CIS017-1 – Computer Systems Structure
CIS095-1 – Databases and Computer Networking 2021-2022
Assignment 1 – Design and Implement a Database

INDIVIDUAL REPORT

Student ID: 2146504
BY : Baibhav Paudel

Table of Contents

Itroduction/Overview

The purpose of the project is to study the fundamentals and structure of the database management system. Given project aid to acquire fundamental analytical and problem-solving abilities, it is important to collect information from various sources and utilize it to design, build, and implement a database system in order to complete the project.

Hotel Provide the data, scenario, and instructions, and a database management system is developed using the following data so that hotel personnel may manage data keeping the registration data, along with the payment data of each customer. The major purpose is to offer the hotel with a full hotel management system as well as a database system for managing all client data.

Following the instructions, entity relationship mode is defined, data normalization is performed, physical tables are designed, a query design is created, and the database system is implemented.

Task Description

A hotel is preparing to launch an online service that will allow customers to check in, book, and plan their next vacation. where customers must register in order to make a reservation

A customer may make multiple reservations, but each reservation can only reserve one room at a time. The orders are not verified until they have been approved by the hotel management. Users have complete control over their orders until they are used, even after they have been verified.

Assumption

- The credit card information is used to guarantee the reservation and payment.
- The authorization of the hotel Administrator is somewhat higher to the authorization of the hotel Personnel.
- Customer account control over their own bookings and hotel employee account control over all customer data where Personnel is permitted to access the info and the Administrator is authorized to both read and update the data.
- Bill is generated for each booking.

Tasks:

Conceptual / Logical Database Design

- **Entity Relationship Model**

Researched From tutorialspoint (2022)

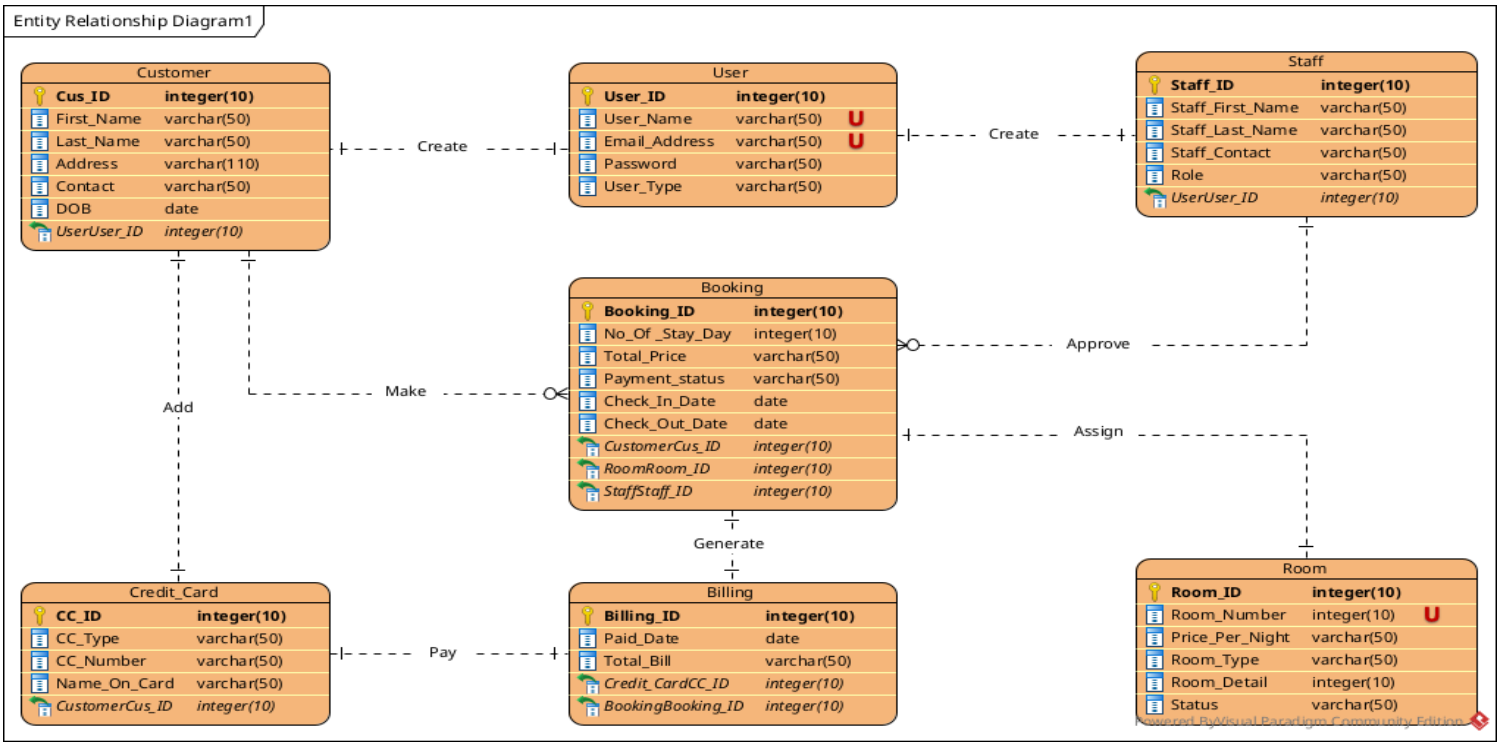

Figure 1: Entity Relationship Model for Hotel Management System

The following entities were defined for the Hotel Booking System: User, Customer, Staff, Credit_Card, Booking, Room and Billing.

Because the Customer and Staff must register in order to use the system, the primary key of User (i.e. User_Id) becomes a foreign key (UserUser_ID*) for them, resulting in a One-to-One relationship because each type of user can only create one account for them.

Customer have control over their own reservations, and Staff have control over all customer Booking where Personnel has access to the information and the Administrator has the permission to view, change, and approve the Booking. As a result, the primary key of Customer(Cus_ID) and Staff(Staff_ID) becomes a foreign key for the Booking (CustomerCus_ID*) and (StaffStaff_ID) , resulting in a one-to-many relationship because each Customer can make many booking as well as staff can access many booking data.

Since only one room can be booked during booking process, they immediately construct a one-to-one relationship by making the Room primary key (i.e. Room_ID) as a foreign key (i.e. RoomRoom_ID*) to the Booking .

Customer Credit Card detail create one to one relation wuth Customer.

While billing bill cointain Credit_Card detail as well as booking detail using Credit_Card(CC_ID) and Booking(Booking_ID) Primary key as foreign(Credit_CardCC_ID) and (BookingBooking_ID) key to Billing.

- ## Normalisation – Hotel Booking System

According to javatpoint (2011-2020)

"A relation will be 1NF if it contains an atomic value.It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

In the 2NF, relational must be in 1NF.

In the second normal form, all non-key attributes are fully functional dependent on the primary key

A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.

3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form."

