ER Diagram for case study-Hotel Group

```
WNB_HOTEL_GROUP <- dbConnect(RSQLite::SQLite(), "WNB_HOTEL_GROUP.sqlite")
```

WNB HOTEL GROUP

Given WNB HOTEL GROUP request on providing a system for optimizing its revenue. The Entity-relationship diagram illustrates how every entity to which we want to compile data such as hotels, guests, rooms, types of rooms, services, invoices, items_invoices, reservations, and booking channels are related to each other within the hotel chain. More details are as follows:

- *hotels*
  - Information about hotels of the chain
- *services*
  - Each hotel provides additional services
- *service_by_hotel*
  - Services provided by each hotel
- *rooms*
  - General information about the rooms provided for each hotel including whether it's occupied or not(is_occupied where, 1=yes 0=no)
- *room_class*
  - Specific characteristics for each room #beds as size and smoking area (room_smoking where, 1=yes 0=no)
- *reservations*
  - Details of reservations(including credit card details, preferences(such as, smoking / non-smoking, number of beds, high or low floor), and reservation status where, 1=pending and 0=cancelled)
- *bookings*
  - Reservations already confirmed
- *booking_channels*
  - Information about booking channels (commissions, fee, payment due date)
- *invoices*
  - Information about services used and paid invoices (invoice_paid where, 1=yes 0=no)
- *payment_details*

- Information about services printed on each invoice, amount_items(cost+tax-payment) which should be =0 once paid when checking out.

- **Context-ER DIAGRAM EXPLANATION**

Each hotel owns a variety of rooms (double/single) 1:N relationship (A hotel can own many rooms, but a room can not be owned by many hotels). It also offers the possibility to use different services (rental of phone, use of facilities) to its customers (M:N pointing out that many hotels offer many services, and a service can be provided by many hotels). As a result, we would like to maintain records on which services are available at which hotel (creating a relationship-table called service_by_hotels) after having assumed the M: N cardinality between hotels and services. Following that, a weak entity connected to rooms called "room-class", a 1:M cardinality, to which we will store information regarding the size (number of beds 1 or 2), and whether it's smoking(indicated as numeric, 1) or no-smoking area(indicated as numeric, 0) to eliminate redundancy from rooms table which will have room-class_ID as a foreign key. Financially speaking, it is also cost-effective to know whether the room is occupied or not, so that records of utilization can be tracked (this is indicated as an additional column in the rooms entity). It enables the business to make management decisions for future plans, and keep on track how popular the service is that is being offered.

For every guest who is visiting and using additional services, all charges (including accommodation, additional services, use of facilities) are charged to the room, then reflected in the check-out invoice. Payment_details acts as the relationship linking Invoices and Services. At the same time, invoices holds each unique booking ID. Therefore, the total invoice will have a full description of the services used by the customer, and methods of payment when checking out. Payment_details relationship table was generated after having established a M:N cardinatily of services and invoices (a service can be reflected on multiple invoices and an invoice can display different services). At the same time, invoices entity contains the balance of charges, taxes and the amount paid (charges + taxes - amount paid) which is reflected in a column called, item amount specifying the details of the invoice.

NOTE: all amount fees are paid when checking out fully. Since every hotel can offer different services, the amount of fee per service will depend on each hotel fee whereas for rooms all costs are equal across hotels.

Regarding guests entity, it is represented as another principal connection in which we are keeping all personal details of visitors. This data is tracked historically by each hotel. It provides crucial attributes such as full name, residence address, and email and phone number records that are non-compulsory to fill out during the registration process. Following that, the cardinality 1:M reflects that one guest can make multiple reservations, but a reservation can only be processed by one guest ID. Likewise, reservations status can be either pending or cancelled. Additionally,it's mandatory to provide credit card details to secure the reservation (NOT NULL). Finally, once the hotel checks all requirements, chooses floor, smoking area according to the client's preferences, then it's reflected on bookings.

Moving on through the ER diagram, those reservations pass to become bookings (1:1 it's just a record transaction). Every booking has a unique ID (booking ID). Subsequently, an invoice is generated for each booking id, regardless of whether it belongs to the same customer. As a result of the M:N relationship within guests and rooms a bookings table tracks historical information of all guests who was accommodated in the the whole hotel's rooms.

The hotel has to record the booking channels to which every customer processed their reservations (M:1 many reservations are secured through one booking channel, and one booking channel can have many reservations), each reservation will have booking_channel_ID as a foreign key. Then,we have the percentage of commission and the channel fee for each channel (Booking.com, Hotels.com, Tripadvisor.com, etc.). The total amount that will be transferred to each third party is calculated based on the total charges excluding taxes, multiplied by the percentage of commission of each booking channel.

Please refer to Appendix Part A1

- **SQL QUERIES**

1.The total spent for the customer for a particular stay (checkout invoice).

