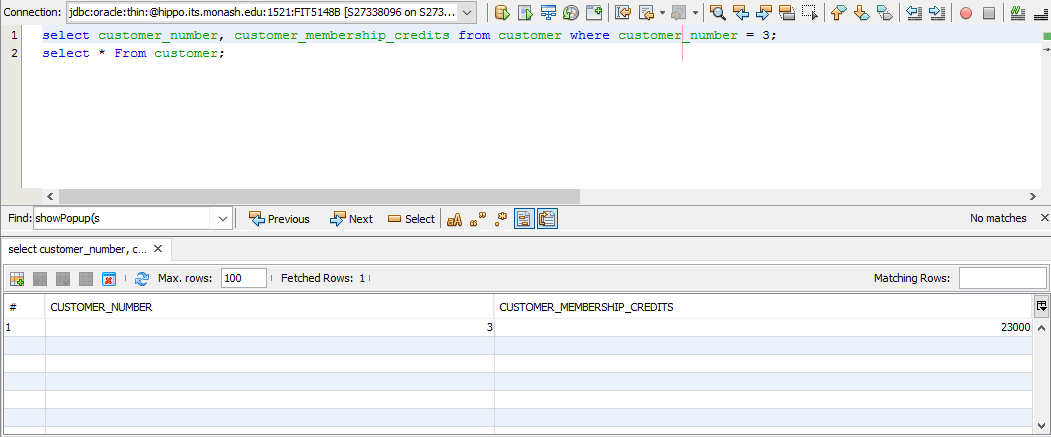


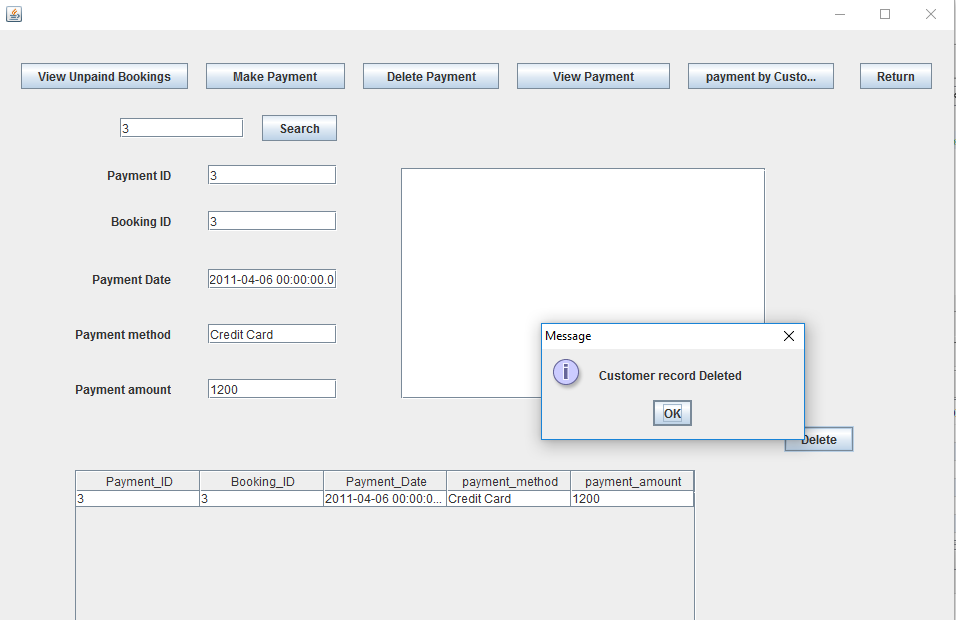


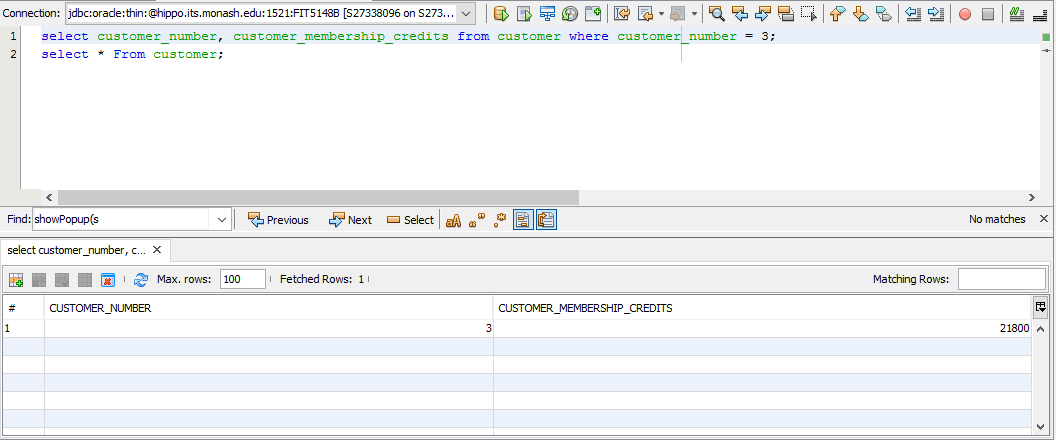


Update Customer Membership Credits:

This shows a removed payment reducing the customer\_membership\_credits. The insert was covered in the previous example.







# **References**

Gagnon, R. (2017, April 2). *Real's How To*. Retrieved from Transfer a ResultSet to a JTable: http://www.rgagnon.com/javadetails/java-0309.html