Student Name:

Student Ref.

**HOTEL BOOKING SYSTEM**

Unit Name: Computer Systems Structure

Unit Code: CIS017-1

Assignment 1- Design and Implement a Database

Deadline:

# **Introduction**

The efficient management of a hotel requires loads of information to be stored based on bookings, rooms, customers and employees. Due to the modernisation and expansion of technology, hotel booking systems are created to ease the workload on staff, reduce errors and overbooking and enhance customers’ experience. As a result, this report investigates the design, development and implementation of a relational database management system that stores information on customers, hotel rooms as well as bookings.

The following tasks give an in-depth investigation into the development of a hotel booking system:

1. Task 1: Entity Relationship Modelling
2. Task 2: Normalisation
3. Task 3: Physical Table Design (including data dictionaries)
4. Task 4: Query Design
5. Task 5: Implementation of Database

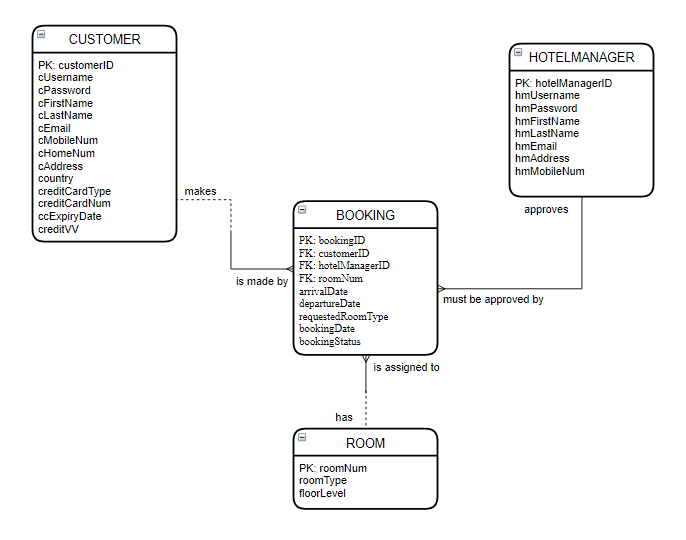
# **Scenario**

A hotel is designing a booking system for customers to make hotel reservations and check their upcoming trips. This database system stores details on customer (name, address, email, telephone number), booking (start and end date, double or single room) and room (double or single room). Customers must log in first before they book a room. When booking a room, the customer must provide a check-in and check-out date as well as whether they would like to stay in a single or double room. After a booking is made, a hotel manager must log in to approve a booking and assign the customer to their desired room type.

# **Task 1: Entity Relationship Modelling**

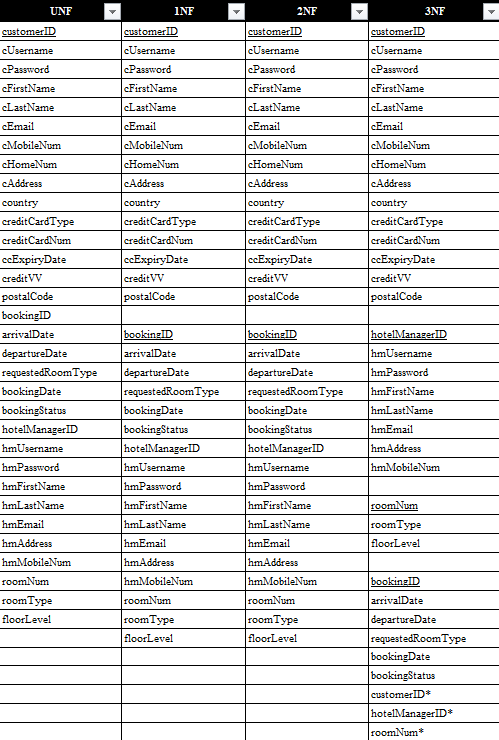
Assumptions

1. While each customer must register to access the hotel’s website, some customers may not make a booking.
2. When hotel managers are approving bookings, customers who requested bookings later
3. One booking requests only one room. Therefore, if a customer would like to reserve another room, the customer needs to make another booking.
4. No two customers or hotel managers can upload the same email and mobile number because these attributes are unique.



