**HOTEL DATABASE SYSTEMS & DESIGN**

**By**

**Michael Kolawole-Idowu**

**Relational Schema**

7 key tables of the database system created from the relational schema:

CUSTOMER (cusID, cusFName, cusLName, last4Digits, cardHolderName, address)

BOOKINGS (bookingID, cusID, checkInDate, checkOutDate, empId)

EMPLOYEE (empId, empFName, empLName, phoneNo)

GUEST (guestsID, Fname, Lname, DOB)

ROOM (RoomNo, RoomType)

ROOM TYPE (RoomType, Room Rate)

BOOKING DETAILS (BookingDetailsNO, bookingID, guestID, RoomNo)

**Create Table Commands**

**Create tables code can be found in the attached submission file “CreateTables.xml”.**

The previously completed normalisation process of the initial schema led to a streamlined table creation process. Through normalisation the tables created allowed for a system which works under condition without duplications or inconsistencies.

When first creating each table, I had to be cautious initially to ensure that the order of creation didn’t provide a conflict. Each table was created in order to allow tables with conflicting foreign keys to be created after the independent table had been created first, for example Room Type was created first and then the room table was created after as the room Table depends on room types.

Order of creation:

1. Room Type
2. Room
3. Employee
4. Customer
5. Guest
6. Bookings
7. Booking Details

Examples of created tables:

2 individual tables (Customer and Employee) which were required to be created before the BookingDetails table which used information from both.

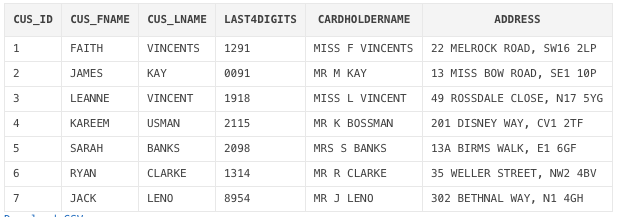


Figure. Customer Table

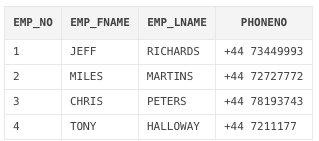


Figure. Employee table

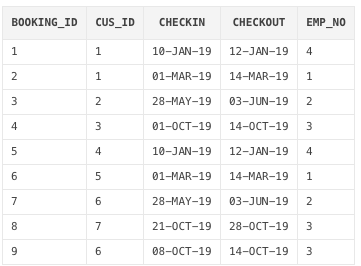


Figure. BookingDetails table

**Insertion of Test Data**

**Insert data SQL code can be found in the attached submission file “Insert.xml”**

Sample data was created with the goal of allowing a variety of situations to be tested via queries. I focused my data to reflect some real-life scenarios, for example I had Guests with multiple bookings, Customers who have made a booking but not stayed, a varied age range etc.

The Primary keys for the Booking, Guest, Customer, Booking Details and employee Tables are all constantly growing, as a result I added a sequence generator for these columns. This allows my tables to be incremented automatically, and lessens administrative workload.

Examples of Insert Data:

1. Customers 1 and 6 have created multiple bookings and so have different Booking ID’s but the same Customer I.

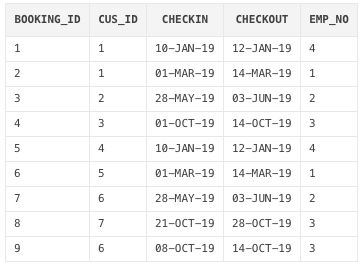


Figure. Bookings table showing variation on bookings

1. Variations of Adult and Child Guests such as Bria Clarke who was born in 1989 and Eve Mathews who was born in 2006

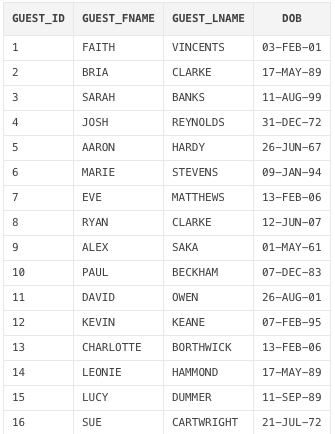
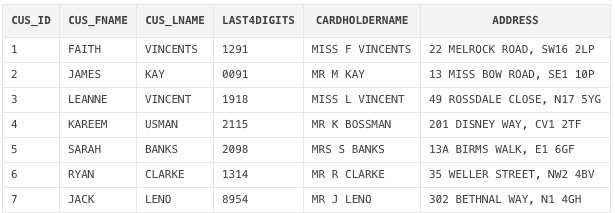


Figure. Bookings tables showing variation on bookings

Customer Kareem Usman has created a booking (so has a unique Customer Id) but did not stay at the hotel (does not have an entry in the Guest table)



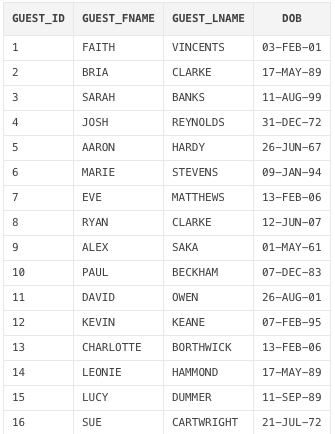


Figure. Highlighting the absence of Kareem Usman from the Guest Table

**Relational Database Views**

**Relational Database Views SQL code can be found in the attached submission file “Views.doc”**

The creation of Relational Database Views assists end users to easily retrieve frequently used data and to access important stored data sets without requiring complex queries. It also helps in creating further complex queries, as complex formulas require multiple combinations and storage. My Views again were selected to solve or display real-life scenarios a Hotel may encounter.