UNF	1NF	2NF	3NF
User_ID	**Cus_ID**	**Cus_ID**	**Cus_ID**
User_Name	First_Name	First_Name	First_Name
Email_Address	Last_Name	Last_Name	Last_Name
Password	Address	Address	Address
User_Type	Contact	Contact	Contact
Cus_Id	DOB	DOB	DOB
First_Name			UserUser_ID*
Last_Name	**Booking_ID**	**Booking_ID**	
Address	No_Of_Stay_Day	No_Of_Stay_Day	**Staff_ID**
Contact	Total_Price	Total_Price	Staff_First_Name
DOB	Payment_Status	Payment_Status	Staff_Last_Name
Staff_ID	Check_In_Date	Check_In_Date	Staff_Contact
Staff_First_Name	Check_Out_Date	Check_Out_Date	Role
	Room_ID	Room_ID	UserUser_ID*
Staff_Contact	Room_Number	Room_Number	
Role	Price_Per_Night	Price_Per_Night	**User_ID**
CC_ID	Room_Type	Room_Type	User_Name
CC_Type	Room_Detail	Room_Detail	Email_Address
CC_Number	CC_ID	CC_ID	Password
Name_On_Card	CC_Type	CC_Type	User_Type
Booking_ID	CC_Number	CC_Number	
No_Of_Stay_Day	Name_On_Card	Name_On_Card	**CC_ID**
Total_Price	Billing_ID	Billing_ID	CC_Type
Payment_Status	Paid_Data	Paid_Data	CC_Number
Check_In_Date	Total_Bill	Total_Bill	Name_On_Card
Check_Out_Date	StaffStaff_ID	StaffStaff_ID	CustomerCus_ID*
Room_ID	User_ID	User_ID	
Room_Number	User_Name	User_Name	**Room_ID**
Price_Per_Night	Email_Address	Email_Address	Room_Number
Room_Type	Password	Password	Price_Per_Night
Room_Detail	User_Type	User_Type	Room_Type
Status	CustomerCus_Id*	CustomerCus_Id*	Room_Detail

Billing_ID	Staff_ID	Staff_ID	
Paid_Data	Staff_First_Name	Staff_First_Name	**Booking_ID**
Total_Bill	Staff_Last_Name	Staff_Last_Name	No_Of_Stay_Day
	Staff_Contact	Staff_Contact	Total_Price
	Role	Role	Payment_Status
			Check_In_Date
			Check_Out_Date
			CustomerCus_ID*
			RoomRoom_ID*
			StaffStaff_ID*
			Billing_ID
			Paid_Data
			Total_Bill
			Credit_CardCC_ID*
			BookingBooking_ID*

Table 1: Normalisation Table

Physical Database Design

- Skeleton Tables

Customer (**Cus_ID,** First_Name, Last_Name, Contact, Address, DOB, UserUser_ID*)

Staff (**Staff_ID,** Staff_First_Name, Staff_Last_Name, Staff_Contact, Role, UserUser_ID*)

User (**User_ID**, User_Name, Email_Address, Password, User_Type)

Credit_Card (**CC_ID**, CC_Type, CC_Number, Name_On_Card, CustomerCus_ID)

Booking (**Booking_ID**, No_Of_Stay_Day, Total_Price, Payment_status, Check_In_Date, Check_Out_Date, CustomerCus_ID*, RoomRoom_ID*, StaffStaff_ID*)

Room (**Room_ID**, Room_Number, Price_Per_Night, Room_Type, Room_Detail, Status)

Billing (**Billing_ID**, Paid_Date, Total_Bill, Credit_CardCC_ID*, BookingBooking_ID*)

- Data Dictionary (for each table)

Customer							
Description: Customer details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Cus_ID	integer	10	PK	NO			
First_Name	varchar	50		NO			
Last_Name	varchar	50		NO			
Address	varchar	110		NO			
Contact	varchar	50		NO			
DOB	date			NO			
UserUser_ID*	integer	10	FK	NO			

Table 2: Customer Table

Indexes

Keyname	**Type**	**Unique**	**Column**	**Null**
PRIMARY	integer	Yes	**Cus_ID**	No
FOREIGN	integer	Yes	UserUser_ID*	No

Staff							
Description: Staff details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Staff_ID	integer	10	PK	NO			
Staff_First_Name	varchar	50		NO			
Staff_Last_Name	varchar	50		NO			
Staff_Contact	varchar	50		NO			
Role	varchar	50		NO			
UserUser_ID*	integer	10	FK	NO			

Table 3: Staff Table

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	integer	Yes	**Staff_ID**	No
FOREIGN	integer	Yes	UserUser_ID*	No

User							
Description: User details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
User_ID	integer	50	PK	NO			
User_Name	varchar	50	Unique	NO			
Email_Address	varchar	50	Unique	NO			
Password	varchar	50		NO			
User_Type	varchar	50		NO			

Table 4: User Table

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	integer	Yes	**User_ID**	No

Credit_Card							
Description: Credit_Card details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
CC_ID	integer	10	PK	NO			
CC_Type	varchar	50		NO			
CC_Number	varchar	50		NO			
Name_On_Card	varchar	50		NO			

CustomerCus_ID*	integer	10	FK	NO			

Table 5: Customer Table

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	integer	Yes	**CC_ID**	No
FOREIGN	integer	Yes	CustomerCus_ID *	No

Booking

Description: Booking details

Field Name	Datatype	Length	Index	Null	Default	Validation rule	Description
Booking_ID	integer	10	PK	NO			
No_Of_Stay_Day	integer	10		NO			
Total_Price	varchar	50		NO			
Payment_Status	varchar	50		NO			
Check_In_Date	date			NO			
Check_Out_Date	date			NO			
CustomerCus_ID*	integer	10	FK	NO			
RoomRoom_ID*	integer	10	FK	NO			
StaffStaff_ID*	integer	10	FK	NO			

Table 6: Booking Table

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	integer	Yes	**Booking_ID**	No
FOREIGN	integer	Yes	CustomerCus_ID *	No

FOREIGN	integer	Yes	RoomRoom_ID*	No
FOREIGN	integer	Yes	StaffStaff_ID*	No

Billing							
Description: Billing details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Billing_ID	integer	10	PK	NO			
Paid_Data	date			NO			
Total_Bill	varchar	50		NO			
Credit_CardCC_ID*	integer	10	FK	NO			
BookingBooking_ID*	integer	10	FK	NO			

Table 7: Billing Table

Indexes

Keyname	**Type**	**Unique**	**Column**	**Null**
PRIMARY	integer	Yes	**Billing_ID**	No
FOREIGN	integer	Yes	Credit_CardCC_ID*	No
FOREIGN	integer	Yes	BookingBooking_ID*	No

Room							
Description: Room details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Room_ID	integer	10	PK	NO			
Room_Number	integer	50	Unique	NO			
Price_Per_Night	varchar	50		NO			

Room_Type	varchar	50		NO			
Room_Detail	varchar	10		NO			
Status	Varchar	50		NO			

Table 8: Room table

Keyname	Type	Unique	Column	Null
PRIMARY	integer	Yes	**Room_ID**	No

- Implementation of the Database

  Learned From Oracle(2020)

MySQL RDMS is used for execution of SQL statements.

```
CREATE TABLE User(
  User_ID INT(10) PRIMARY KEY,
  User_Name VARCHAR(50) NOT NULL UNIQUE,
  Email_Address VARCHAR(50) NOT NULL UNIQUE,
  Password VARCHAR(50) NOT NULL,
  User_Type VARCHAR(50) NOT NULL
  );
```

Figure 2: Created User

SQL Statement to create table for User.SQL Statement to insert values in User.

```
INSERT INTO User(
  User_ID, User_Name, Email_Address, Password, User_Type )
  VALUES
  (1001, 'bwilliams', 'bwilliams@gmail.com', 'bwilliams123%','Customer'),
  (1002, 'abrown', 'abrown@gmail.com', 'abrown123%','Customer'),
  (1003, 'psmith', 'psmith@gmail.com', 'psmith123%','Customer' ),
  (1004, 'kevans', 'kevans@gmail.com', 'kevans123%','Customer'),
  (1005, 'ssmith', 'ssmith@gmail.com', 'ssmith123%','Administrator'),
  (1006, 'kJones', 'kJones@gmail.com', 'kJones123%','Customer'),
  (1007, 'jbrown', 'jbrown@gmail.com', 'jbrown123%','Customer'),
  (1008, 'jtaylor', 'jtaylor@gmail.com', 'jtaylor123%','Personnel'),
  (1009, 'rwilliams', 'rwilliams@gmail.com', 'rwilliams123%','Personnel'),
  (1010, 'sgreen', 'sgreen@gmail.com', 'sgreen123%','Customer');
```