As we have made the following assumption: we have item_amount which contains the balance (charges+taxes-amountpaid) , then item_amount column from invoice items entity contains these three values on different rows. We applied sum function for item_amount and multiply by -1 to generate a positive value.

```
SELECT p1.guest_id , p1.departure_date,
SUM(p3.item_amount)*-1 AS "Total_Spent"
FROM invoices AS p2
INNER JOIN payment_details AS p3
ON p2.invoice_id = p3.invoice_id
INNER JOIN bookings AS p1
ON p2.booking_id = p1.booking_id
WHERE p3.item_amount<0
GROUP BY guest_Id
```

0 records

guest_id	departure_date	Total_Spent

2.The most valuable customers in (a) the last two months, (b) past year and (c) from the beginning of the records.

Here, we have the same explanation as above for total spent. Assuming that Today's date is 2021-11-19 00:00:00 , we can use BETWEEN to calculate the total spent in the last two months. Also,we have to consider bookings instead of reservations since that is the real revenue (all booking are confirmed, not longer reservations).Finally, everything has been ordered by total spent descendingly.

```sql
-- most valuable customers in (a) the last two months TOP 20

----Today's date 2021-11-19 00:00:00

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND p3.arrival_date BETWEEN DATETIME("2021-09-19
00:00:00") AND DATETIME("2021-11-19 00:00:00") AND item_amount<0
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20 ;
```

0 records

guest_id TOTAL_SPENT

```sql
-- most valuable customers in (b) the past year  TOP 20

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND p3.arrival_date BETWEEN DATETIME("2020-11-19
00:00:00") AND DATETIME("2021-11-19 00:00:00") AND item_amount <0
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20
;
```

0 records

guest_id TOTAL_SPENT

```sql
-- most valuable customers (c) beginning of the records  TOP 20

SELECT p3.guest_id,
 (sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices  AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND item_amount <0
```

```
GROUP BY  p3.guest_id
ORDER BY TOTAL_SPENT desc
LIMIT 20
;
```

0 records

guest_id	TOTAL_SPENT

3.Which are the top countries where our customers come from ?

This query can be run by selecting columns from the guests entity: guest_country and guest_id, counting each country, and finally ordering descendingly.

```
---TOP 10 COUNTRIES

SELECT p1.guest_country AS "Country" ,COUNT(p1.guest_id) AS "total_guests"
FROM guests AS p1
GROUP BY p1.guest_country
ORDER BY total_guests desc
LIMIT 10;
```

0 records

Country	total_guests

4. How much did the hotel pay in referral fees for each of the platforms that we have contracted with?

On our database referral fees are in the column called booking_channel_commission % from the booking channels entity.

We are assuming that we pay the % of commissions based on the item amount without taxes.Therefore, we have to multiply the % of commission (booking_channel_commission) by the total item amount. Finally, as the balance item_amount is <0, we have to multiply the sum by -1 so as to generate a positive value.

```
SELECT p1.booking_channel_id,
p1.booking_channel_name,
(SUM (p1.booking_channel_commission) *(p4.item_amount))*-1 AS Referral_Fees
FROM booking_channels AS p1
INNER JOIN reservations AS p2
ON p2.booking_channel_id  = p1.booking_channel_id
INNER JOIN invoices AS p3
ON p3.booking_id = p2.booking_id
INNER JOIN payment_details AS p4
ON p3.invoice_id = p4.invoice_id
```

```
WHERE p3.invoice_paid =1 AND p4.is_tax=0 AND p4.item_amount<0
GROUP BY p1.booking_channel_id
ORDER BY Referral_Fees desc
;
```

0 records

booking_channel_id	booking_channel_name	Referral_Fees

5. What is the utilization rate for each hotel (that is the average billable days of a hotel specified as the average utilization of room bookings for the last 12 months)

The utilization rate will be # of rooms which are occupied (room_occupied='1') divided by the total of rooms either occupied or not (room_occupied)

```
----Today's date 2021-11-19 00:00:00

SELECT p2.hotel_id,(count(p2.room_occupied='1'))/(count(p2.room_occupied)) AS
"RATE_UTILIZATION"
FROM rooms AS p2
INNER JOIN bookings AS p1
ON p2.room_id= p1.room_id
WHERE p1.arrival_date BETWEEN DATETIME("2020-11-19 00:00:00") AND
DATETIME("2021-11-19 00:00:00")
GROUP BY p2.hotel_id
ORDER BY RATE_UTILIZATION DESC
;
```

0 records

hotel_id	RATE_UTILIZATION

6.Calculate the Customer Value in terms of total spent for each customer before the current booking.

To calculate this particular task we calculated the total item amount (the balance described previously) and filter by the arrival dates before the current date which is "now". It takes all previous visits from each customer without the current booking.

```
SELECT p3.guest_id,
(sum(p1.item_amount)*-1) AS "TOTAL_SPENT"
FROM payment_details AS p1
INNER JOIN invoices AS p2
ON p1.invoice_id = p2.invoice_id
INNER JOIN bookings AS p3
ON p2.booking_id= p3.booking_id
WHERE p2.invoice_paid = 1 AND  p3.arrival_date < DATETIME("now") AND
```

```
p1.item_amount <0
GROUP BY p3.guest_id
ORDER BY TOTAL_SPENT DESC
;
```

0 records

guest_id	TOTAL_SPENT