**Figure 1: Entity Relationship Model Showing the Relationship among Entities for the Hotel Booking System**

# **Task 2: Normalisation**



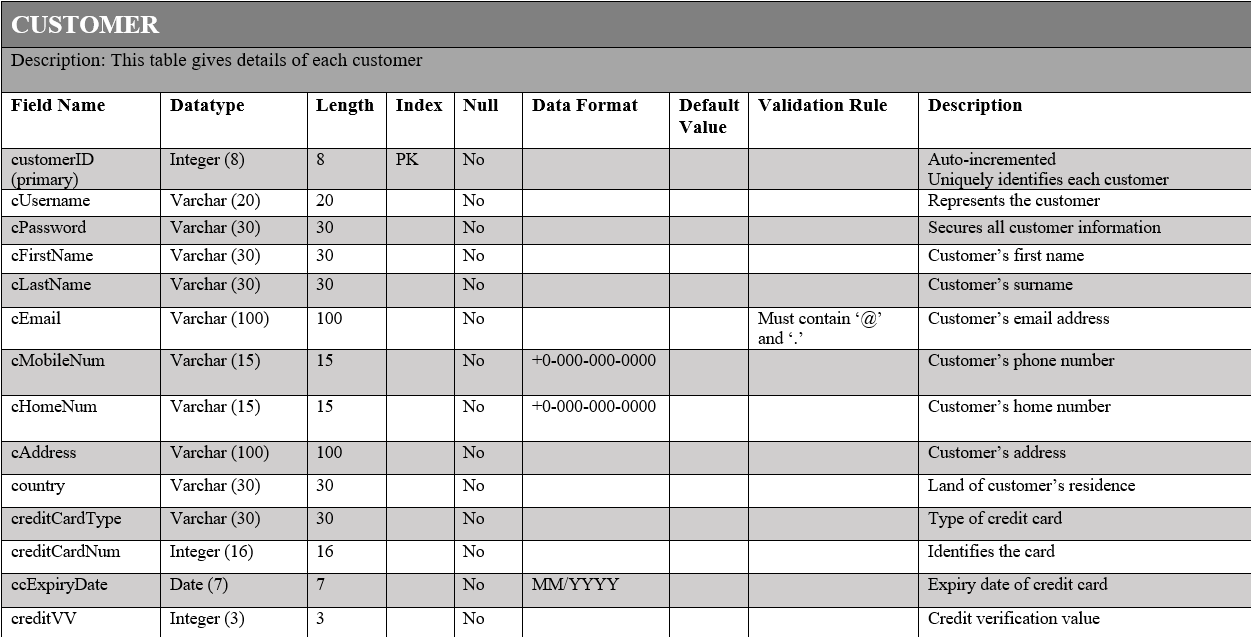
**Figure 2: Table Showing the Normalised Set of Tables/Entities for the Hotel Booking System**

# **Task 3: Physical Table Design (including Data Dictionary)**

## **Part A: Skeleton Tables**

1. Customer (customerID, cUsername, cPassword, cFirstName, cLastName, cEmail, cMobileNum, cHomeNum, cAddress, country, creditCardType, creditCardNum, ccExpiryDate, creditVV)
2. HotelManager (hotelManagerID, hmUsername, hmPassword, hmFirstName, hmLastName, hmEmail, hmAddress, hmMobileNum)
3. Room (roomNum, roomType, floorLevel)
4. Booking (bookingID, arrivalDate, departureDate, requestedRoomType, bookingDate, bookingStatus, customerID\*, hotelManagerID\*, roomNum\*)