Figure 3: Insert User Data

Result for User sucessfully created table.

```
Database changed
MariaDB [hotel_management]> SELECT * FROM User;
+---------+-----------+--------------------+---------------+---------------+
| User_ID | User_Name | Email_Address      | Password      | User_Type     |
+---------+-----------+--------------------+---------------+---------------+
|    1001 | bwilliams | bwilliams@gmail.com | bwilliams123% | Customer      |
|    1002 | abrown    | abrown@gmail.com    | abrown123%    | Customer      |
|    1003 | psmith    | psmith@gmail.com    | psmith123%    | Customer      |
|    1004 | kevans    | kevans@gmail.com    | kevans123%    | Customer      |
|    1005 | ssmith    | ssmith@gmail.com    | ssmith123%    | Administrator |
|    1006 | kJones    | kJones@gmail.com    | kJones123%    | Customer      |
|    1007 | jbrown    | jbrown@gmail.com    | jbrown123%    | Customer      |
|    1008 | jtaylor   | jtaylor@gmail.com   | jtaylor123%   | Personnel     |
|    1009 | rwilliams | rwilliams@gmail.com | rwilliams123% | Personnel     |
|    1010 | sgreen    | sgreen@gmail.com    | sgreen123%    | Customer      |
+---------+-----------+--------------------+---------------+---------------+
10 rows in set (0.001 sec)
```

Figure 4: User Data Output

SQL Statement to create table for Customer.

```
CREATE TABLE Customer (
 Cus_ID INT(10) PRIMARY KEY,
 First_Name VARCHAR(50) NOT NULL,
 Last_Name VARCHAR(50) NOT NULL,
 Address VARCHAR(110) NOT NULL,
 Contact VARCHAR(50) NOT NULL,
 DOB date NOT NULL,
 UserUser_ID INT(10) NOT NULL,
 FOREIGN KEY (UserUser_ID) REFERENCES User(User_ID)
 );
```

Figure 5: Create Customer

SQL Statement to insert values in Customer Table.

```
INSERT  INTO Customer(
 Cus_ID, First_Name, Last_Name, Address, Contact, DOB, UserUser_ID
 )
 VALUES
 (1, 'Brody',    'Williams', 'Baltasound-68 Ballifeary Road',    '079 2556 3181',    '1996-08-20', 1001),
 (2, 'Andrew',   'Brown',    'Wilby-68 Ramsgate Rd', '079 5528 4239',    '1979-03-10', 1002),
 (3, 'Peter',    'Smith',    'Underriver-95 Helland Bridge', '079 7622 4215',    '1988-09-08', 1003),
 (4, 'Krish',    'Evans',    'Nonington-25 Cambridge Road',  '079 7622 6544',    '1980-09-05', 1004),
 (5, 'Ken',  'Jones',    'Wilby-68 Ramsgate Rd', '079 4457 5477',    '1959-08-28', 1006),
 (6, 'Julia',    'Brown',    'Frith Bank-61 Golden Knowes Road', '079 5547 6621',    '1976-03-24', 1007),
 (7,'Sanaya',    'Green',    'Underriver-95 Helland Bridge', '079 1754 6847',    '1979-11-07', 1010);
```

Figure 6: Insert Customer Data

Result for sucessfully Customer table.

```
MariaDB [hotel_management]> SELECT * FROM Customer;
+--------+------------+-----------+---------------------------------+---------------+------------+-------------+
| Cus_ID | First_Name | Last_Name | Address                         | Contact       | DOB        | UserUser_ID |
+--------+------------+-----------+---------------------------------+---------------+------------+-------------+
|      1 | Brody      | Williams  | Baltasound-68 Ballifeary Road   | 079 2556 3181 | 1996-08-20 |        1001 |
|      2 | Andrew     | Brown     | Wilby-68 Ramsgate Rd            | 079 5528 4239 | 1979-03-10 |        1002 |
|      3 | Peter      | Smith     | Underriver-95 Helland Bridge    | 079 7622 4215 | 1988-09-08 |        1003 |
|      4 | Krish      | Evans     | Nonington-25 Cambridge Road     | 079 7622 6544 | 1980-09-05 |        1004 |
|      5 | Ken        | Jones     | Wilby-68 Ramsgate Rd            | 079 4457 5477 | 1959-08-28 |        1006 |
|      6 | Julia      | Brown     | Frith Bank-61 Golden Knowes Road| 079 5547 6621 | 1976-03-24 |        1007 |
|      7 | Sanaya     | Green     | Underriver-95 Helland Bridge    | 079 1754 6847 | 1979-11-07 |        1010 |
+--------+------------+-----------+---------------------------------+---------------+------------+-------------+
7 rows in set (0.001 sec)
```

Figure 7: Customer Data Output

SQL Statement to create table for Staff.

```
CREATE TABLE Staff (
  Staff_ID INT(10) PRIMARY KEY,
  staff_First_Name VARCHAR(50) NOT NULL,
  Staff_Last_Name VARCHAR(50) NOT NULL,
  Staff_Contact VARCHAR(50) NOT NULL,
  Role VARCHAR(50) NOT NULL,
  UserUser_ID INT(10) NOT NULL,
  FOREIGN KEY (UserUser_ID) REFERENCES User(User_ID)
  );
```

Figure 8: Created Staff

SQL Statement to insert values in Staff Table.

```
INSERT INTO Staff (
  Staff_ID, staff_First_Name, Staff_Last_Name, Staff_Contact, Role, UserUser_ID
  )
  VALUES
  (101,   'Sam',  'Smith',    '079 2122 5487',    'Administrator', 1005),
  (102,   'Jenet',    'Taylor',   '079 5877 6645',    'Personnel', 1008),
  (103,   'Ruby', 'Williams', '079 2122 2211',    'Personnel', 1009);
```

Figure 9: Insert Staff Data

Result for Staff sucessfully created table.

```
MariaDB [hotel_management]> SELECT * FROM Staff;
+----------+-----------------+----------------+---------------+---------------+-------------+
| Staff_ID | staff_First_Name | Staff_Last_Name | Staff_Contact | Role          | UserUser_ID |
+----------+-----------------+----------------+---------------+---------------+-------------+
|      101 | Sam              | Smith          | 079 2122 5487 | Administrator |        1005 |
|      102 | Jenet            | Taylor         | 079 5877 6645 | Personnel     |        1008 |
|      103 | Ruby             | Williams       | 079 2122 2211 | Personnel     |        1009 |
+----------+-----------------+----------------+---------------+---------------+-------------+
3 rows in set (0.001 sec)
```

Figure 10: Staff Data Output

SQL Statement to create table forRoom.

```
CREATE TABLE Room(
  Room_ID INT(10) PRIMARY KEY,
  Room_Number INT(10) NOT NULL UNIQUE,
  Price_Per_Night VARCHAR(50) NOT NULL,
  Room_Type VARCHAR(50) NOT NULL,
  Room_Detail VARCHAR(50) NOT NULL,
  Status VARCHAR(50) NOT NULL
  );
```