1. **Booking Length View -**

A view was created to show the length of stay for each booking and room stayed at the hotel. This view was useful to create multiple queries and views, as well as to create equations such as, average stay time, most popular rooms etc.

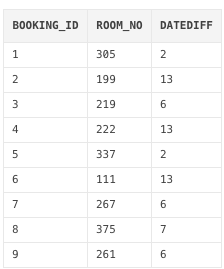


Figure. “Days Stayed” Relational Database View

1. **Extended Booking Details View -**

I created a View for all extended information attached to Guests and their booking. This view would be very useful for hotel staff employees, as it quickly and easily shows which guests stayed at the hotel, how often and what days they stayed.

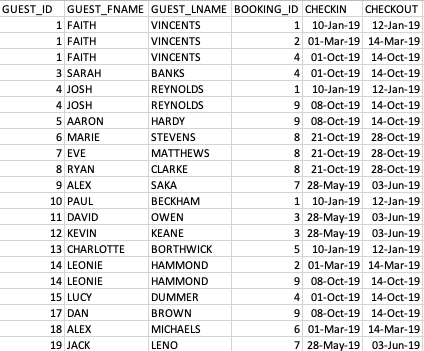
****

Figure. Created View “Extended Booking Details”

1. **Pricing View**

To calculate the pricing of each booking, this view creates and combines many other views together as the price is made up of many components. Firstly, the base price (retrieved from the room type) was added to by the number of adult and/or children staying in the room multiplied by the number of days spent in the room:

Therefore, the pricing View required the creation of several separate database views to build off of.

Two views were then created to highlight the adult and child guests staying at the hotel.

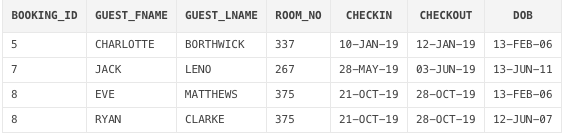


Figure. “Child Guests” Relational Database View

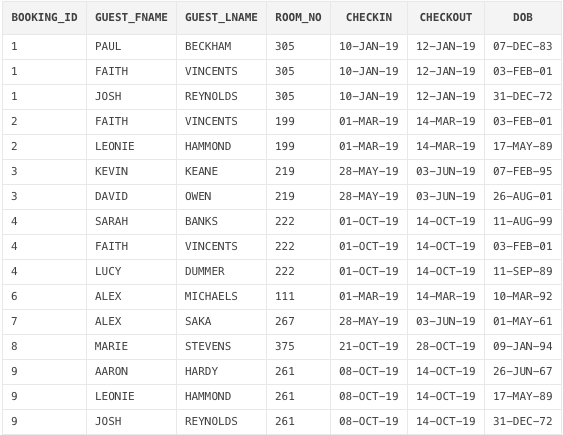


Figure. “Adult Guests’’ Relational Database View

Two count views were created to track the number of adults or children attached to a booking

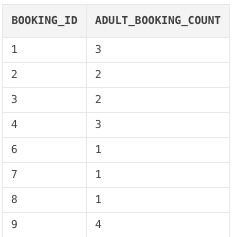


Figure. “Pricing’’ Relational Database View



Figure. “Child Count” Relational Database View

A combination count was then created to show both adults and children per booking. This View was then used to develop the formula for the final Pricing view creation.

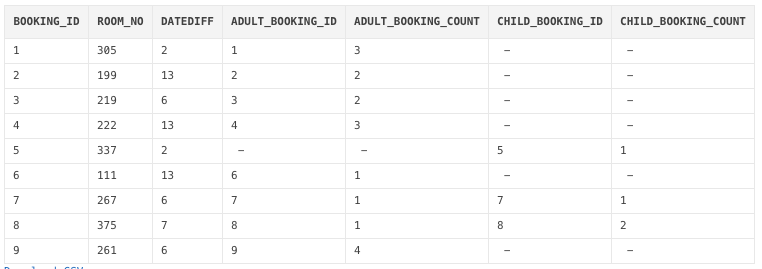


Figure. “Combo Count’ Relational Database View

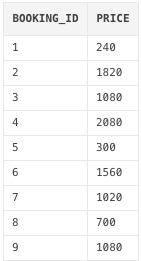


Figure. “Pricing’’ Relational Database View

**Database Queries**

**SQL code to run Queries can be found in the attached submission file “Queries.doc”**

To understand reliability, complex queries must be used to test the consistency of a system.