## **Part B: Data Dictionary**

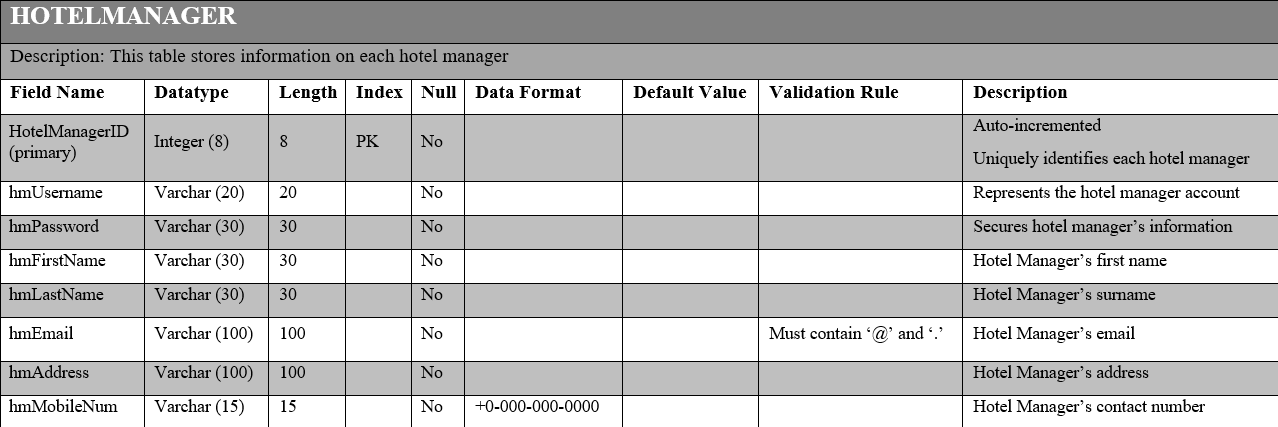


**Figure 3.1: Data Dictionary Showing Full Details of Customer Table**

Index



**Figure 3.1.2: Table Showing the Indexes of Customer Table**

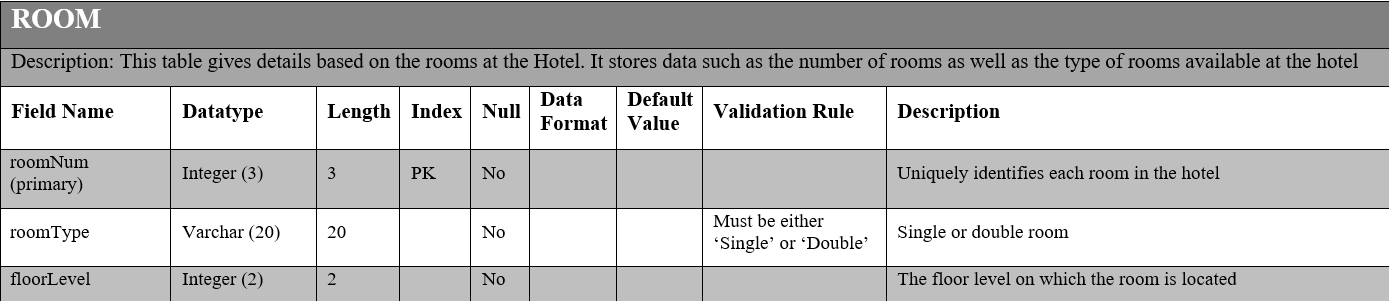


**Figure 3.2.1: Data Dictionary Showing Full Details on the Hotel Manager Table**

Index



**Figure 3.2.2: Table Showing the Indexes of Hotel Manager Table**

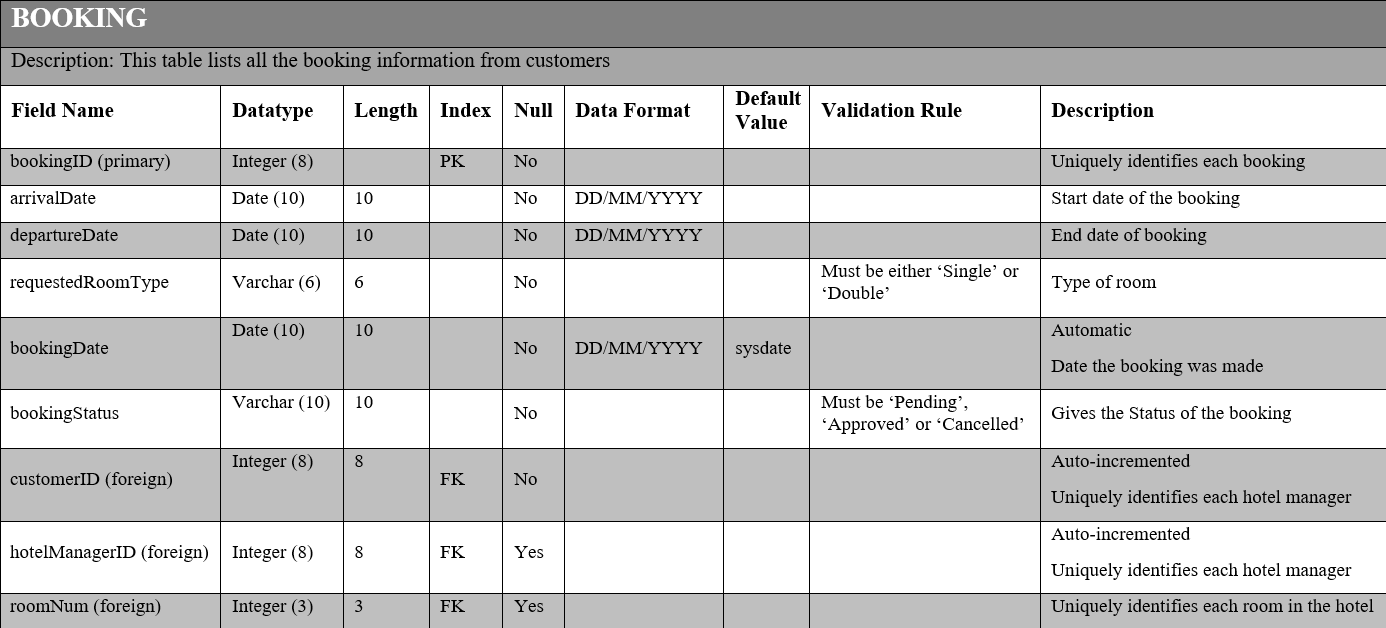


**Figure 3.3.1: Data Dictionary Showing Full Details on the Room Table**

Index



**Figure 3.3.2: Table Showing the Indexes of Room Table**



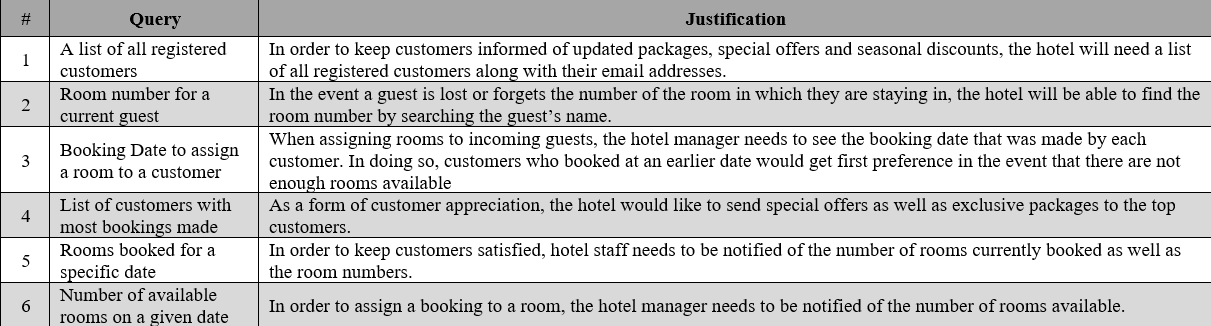
**Figure 3.4.1: Data Dictionary Showing Full Details of Booking Table**



**Figure 3.4.2: Table Showing the Indexes of the Booking Table**

# **Task 4: Query Design**

## **Part A: List of Queries and their Purpose**



**Figure 4: Table Showing SQL Queries to be Implemented in the Database**

## **Part B: SQL Queries**

### List of All Registered Customers

**SELECT DISTINCT** customerID, cFirstName, cLastName, cEmail

**FROM** customer

### Room Number for Guest Staying at Hotel

**SELECT DISTINCT** c.customerID, cFirstName, cLastName, b.roomNum, floorLevel

**FROM** customer c, booking b, room r

**WHERE** b.customerID = c.customerID **AND** b.roomNum = r.roomNum

**AND** bookingStatus <> 'Cancelled'

**AND** '26/12/2020' BEWTWEEN arrivalDate AND departureDate

**AND** cFirstName = 'Kenya'

**AND** cLastName = 'Denoon'

### Booking Date to assign a room to a customer

**SELECT DISTINCT** c.customerID, cFirstName, cLastName, cEmail, bookingID, arrivalDate, departureDate, requestedRoomType, bookingDate