Figure 11: Created Room

SQL Statement to insert values in Room Table.

```
INSERT INTO Room (
  Room_ID, Room_Number, Price_Per_Night, Room_Type, Room_Detail, Status
  )
  VALUES
  (111,   1010,   '2500', 'Double',   'Don\'t have balcony', 'Booked'),
  (222,   2020,   '1500', 'Single',   'have balcony','Booked'),
  (333,   3030,   '2500', 'Double',   'Don\'t have balcony','Open'),
  (444,   4040,   '2500', 'Double',   'Don\'t have balcony','Booked'),
  (555,   5050,   '1500', 'Single',   'have balcony','Booked'),
  (666,   6060,   '2500', 'Double',   'have balcony','Open'),
  (777,   7070,   '1500', 'Single',   'Don\'t have balcony','Reserved'),
  (888,   8080,   '1500', 'Single',   'have balcony','Open'),
  (999,   9090,   '2500', 'Double',   'Don\'t have balcony','Reserved'),
  (1111,  10101,  '1500', 'Single',   'Don\'t have balcony','Reserved');
```

Figure 12: Insert Room Data

Result for Room sucessfully created table.

```
MariaDB [hotel_management]> SELECT * FROM Room;
+---------+-------------+-----------------+-----------+--------------------+----------+
| Room_ID | Room_Number | Price_Per_Night | Room_Type | Room_Detail        | Status   |
+---------+-------------+-----------------+-----------+--------------------+----------+
|     111 |        1010 | 2500            | Double    | Don't have balcony | Booked   |
|     222 |        2020 | 1500            | Single    | have balcony       | Booked   |
|     333 |        3030 | 2500            | Double    | Don't have balcony | Open     |
|     444 |        4040 | 2500            | Double    | Don't have balcony | Booked   |
|     555 |        5050 | 1500            | Single    | have balcony       | Booked   |
|     666 |        6060 | 2500            | Double    | have balcony       | Open     |
|     777 |        7070 | 1500            | Single    | Don't have balcony | Reserved |
|     888 |        8080 | 1500            | Single    | have balcony       | Open     |
|     999 |        9090 | 2500            | Double    | Don't have balcony | Reserved |
|    1111 |       10101 | 1500            | Single    | Don't have balcony | Reserved |
+---------+-------------+-----------------+-----------+--------------------+----------+
10 rows in set (0.001 sec)
```

Figure 13: Room Data Output

SQL Statement to create table for Booking.

```
CREATE TABLE Booking(
  Booking_ID INT(10) PRIMARY KEY,
  No_Of_Stay_Day INT(10) NOT NULL,
  Total_Price VARCHAR(50) NOT NULL,
  Payment_Status VARCHAR(50) NOT NULL,
  Check_IN_Date DATE NOT NULL,
  Check_Out_Date DATE NOT NULL,
  CustomerCus_ID INT(10) NOT NULL,
  RoomRoom_ID INT(10) NOT NULL,
  StaffStaff_ID INT(10) NOT NULL,
  FOREIGN KEY (CustomerCus_ID) REFERENCES Customer(Cus_ID),
  FOREIGN KEY (RoomRoom_ID) REFERENCES Room(Room_ID),
  FOREIGN KEY (StaffStaff_ID) REFERENCES Staff(Staff_ID)
  );
```

Figure 14: Created Booking

SQL Statement to insert values in Booking Table.

```
INSERT INTO Booking (
  Booking_ID, No_Of_Stay_Day, Total_Price, Payment_Status, Check_In_Date,
  Check_Out_Date, CustomerCus_ID, RoomRoom_ID, StaffStaff_ID
  )
  VALUES
  (123,   7, '17,500', 'Pending', '2022-04-21',    '2022-04-28',  1, 111, 101),
  (234,   15,'22,500', 'Paid', '2022-05-02',   '2022-05-17',  2, 222, 101),
  (456,   21, '52,500','Pending', '2022-04-24',    '2022-05-15',  3, 444, 101),
  (567,   12, '8,000', 'Pending','2022-05-12',    '2022-05-24',   4, 555, 101),
  (789,   10, '15,000', 'Paid', '2022-05-09', '2022-05-19',   5, 777, 101),
  (912,   17, '42,500', 'Pending', '2022-04-27', '2022-04-14',    6, 999, 101),
  (1234,  30, '45,000', 'Paid', '2023-06-01', '2023-07-01',   7, 1111, 101);
```

Figure 15: Insert Booking Data

Result Booking for sucessfully created table.

```
MariaDB [hotel_management]> SELECT * FROM Booking;
+------------+---------------+-------------+----------------+--------------+----------------+----------------+-------------+--------------+
| Booking_ID | No_Of_Stay_Day | Total_Price | Payment_Status | Check_IN_Date | Check_Out_Date | CustomerCus_ID | RoomRoom_ID | StaffStaff_ID |
+------------+---------------+-------------+----------------+--------------+----------------+----------------+-------------+--------------+
|        123 |             7 | 17,500      | Pending        | 2022-04-21    | 2022-04-28     |              1 |         111 |          101 |
|        234 |            15 | 22,500      | Paid           | 2022-05-02    | 2022-05-17     |              2 |         222 |          101 |
|        456 |            21 | 52,500      | Pending        | 2022-04-24    | 2022-05-15     |              3 |         444 |          101 |
|        567 |            12 | 8,000       | Pending        | 2022-05-12    | 2022-05-24     |              4 |         555 |          101 |
|        789 |            10 | 15,000      | Paid           | 2022-05-09    | 2022-05-19     |              5 |         777 |          101 |
|        912 |            17 | 42,500      | Pending        | 2022-04-27    | 2022-04-14     |              6 |         999 |          101 |
|       1234 |            30 | 45,000      | Paid           | 2023-06-01    | 2023-07-01     |              7 |        1111 |          101 |
+------------+---------------+-------------+----------------+--------------+----------------+----------------+-------------+--------------+
7 rows in set (0.001 sec)
```

Figure 16: Booking Data Output

SQL Statement to create table for Credit_Card.

```
CREATE TABLE Credit_Card(
 CC_ID INT(10) PRIMARY KEY,
 CC_Type VARCHAR(50) NOT NULL,
 CC_NUmber VARCHAR(50) NOT NULL,
 Name_On_Card VARCHAR(50) NOT NULL,
 CustomerCus_ID INT(10) NOT NULL,
 FOREIGN KEY (CustomerCus_ID) REFERENCES Customer(Cus_ID)
 );
```

Figure 17: Created Credit Card

SQL Statement to insert values in Credit_Card Table.

```
INSERT INTO Credit_Card(
 CC_Type, CC_Number, Name_On_Card, CC_ID, CustomerCus_ID
 )
 VALUES
 ('VISA', '4024007113307606', 'Brody', 7645, 1),
 ('Mastercard', '5167643046551870', 'Andrew', 2587, 2),
 ('Amex', '349759069035542', 'Peter', 4587, 3),
 ( 'VISA', '4556531443487008', 'Krish', 1542, 4),
 ( 'VISA', '4532333266362031', 'Ken' ,5287, 5),
 ( 'Mastercard', '5202689840219096', 'Julia', 6654, 6),
 ( 'Mastercard', '5328173012741426', 'Sanaya', 5437, 7);
```

Figure 18: Insert Credit Card Data

Result Credit_Card for sucessfully created table.

```
MariaDB [hotel_management]> SELECT * FROM Credit_Card;
+-------+------------+------------------+-------------+---------------+
| CC_ID | CC_Type    | CC_NUmber        | Name_On_Card | CustomerCus_ID |
+-------+------------+------------------+-------------+---------------+
|  1542 | VISA       | 4556531443487008 | Krish       |             4 |
|  2587 | Mastercard | 5167643046551870 | Andrew      |             2 |
|  4587 | Amex       | 349759069035542  | Peter       |             3 |
|  5287 | VISA       | 4532333266362031 | Ken         |             5 |
|  5437 | Mastercard | 5328173012741426 | Sanaya      |             7 |
|  6654 | Mastercard | 5202689840219096 | Julia       |             6 |
|  7645 | VISA       | 4024007113307606 | Brody       |             1 |
+-------+------------+------------------+-------------+---------------+
7 rows in set (0.001 sec)
```