1. Display all guests and the room they stayed in, ordered by check in date

SELECT GUEST.GUEST\_FNAME, GUEST.GUEST\_LNAME, ROOM\_NO, BOOKINGS.CHECKIN, BOOKINGS.CHECKOUT

FROM BOOKINGDETAIL

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

INNER JOIN BOOKINGS ON BOOKINGS.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

order by BOOKINGS.CHECKIN;

2. List all the Guests older than 18

SELECT GUEST\_FNAME, GUEST.GUEST\_LNAME, DOB

FROM GUEST

where DOB < '13-DEC-2001';

3. Increase room rates by 10%

UPDATE ROOMTYPE

SET ROOM\_RATE = ROOM\_RATE \* 1.1;

4. List the Guests by youngest to oldest

SELECT GUEST\_FNAME, GUEST\_LNAME, DOB

FROM GUEST

ORDER BY DOB DESC;

5. Display how many guests per booking

SELECT BOOKING\_ID, COUNT(BOOKING\_ID) as NumberofGuest

FROM BOOKINGDETAIL

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

Group by BOOKING\_ID

order by BOOKING\_ID;

6. List all hotel rooms that have not had a Booking

SELECT DISTINCT ROOM.ROOM\_NO

FROM ROOM

FULL JOIN BOOKINGDETAIL ON ROOM.ROOM\_NO = BOOKINGDETAIL.ROOM\_NO

WHERE NVL(BOOKINGDETAIL.BOOKING\_ID, 0) = 0;

7. Display all the Guests checked in by Miles

SELECT BOOKINGDETAIL.GUEST\_ID, GUEST.GUEST\_FNAME, GUEST.GUEST\_LNAME, BOOKINGS.BOOKING\_ID, BOOKINGS.CHECKIN, BOOKINGS.CHECKOUT

FROM BOOKINGDETAIL

INNER JOIN BOOKINGS ON BOOKINGS.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

WHERE EMP\_NO IN (SELECT EMP\_NO FROM BOOKINGS WHERE EMP\_NO = 2);

8. Which guests stayed longer than the average stay time  (Display Guest ID & days stayed)

SELECT VIEW\_DAYS\_STAYED.BOOKING\_ID, BOOKINGDETAIL.GUEST\_ID, VIEW\_DAYS\_STAYED.DATEDIFF

FROM VIEW\_DAYS\_STAYED

INNER JOIN BOOKINGDETAIL ON VIEW\_DAYS\_STAYED.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

where DATEDIFF > (select AVG(DATEDIFF) from VIEW\_DAYS\_STAYED)

ORDER BY VIEW\_DAYS\_STAYED.BOOKING\_ID;

9. Which Guests spent the most money (Display Guest ID & Price)

Select BOOKINGDETAIL.BOOKING\_ID, BOOKINGDETAIL.GUEST\_ID, PRICING.PRICE

from PRICING

INNER JOIN BOOKINGDETAIL ON PRICING.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

where PRICE = (select max(PRICE) from PRICING);

10. List all guests who stayed between 1-JAN-2019 dates and 31-MAR-2019. Display

Guest name, Room stayed and check in/out date.

SELECT GUEST.GUEST\_FNAME, GUEST.GUEST\_LNAME, ROOM\_NO, BOOKINGS.CHECKIN, BOOKINGS.CHECKOUT

FROM BOOKINGDETAIL

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

INNER JOIN BOOKINGS ON BOOKINGS.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

WHERE CHECKIN >= '01-JAN-19'AND CHECKOUT<= '31-MAR-19'

ORDER BY CHECKIN;

11. List which Months had bookings and the amount of bookings per month -

SELECT DISTINCT to\_char(BOOKINGS.CHECKIN, 'MM') AS MONTHS\_BOOKED, COUNT(BOOKINGDETAIL.BOOKING\_ID) AS AMOUNT\_OF\_BOOKINGS

FROM BOOKINGDETAIL

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

INNER JOIN BOOKINGS ON BOOKINGS.BOOKING\_ID = BOOKINGDETAIL.BOOKING\_ID

GROUP BY to\_char(BOOKINGS.CHECKIN, 'MM')

ORDER BY to\_char(BOOKINGS.CHECKIN, 'MM');

12. List all guests that stayed in a Deluxe suite  -

SELECT BOOKINGDETAIL.GUEST\_ID, GUEST.GUEST\_FNAME, GUEST.GUEST\_LNAME, BOOKING\_ID, ROOM.ROOM\_NO, ROOM.ROOM\_TYPE

FROM BOOKINGDETAIL

INNER JOIN GUEST ON GUEST.GUEST\_ID = BOOKINGDETAIL.GUEST\_ID

INNER JOIN ROOM ON ROOM.ROOM\_NO = BOOKINGDETAIL.ROOM\_NO

WHERE ROOM\_TYPE='DELUXE';