**FROM** customer c, booking b

**WHERE** b.customerID = c.customerID

**AND** bookingStatus <> ‘Cancelled’

**AND** arrivalDate = ‘30/12/2020’

**ORDER BY** bookingDate **DESC**

### List of Customers with Most Bookings Made

**SELECT** c.customerID, cFirstName, cLastName, cEmail, count (b.customerID) as TotalBookings

**FROM** customer c, booking b

**WHERE** b.customerID = c.customerID

**AND** bookingStatus <> ‘Cancelled’

**GROUP BY** c.customerID, cFirstName, cLastName

**ORDER BY** TotalBookings **DESC**

### Rooms booked for a Specific Date

**SELECT** b.roomNum, r.roomType, c.customerID, cFirstName, cLastName

**FROM** customer c, booking b, room r

**WHERE** b.customerID = c.customerID **AND** b.roomNum = r.roomNum

**AND** bookingStatus <> ‘Cancelled’

**AND** ‘20/09/2020’ **BETWEEN** arrivalDate **AND** departureDate

**ORDER BY** b.roomNum **ASC**

### Number of Available Rooms on a Given Date

**SELECT** r.roomType, r.roomNum

**FROM** room r

**WHERE** r.roomNum **NOT IN**

(**SELECT DISTINCT** roomNum

**FROM** booking

**WHERE** ‘20/09/2020’ **BETWEEN** arrivalDate **AND** departureDate

**AND** bookingStatus <> ‘Cancelled’)

**ORDER BY** r.roomNum **ASC**, r.roomType

# **Task 5: Implementation of Database**

## **Part A: Creating Tables**

1. Customer

CREATE TABLE customer (

CustomerID INTEGER PRIMARY KEY AUTOINCREMENT,

cUsername VARCHAR (20) NOT NULL UNIQUE,

cPassword VARCHAR (30) NOT NULL UNIQUE,

cFirstName VARCHAR (30) NOT NULL,

cLastName VARCHAR (30) NOT NULL,

cEmail VARCHAR (100) NOT NULL,

cMobileNum VARCHAR (15) NOT NULL UNIQUE,

cHomeNum VARCHAR (15) NOT NULL,

cAddress VARCHAR (100) NOT NULL,

country VARCHAR (30) NOT NULL,

creditCardType INTEGER NOT NULL,

creditCardNum INTEGER NOT NULL,

ccExpiryDate DATE NOT NULL,

creditVV NOT NULL

);



**Figure 5.1.1: Showing the Customer Table Created in the Database Using SQLite**

1. Hotel Manager

CREATE TABLE HotelManager (

hotelManagerID INTEGER PRIMARY KEY AUTOINCREMENT,

hmUsername VARCHAR (20) NOT NULL UNIQUE,

hmPassword VARCHAR (30) NOT NULL UNIQUE,

hmFirstName VARCHAR (30) NOT NULL,

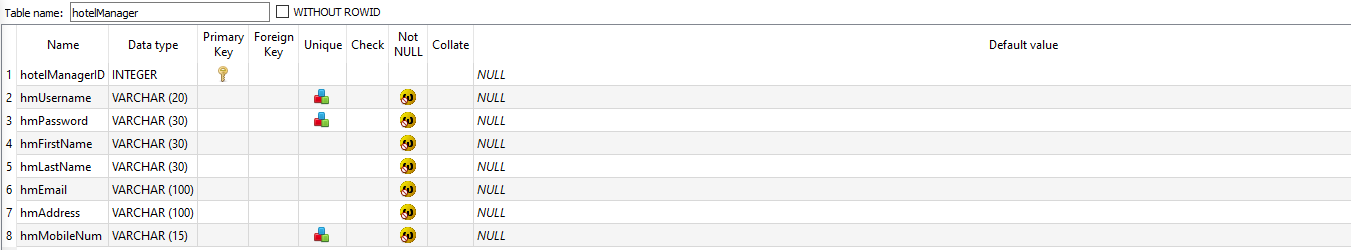
hmLastName VARCHAR (30) NOT NULL,

hmEmail VARCHAR (100) NOT NULL,

hmAddress VARCHAR (100) NOT NULL,

hmMobileNum VARCHAR (15) NOT NULL UNIQUE

);