Figure 19: Credit Card Data Output

SQL Statement to create table for Billing.

```
CREATE TABLE Billing(
Billing_ID INT(10) PRIMARY KEY,
Paid_Date DATE NOT NULL,
Total_Bill VARCHAR(50) NOT NULL,
Credit_CardCC_ID INT(10) NOT NULL,
BookingBooking_ID INT(10) NOT NULL,
FOREIGN KEY (Credit_CardCC_ID) REFERENCES Credit_Card(CC_ID),
FOREIGN KEY (BookingBooking_ID) REFERENCES Booking(Booking_ID)
);
```

Figure 20: Created Billing

SQL Statement to insert values in Billing Table.

```
INSERT INTO Billing(
Billing_ID, Paid_Date, Total_Bill, Credit_CardCC_ID, BookingBooking_ID
)
VALUES
(2753, '2022-05-17', '22,500', 2587, 234),
(7845, '2022-05-17', '15,000', 5287, 789),
(6578, '2022-05-17', '17,500', 5437, 1234);
```

Figure 21: Insert Billing Data

Result Billing for sucessfully created table.

```
MariaDB [hotel_management]> SELECT * FROM Billing;
+------------+------------+------------+-----------------+------------------+
| Billing_ID | Paid_Date  | Total_Bill | Credit_CardCC_ID | BookingBooking_ID |
+------------+------------+------------+-----------------+------------------+
|       2753 | 2022-05-17 | 22,500     |            2587 |              234 |
|       6578 | 2022-05-17 | 17,500     |            5437 |             1234 |
|       7845 | 2022-05-17 | 15,000     |            5287 |              789 |
+------------+------------+------------+-----------------+------------------+
3 rows in set (0.001 sec)
```

Figure 22: Billing Data Output

- **Query Design**

Researched From W3Schools(1999-2022)

QUERY 1 : All registered users, along with their user type.

Description : Management must recognize if the user who are actual customer.

```
SELECT User_Name, Email_Address
FROM `User` AS U
WHERE User_Type = 'Customer';
```

Figure 23: Query 1

	User_Name	Email_Address
1	bwilliams	bwilliams@gmail.com
2	abrown	abrown@gmail.com
3	psmith	psmith@gmail.com
4	kevans	kevans@gmail.com
5	kJones	kJones@gmail.com
6	jbrown	jbrown@gmail.com
7	sgreen	sgreen@gmail.com

Figure 24: Query 1 Output

QUERY 2: Viewing client, reservation, and payment card information.

Description : Management must be aware of the customer's data in order to accept a booking request. such that following management approval, the reserved booking status switches to booked.

```
SELECT c.Cus_ID, CONCAT(c.First_Name, ' ', c.Last_Name) AS Name, u.Email_Address,
b.Check_In_Date, b.Check_Out_Date, cc.CC_Number,  r.Room_ID, r.status

FROM User u
INNER JOIN Customer c
ON u.User_ID = c.UserUser_ID
INNER JOIN Credit_Card cc
ON cc.CustomerCus_ID = c.Cus_ID
INNER JOIN Booking b
ON b.CustomerCus_ID = c.Cus_ID
INNER JOIN Room r
ON r.Room_ID  = b.RoomRoom_ID
WHERE r.Status = 'Reserved';
```

Figure 25: Query 2

	Cus_ID	Name	Email_Address	Check_In_Date	Check_Out_Date	CC_Number	Room_ID	status
1	5	Ken Jones	kJones@gmail.com	2022-05-09	2022-05-19	4532333266362031	777	Reserved
2	7	Sanaya Green	sgreen@gmail.com	2023-06-01	2023-07-01	5328173012741426	1,111	Reserved
3	6	Julia Brown	jbrown@gmail.com	2022-04-27	2022-04-14	5202689840219096	999	Reserved

Figure 26: Query 2 Output

QUERY 3: Booked room detail and information about the client who booked it

Description : Management must understand their clients' accommodation preferences. for efficient advertising.

```
SELECT c.Cus_ID, CONCAT(c.First_Name, ' ', c.Last_Name) AS Name,
TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) AS Age,
c.Address, b.Check_In_Date, b.No_Of_Stay_Day,
cc.CC_Number, r.Room_Type

FROM Customer c
INNER JOIN Credit_Card cc
ON cc.CustomerCus_ID = c.Cus_ID
INNER JOIN Booking b
ON b.CustomerCus_ID = c.Cus_ID
INNER JOIN Room r
ON r.Room_ID  = b.RoomRoom_ID
WHERE r.Status = 'Booked';
```

Figure 27: Query 3

	Cus_ID	Name	Age	Address	Check_In_Date	No_Of_Stay_Day	CC_Number	Room_Type
1	4	Krish Evans	41	Nonington-25 Cambridge Road	2022-05-12	12	4556531443487008	Single
2	2	Andrew Browr	43	Wilby-68 Ramsgate Rd	2022-05-02	15	5167643046551870	Single
3	3	Peter Smith	33	Underriver-95 Helland Bridge	2022-04-24	21	349759069035542	Double
4	1	Brody Williams	25	Baltasound-68 Ballifeary Road	2022-04-21	7	4024007113307606	Double

Figure 28: Query 3 Output

QUERY 4: The bill comprising the Customer ID, Room Number, Number of Nights Stayed, and Total Bill Generated.

Description : When creating a bill, it must have all necessary information for the customer to understand.

```
SELECT
c.Cus_ID, r.Room_Number, r.Room_Type, r.Price_Per_Night, b.No_Of_Stay_Day,
(b.No_Of_Stay_Day * r.Price_Per_Night) As Total_Bill, b.Payment_Status

FROM Booking b, Customer c, Room r
WHERE
b.CustomerCus_ID  = c.Cus_ID
AND
r.Room_ID = b.RoomRoom_ID
```

Figure 29: Query 4

	Cus_ID	Room_Number	Room_Type	Price_Per_Night	No_Of_Stay_Day	Total_Bill	Payment_Status
1	1	1,010	Double	2500	7	17,500	Pending
2	2	2,020	Single	1500	15	22,500	Paid
3	3	4,040	Double	2500	21	52,500	Pending
4	4	5,050	Single	1500	12	18,000	Pending
5	5	7,070	Single	1500	10	15,000	Paid
6	6	9,090	Double	2500	17	42,500	Pending
7	7	10,101	Single	1500	30	45,000	Paid

Figure 30: Query 4 Output

QUERY 5: Making Booked room status into Open when payment status is Paid. (As bill is paid at the end of booking)

Description : After the customer leaves, the room status must be updated.

```
UPDATE Room
INNER JOIN Booking  ON Booking.RoomRoom_ID  = Room_ID
SET Room.Status  = 'Open'

WHERE Booking.Payment_Status  = 'Paid';
```

Statistics 1 | Output

Name	Value
Updated Rows	3
Query	UPDATE Room INNER JOIN Booking ON Booking.RoomRoom_ID = Room_ID SET Room.Status = 'Open' WHERE Booking.Payment_Status = 'Paid'
Finish time	Mon Aug 15 19:15:54 NPT 2022

Figure 31: Query 5

	Room_ID	Room_Number	Price_Per_Night	Room_Type	Room_Detail	Status
1	111	1,010	2500	Double	Don't have balcony	Booked
2	222	2,020	1500	Single	have balcony	Open
3	333	3,030	2500	Double	Don't have balcony	Open
4	444	4,040	2500	Double	Don't have balcony	Booked
5	555	5,050	1500	Single	have balcony	Booked
6	666	6,060	2500	Double	have balcony	Open
7	777	7,070	1500	Single	Don't have balcony	Open
8	888	8,080	1500	Single	have balcony	Open
9	999	9,090	2500	Double	Don't have balcony	Reserved
10	1,111	10,101	1500	Single	Don't have balcony	Open

Figure 32: Query 5 Output

QUERY 6: Room status, as well as the booked room availability date.

Description : The customer should be able to see all of the room status as well as when their particular room will be available.

```
SELECT r.Room_ID,r.Room_Number, r.Room_Type, DATE_ADD(b.Check_Out_Date , INTERVAL 2 DAY) AS WillAvailableOn
FROM Booking b INNER JOIN
( SELECT * FROM Room
WHERE Status = 'Booked') r
ON r.room_id = b.RoomRoom_ID
```

Figure 33: Query 6

	Room_ID	Room_Number	Room_Type	WillAvailableOn
1	111	1,010	Double	2022-04-30
2	444	4,040	Double	2022-05-17
3	555	5,050	Single	2022-05-26

Figure 34: Query 6 Output

QUERY 7: Find Room number with customer name Sanaya.

Description : The customer must be able to identify their room number from their registered name.

```
SELECT
CONCAT(c.First_Name, ' ', c.Last_Name) AS Name, r.Room_Number
FROM Customer c, Room r, Booking b
WHERE
b.CustomerCus_ID = c.Cus_ID AND
r.Room_ID = b.RoomRoom_ID AND
c.First_Name LIKE '%Sanaya%'
GROUP BY c.First_Name;
```

Figure 35: Query 7

	Name	Room_Number
1	Sanaya Green	10,101

Figure 36: Query 7 Output

- **Discussion / Critical Analysis / Reflection**

To complete the project, information must be gathered from many sources and used to design, construct, and deploy a database system. From research database shown below was established.

While researching potential users of hotel online booking, the user entity was created to keep all registration data for both customers and hotel workers. The following entity contains data such as the user's username, email address, and password. A user entity also has a unique attribute called User Typer that distinguishes between customers and hotel staff.

The second entity, "Customer," was formed to store the personal data of customer type users. The following entities have the customer's name, address, contact information, and date of birth. This information assists hotel management in tracking consumer information.

The third entity was the staff. The Staff must be registered on the hotel system in order to approve the customer's booking request. Staff entities have the same qualities as consumer personal data, but they also have special attributes. The role that represents the registration staff is an administrator or a personeel, and we assume that authorization varies depending on the kind of staff.

Credit Card is the fourth entity in charge of handling the credit card information needed to ensure the booking. Card Type, Card Number, and Name On Card are characteristics of the following entity. Each credit card detail contains the customer information, which is needed to generate a bill.

The Booking Entity is the fifth entity formed, and it is nearly the major entity that maintains all of the facts linked to the customer's booking, such as the number of days intended to stay, payment status, check in date, and check out date. The booking contains all of the information from the client who made the booking, the staff who approved the booking request, and the room details provided to the customer.

When it comes to rooms, the sixth entity was built by combing through all of the data associated with a room, such as the room number, the price of the room per night, the kind of room, and the status, which indicates whether the room is available, reserved, or booked. There is a unique feature. Room
Each room's physical appearance is addressed in this detail.

Finally, once all of the entities have been processed, the final entity is billing, which handles all of the financial data for the user after the payment has been paid. In order to present the customer with comprehensive facts about their produced bill, attributes such as total amount paid and paid date, as well as all credit card and booking information, are used.

Seven entities were built with each requirement in consideration and for better data management.
Following the selection of all potential characteristics, a raw UNF was formed, and data were normalized using the normalization rule, generating possible relationships within themselves.

Following the standardized form of attributes, raw data were prepared from numerous online sources to make the data as real as feasible.

The data dictionary and skeleton table were then created, with the database structure serving as the basic model.The data dictionary and skeleton table were then created, with the database structure serving as the basic model.

When all of the diagram design preparation was finished, it was time to put the database into action. DBeaver as a script editor and MySQL as a database language are used. A table was established and dummy data was added at the same time, as shown in the screenshot and attached to the database implementation section. The success of the code is additionally attched with a suitable output box screenshot.

Seven queries were generated, each with a rationale of why we would need these inquiries to help the hotel flourish in the real world. The queries were chosen to highlight crucial functionality such as join format and sub-query format. I answered all of the questions posed in the assignment.

It was difficult to distinguish entities and attributes because it was my first time constructing a database with a real-world problem. It required some time and effort to get to the assignment's close criteria.

Some prior attempts at an ERM Diagram are included below.