**Figure 5.1.2 Showing the Hotel Manager Table Created in the Database Using SQLite**

1. Room

CREATE TABLE room (

roomNum INTEGER PRIMARY KEY,

roomType VARCHAR (20) NOT NULL,

floorLevel INTEGER NOT NULL

);



**Figure 5.1.3 Showing the Room Table Created in the Database Using SQLite**

1. Booking

CREATE TABLE booking (

bookingID INTEGER NOT NULL,

arrivalDate DATE NOT NULL,

departureDate DATE NOT NULL,

requestedRoomType VARCHAR (20) NOT NULL,

bookingDate DATE DEFAULT SYSDATE NOT NULL,

bookingStatus VARCHAR (10) NOT NULL,

customerID INTEGER NOT NULL,

hotelManagerID INTEGER,

roomNum INTEGER,

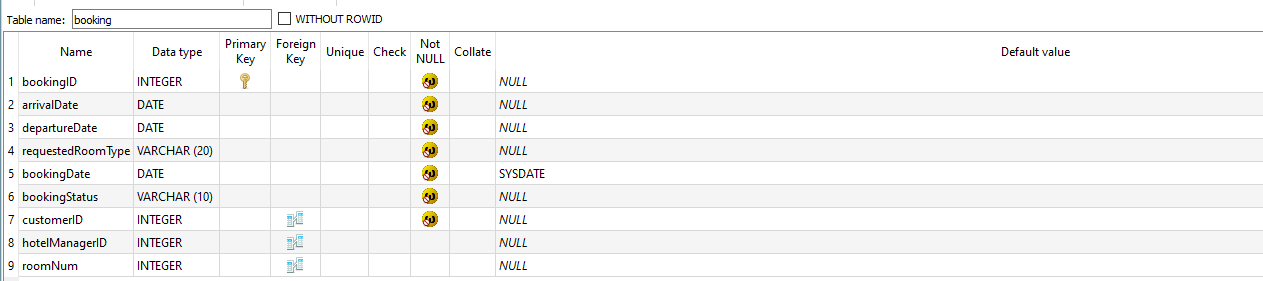
PRIMARY KEY (bookingID),

FOREIGN KEY (customerID) REFERENCES customer (customerID),

FOREIGN KEY (hotelManagerID) REFERENCES HotelManager (hotelManagerID),

FOREIGN KEY (roomNum) REFERENCES room (roomNum)

);



**Figure 5.1.4 Showing the Booking Table Created in the Database Using SQLite**

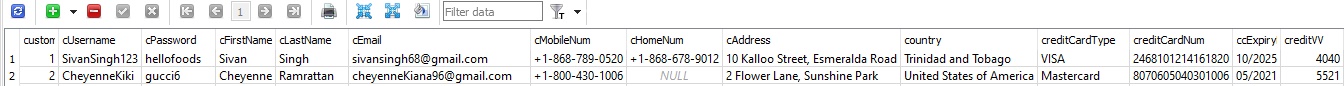
## **Part B: Inserting Data Into Tables**

1. Customer

INSERT INTO customer

VALUES (1, ‘SivanSingh123’, ‘hellofoods’, ‘Sivan’, ‘Singh’, ‘sivansingh68@gmail.com’, ‘+1-868-789-0520’, ‘+1-868-678-9012’, ’10 Kalloo Street, Esmeralda Road’, ‘Trinidad and Tobago’, ‘VISA’, 24681012141620, ‘10/2025’, 4040),

(2, ‘CheyenneKiki’, ‘gucci6’, ‘Cheyenne’, ‘Ramrattan’, ‘cheyennekiana96@gmail.com’, ‘+1-800-430-1006’, null, ‘2 Flower Lane, Sunshine Park’, ‘United States of America’, ‘Mastercard’, 8070605040301006, ‘05/2021’, 5521);