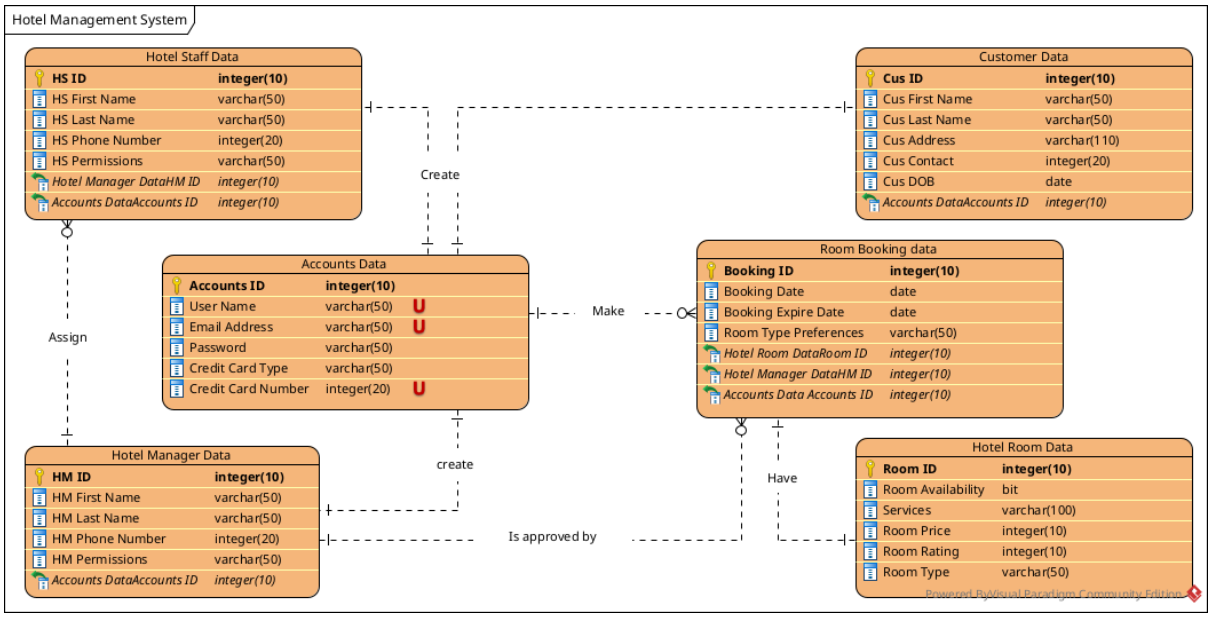

Figure 37: ERM Diagram Try 1

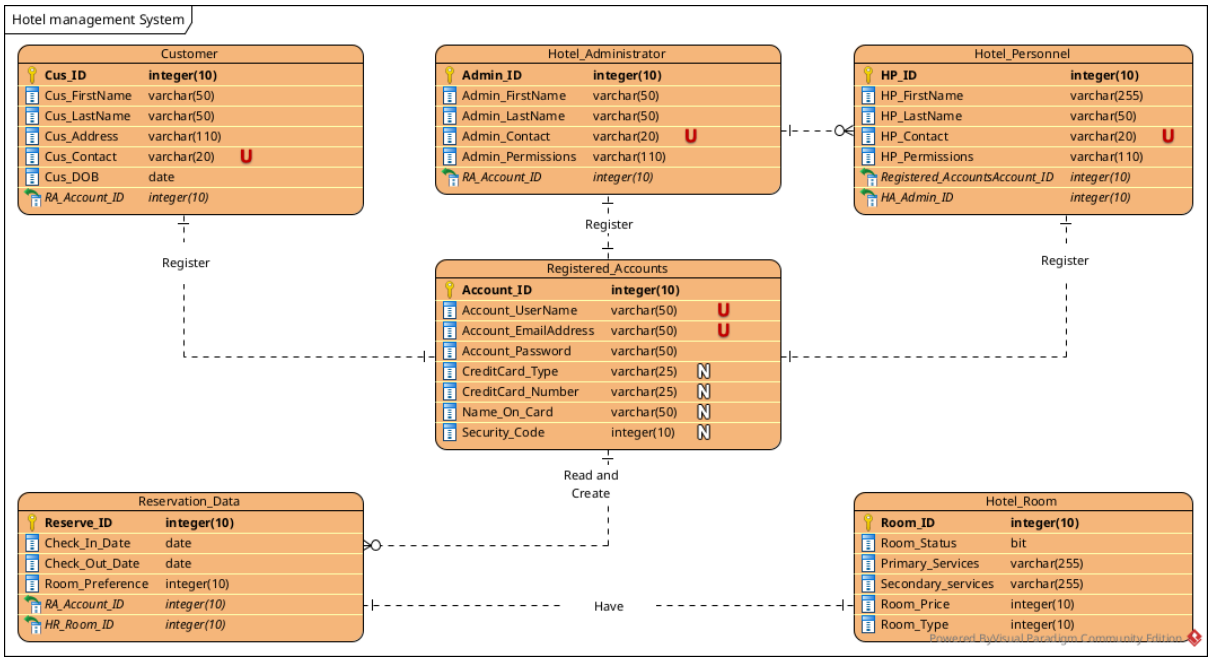

Figure 38: ERM Diagram Try 2

It was quite difficult to be successful on the first try, so as research and learning progressed, the concept changed and the data base was altered . Working with real-world situations was both challenging and thrilling. Finding one's own faults and working as extensively as possible to remedy them via study was a difficult effort, but it felt honest and encouraging. Working with so much at once as an individual was exhausting, but learning and dedicating my time improved my perspective and was really enjoyable.

- ## Conclusion

The overall brief was for a full online hotel administration system. And, the assignment was completed successfully, with all of the requirements met. All of the challenges were solved and all of the assigned tasks were performed via ongoing study and learning. So, the online hotel management system database is now operational.

- ## References

tutorialspoint 2022.ER Diagram Representation. Accessed on: 16 th August, 2022. Retrieved from:https://www.tutorialspoint.com/dbms/er_diagram_representation.htm#

javatpoint 2011-2020. Normalization. Accessed on: 16 th August, 2022. Retrieved from:https://stackoverflow.com/questions/1258743/normalization-in-mysql

Oracle 2022. Loading Data into a Table. Accessed on: 16 th August, 2022. Retrieved from: https://dev.mysql.com/doc/refman/8.0/en/loading-tables.html

w3schools 2022. MySQL SQL. Accessed on: 16 th August, 2022. Retrieved from: https://www.w3schools.com/mysql/mysql_sql.asp

- ## Appendix

#Creating and Inserting Data into Tables

```
CREATE TABLE User(
#Creating User Table
User_ID INT(10) PRIMARY KEY,
User_Name VARCHAR(50) NOT NULL UNIQUE,
Email_Address VARCHAR(50) NOT NULL UNIQUE,
Password VARCHAR(50) NOT NULL,
User_Type VARCHAR(50) NOT NULL
);

INSERT INTO User(
#Inserting data into User Table
User_ID, User_Name, Email_Address, Password, User_Type )
VALUES
(1001, 'bwilliams', 'bwilliams@gmail.com', 'bwilliams123%','Customer'),
(1002, 'abrown', 'abrown@gmail.com', 'abrown123%','Customer'),
(1003, 'psmith', 'psmith@gmail.com', 'psmith123%','Customer' ),
(1004, 'kevans', 'kevans@gmail.com', 'kevans123%','Customer'),
(1005, 'ssmith', 'ssmith@gmail.com', 'ssmith123%','Administrator'),
(1006, 'kJones', 'kJones@gmail.com', 'kJones123%','Customer'),
(1007, 'jbrown', 'jbrown@gmail.com', 'jbrown123%','Customer'),
(1008, 'jtaylor', 'jtaylor@gmail.com', 'jtaylor123%','Personnel'),
(1009, 'rwilliams', 'rwilliams@gmail.com', 'rwilliams123%','Personnel'),
(1010, 'sgreen', 'sgreen@gmail.com', 'sgreen123%','Customer');


CREATE TABLE Customer (
#Creating Customer Table
Cus_ID INT(10) PRIMARY KEY,
First_Name VARCHAR(50) NOT NULL,
Last_Name VARCHAR(50) NOT NULL,
Address VARCHAR(110) NOT NULL,
Contact VARCHAR(50) NOT NULL,
DOB date NOT NULL,
UserUser_ID INT(10) NOT NULL,
FOREIGN KEY (UserUser_ID) REFERENCES User(User_ID)
);

INSERT  INTO Customer(
#Inserting data into Customer Table
Cus_ID, First_Name, Last_Name, Address, Contact, DOB, UserUser_ID
)
VALUES
```

```
(1,     'Brody',     'Williams',    'Baltasound-68 Ballifeary Road',    '079 2556 3181',
        '1996-08-20', 1001),
(2,     'Andrew',    'Brown',       'Wilby-68 Ramsgate Rd',    '079 5528 4239',
        '1979-03-10', 1002),
(3,     'Peter', 'Smith',    'Underriver-95 Helland Bridge',    '079 7622 4215',
        '1988-09-08', 1003),
(4,     'Krish', 'Evans',    'Nonington-25 Cambridge Road',    '079 7622 6544',
        '1980-09-05', 1004),
(5,     'Ken',  'Jones', 'Wilby-68 Ramsgate Rd',    '079 4457 5477',    '1959-08-28',
1006),
(6,     'Julia',  'Brown',    'Frith Bank-61 Golden Knowes Road',    '079 5547 6621',
        '1976-03-24', 1007),
(7,'Sanaya',    'Green',    'Underriver-95 Helland Bridge',    '079 1754 6847',
        '1979-11-07', 1010);


CREATE TABLE Staff (
#Creating Staff Table
Staff_ID INT(10) PRIMARY KEY,
staff_First_Name VARCHAR(50) NOT NULL,
Staff_Last_Name VARCHAR(50) NOT NULL,
Staff_Contact VARCHAR(50) NOT NULL,
Role VARCHAR(50) NOT NULL,
UserUser_ID INT(10) NOT NULL,
FOREIGN KEY (UserUser_ID) REFERENCES User(User_ID)
);

INSERT INTO Staff (
#Inserting data into Staff Table
Staff_ID, staff_First_Name, Staff_Last_Name, Staff_Contact, Role, UserUser_ID
)
VALUES
(101,  'Sam',  'Smith',     '079 2122 5487',    'Administrator', 1005),
(102,  'Jenet', 'Taylor',   '079 5877 6645',    'Personnel', 1008),
(103,  'Ruby', 'Williams',  '079 2122 2211',    'Personnel', 1009);


CREATE TABLE Room(
#Creating Room Table
Room_ID INT(10) PRIMARY KEY,
Room_Number INT(10) NOT NULL UNIQUE,
Price_Per_Night VARCHAR(50) NOT NULL,
Room_Type VARCHAR(50) NOT NULL,
Room_Detail VARCHAR(50) NOT NULL,
Status VARCHAR(50) NOT NULL
);

INSERT INTO Room (
#Inserting data into Room Table
Room_ID, Room_Number, Price_Per_Night, Room_Type, Room_Detail, Status
```

```
)
VALUES
(111,  1010,  '2500', 'Double',       'Don\'t have balcony', 'Booked'),
(222,  2020,  '1500', 'Single',       'have balcony','Booked'),
(333,  3030,  '2500', 'Double',       'Don\'t have balcony','Open'),
(444,  4040,  '2500', 'Double',       'Don\'t have balcony','Booked'),
(555,  5050,  '1500', 'Single',       'have balcony','Booked'),
(666,  6060,  '2500', 'Double',       'have balcony','Open'),
(777,  7070,  '1500', 'Single',       'Don\'t have balcony','Reserved'),
(888,  8080,  '1500', 'Single',       'have balcony','Open'),
(999,  9090,  '2500', 'Double',       'Don\'t have balcony','Reserved'),
(1111, 10101, '1500', 'Single',       'Don\'t have balcony','Reserved');


CREATE TABLE Booking(
#Creating Booking Table
Booking_ID INT(10) PRIMARY KEY,
No_Of_Stay_Day INT(10) NOT NULL,
Total_Price VARCHAR(50) NOT NULL,
Payment_Status VARCHAR(50) NOT NULL,
Check_IN_Date DATE NOT NULL,
Check_Out_Date DATE NOT NULL,
CustomerCus_ID INT(10) NOT NULL,
RoomRoom_ID INT(10) NOT NULL,
StaffStaff_ID INT(10) NOT NULL,
FOREIGN KEY (CustomerCus_ID) REFERENCES Customer(Cus_ID),
FOREIGN KEY (RoomRoom_ID) REFERENCES Room(Room_ID),
FOREIGN KEY (StaffStaff_ID) REFERENCES Staff(Staff_ID)
);

INSERT INTO Booking (
#Inserting data into Booking Table
Booking_ID, No_Of_Stay_Day, Total_Price, Payment_Status, Check_In_Date,
Check_Out_Date, CustomerCus_ID, RoomRoom_ID, StaffStaff_ID
)
VALUES
(123,  7, '17,500', 'Pending', '2022-04-21',    '2022-04-28',   1, 111, 101),
(234,  15,'22,500', 'Paid', '2022-05-02',       '2022-05-17',  2,  222, 101),
(456,  21, '52,500','Pending', '2022-04-24',    '2022-05-15',   3, 444, 101),
(567,  12, '8,000', 'Pending','2022-05-12',    '2022-05-24',    4, 555, 101),
(789,  10, '15,000', 'Paid', '2022-05-09', '2022-05-19',        5,  777, 101),
(912,  17, '42,500', 'Pending', '2022-04-27', '2022-04-14',   6, 999, 101),
(1234, 30, '45,000', 'Paid', '2023-06-01', '2023-07-01',        7, 1111, 101);


CREATE TABLE Credit_Card(
#Creating Credit_Card Table
CC_ID INT(10) PRIMARY KEY,
CC_Type VARCHAR(50) NOT NULL,
CC_NUmber VARCHAR(50) NOT NULL,
```

```
Name_On_Card VARCHAR(50) NOT NULL,
CustomerCus_ID INT(10) NOT NULL,
FOREIGN KEY (CustomerCus_ID) REFERENCES Customer(Cus_ID)
);

INSERT INTO Credit_Card(
#Inserting data into Credit_Card Table
CC_Type, CC_Number, Name_On_Card, CC_ID, CustomerCus_ID
)
VALUES
('VISA', '4024007113307606', 'Brody', 7645, 1),
('Mastercard', '5167643046551870', 'Andrew', 2587, 2),
('Amex', '349759069035542', 'Peter', 4587, 3),
( 'VISA', '4556531443487008', 'Krish', 1542, 4),
( 'VISA', '4532333266362031', 'Ken' ,5287, 5),
( 'Mastercard', '5202689840219096', 'Julia', 6654, 6),
( 'Mastercard', '5328173012741426', 'Sanaya', 5437, 7);


CREATE TABLE Billing(
#Creating Billing Table
Billing_ID INT(10) PRIMARY KEY,
Paid_Date DATE NOT NULL,
Total_Bill VARCHAR(50) NOT NULL,
Credit_CardCC_ID INT(10) NOT NULL,
BookingBooking_ID INT(10) NOT NULL,
FOREIGN KEY (Credit_CardCC_ID) REFERENCES Credit_Card(CC_ID),
FOREIGN KEY (BookingBooking_ID) REFERENCES Booking(Booking_ID)
);

INSERT INTO Billing(
#Inserting data into Billing Table
Billing_ID, Paid_Date, Total_Bill, Credit_CardCC_ID, BookingBooking_ID
)
VALUES
(2753, '2022-05-17', '22,500', 2587, 234),
(7845, '2022-05-17', '15,000', 5287, 789),
(6578, '2022-05-17', '17,500', 5437, 1234);
```

#Queries

```
#Use of normal Query Concept

SELECT User_Name, User_Type
FROM User u
WHERE User_Type = 'Customer';
```

```sql
#Qurey using MySql Functions and INNER JOIN to link between
tables
SELECT c.Cus_ID, CONCAT(c.First_Name, ' ', c.Last_Name) AS
Name,
TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) AS Age,
c.Address, b.Check_In_Date, b.No_Of_Stay_Day,
cc.CC_Number, r.Room_Type

FROM Customer c
INNER JOIN Credit_Card cc
ON cc.CustomerCus_ID = c.Cus_ID
INNER JOIN Booking b
ON b.CustomerCus_ID = c.Cus_ID
INNER JOIN Room r
ON r.Room_ID  = b.RoomRoom_ID
WHERE r.Status = 'Booked';


#Use of normal Query to link between tables
SELECT
c.Cus_ID, r.Room_Number, r.Room_Type, r.Price_Per_Night,
b.No_Of_Stay_Day,
(b.No_Of_Stay_Day * r.Price_Per_Night) As Total_Bill,
b.Payment_Status

FROM Booking b, Customer c, Room r
WHERE
b.CustomerCus_ID  = c.Cus_ID
AND
r.Room_ID = b.RoomRoom_ID


#Use of Update Function
UPDATE Room
INNER JOIN Booking  ON Booking.RoomRoom_ID  = Room_ID
SET Room.Status  = 'Open'

WHERE Booking.Payment_Status  = 'Paid';

SELECT * From Room;
```

```sql
#Using Like Function to find data from given data
SELECT
CONCAT(c.First_Name, ' ', c.Last_Name) AS Name, r.Room_Number
FROM Customer c, Room r, Booking b
WHERE
b.CustomerCus_ID = c.Cus_ID AND
r.Room_ID = b.RoomRoom_ID AND
c.First_Name LIKE '%Sanaya%'
GROUP BY c.First_Name;


#Query using SubQuery concept
SELECT r.Room_ID,r.Room_Number, r.Room_Type,
DATE_ADD(b.Check_Out_Date , INTERVAL 2 DAY) AS WillAvailableOn
FROM Booking b INNER JOIN
( SELECT * FROM Room
WHERE Status = 'Booked') r
ON r.room_id = b.RoomRoom_ID
```