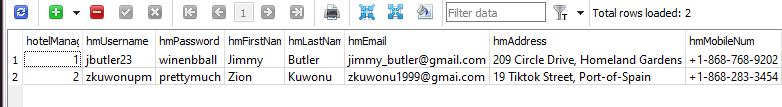
**Figure 5.2.1 Showing Records Inserted into Customer Table**

1. Hotel Manager

INSERT INTO HotelManager

VALUES (1, ‘jbutler’, ‘winenbball’, ‘Jimmy’, ‘Butler’, ‘jimmy\_butler@gmail.com’, ‘209 Circle Drive, Homeland Gardens’, ‘+1-868-768-9202’),

(2, ‘zkuwonupm’, ‘prettymuch’, ‘Zion’, ‘Kuwonu’, ‘ [zkuwonu1996@gmail.com](mailto:zkuwonu1996@gmail.com)’, ‘ 19 Tiktok Street, Port-of-Spain’, ‘+1-868-2833454’);

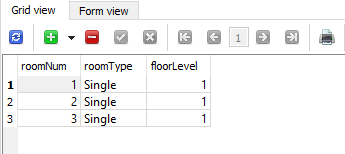


**Figure 5.2.2 Showing Records Inserted into Hotel Manager Table**

1. Room

INSERT INTO room

VALUES (1, ‘Single’, 1), (2, ‘Single’, 1), (3, ‘Single’, 1);



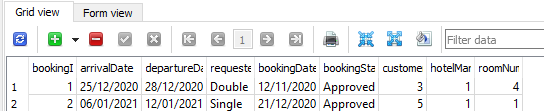
**Figure 5.2.3 Showing Records Inserted into Room Table**

1. Booking

INSERT INTO booking

VALUES (1, ‘25/12/2020’, ‘28/12/2020’, ‘Double’, ‘12/11/2020’, ‘Approved’, 3, 1,,4),

(2, ‘06/01/2021’, ‘12/01/2021’, ‘Single’, ‘21/12/2020’, ‘Approved’, 5, 1, 1);



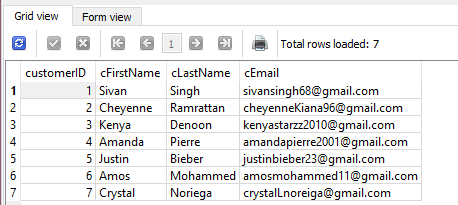
**Figure 5.2.4 Showing Records Inserted Into Booking Table**

## **PART C: SQLite Queries**

### List of All Registered Customers

**SELECT DISTINCT** customerID, cFirstName, cLastName, cEmail

**FROM** customer



**Figure 5.3.1 Showing the Result of the Query Listing All Registered Customers**

### Room Number for Guest Staying at Hotel

**SELECT DISTINCT** c.customerID, cFirstName, cLastName, b.roomNum, floorLevel

**FROM** customer c, booking b, room r

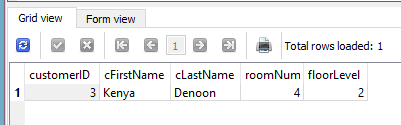
**WHERE** b.customerID = c.customerID **AND** b.roomNum = r.roomNum

**AND** bookingStatus <> 'Cancelled'

**AND** '26/12/2020' BEWTWEEN arrivalDate AND departureDate

**AND** cFirstName = 'Kenya'

**AND** cLastName = 'Denoon'



**Figure 5.3.2 Showing the Result of the Query Revealing the Name of a Current Guest at the Hotel**

### Booking Date to assign a room to a customer

**SELECT DISTINCT** c.customerID, cFirstName, cLastName, cEmail, bookingID, arrivalDate, departureDate, requestedRoomType, bookingDate

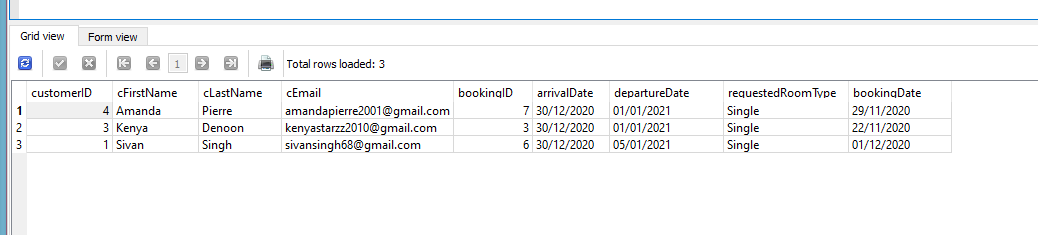
**FROM** customer c, booking b

**WHERE** b.customerID = c.customerID

**AND** bookingStatus <> ‘Cancelled’

**AND** arrivalDate = ‘30/12/2020’

**ORDER BY** bookingDate **DESC**

****

**Figure 5.3.3 showing the Result of the Query, which shows the Booking Dates of customers for Hotel Managers to assign Bookings**

### List of Customers with Most Bookings Made

**SELECT** c.customerID, cFirstName, cLastName, cEmail, count (b.customerID) as TotalBookings

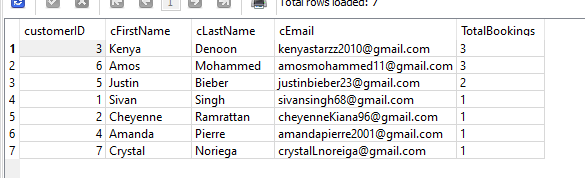
**FROM** customer c, booking b

**WHERE** b.customerID = c.customerID

**AND** bookingStatus <> ‘Cancelled’

**GROUP BY** c.customerID, cFirstName, cLastName

**ORDER BY** TotalBookings **DESC**

****

**Figure 5.3.4 showing the Result of the Query showing the Customers with the Most Bookings Made**

### Rooms booked for a Specific Date

**SELECT** b.roomNum, r.roomType, c.customerID, cFirstName, cLastName

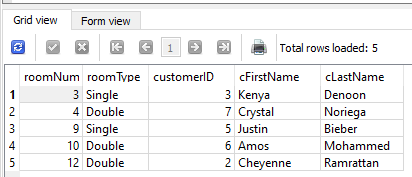
**FROM** customer c, booking b, room r

**WHERE** b.customerID = c.customerID **AND** b.roomNum = r.roomNum

**AND** bookingStatus <> ‘Cancelled’

**AND** ‘20/09/2020’ **BETWEEN** arrivalDate **AND** departureDate

**ORDER BY** b.roomNum **ASC**



**Figure 5.3.5 showing the Result of a Query for Rooms booked at a Specific Date**

### Number of Available Rooms on a Given Date

**SELECT** r.roomType, r.roomNum

**FROM** room r

**WHERE** r.roomNum **NOT IN**

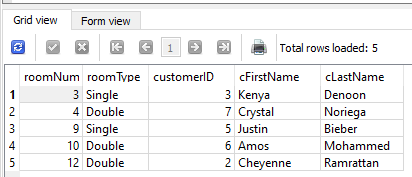
(**SELECT DISTINCT** roomNum

**FROM** booking

**WHERE** ‘20/09/2020’ **BETWEEN** arrivalDate **AND** departureDate

**AND** bookingStatus <> ‘Cancelled’)

**ORDER BY** r.roomNum **ASC**, r.roomType



**Figure 5.3.6 showing the Result of a Query for the Number of Rooms Available on a given Date**

# **Discussion and Reflection**

Assignment CIS017-1 aimed at designing, developing and implementing a database system to manage hotel bookings, which facilitates storage as well as easy accessibility to details regarding customers, rooms, bookings and hotel managers. In order to ensure full comprehension on the topic of databases, the student received several tasks to execute.

Using the Entity Relationship Model, the student was mainly able to identify the relationships among the tables being used. The normalised table helped to prevent data redundancy and enhance data integrity. The table design displayed all tables along with their attributes, identifying foreign keys and primary keys. The data dictionary disclosed information regarding the properties and indexes of each table used in the database. This would later be useful when creating tables in the Database.

By incorporating all tasks described above, the student was able to create queries related to the database. However, in order to manage time more efficiently, the student chose to implement the database before completing Task 4. While the student manually created tables and inserted data, she felt more capable of brainstorming ideas for query design, as she was aware of the data that was inserted. Normalising tables also presented itself as a challenge to the student. However, by knowing the result of the third normalised table using the Entity Relationship Model, the student was able to work backwards and complete the remainder of the table.