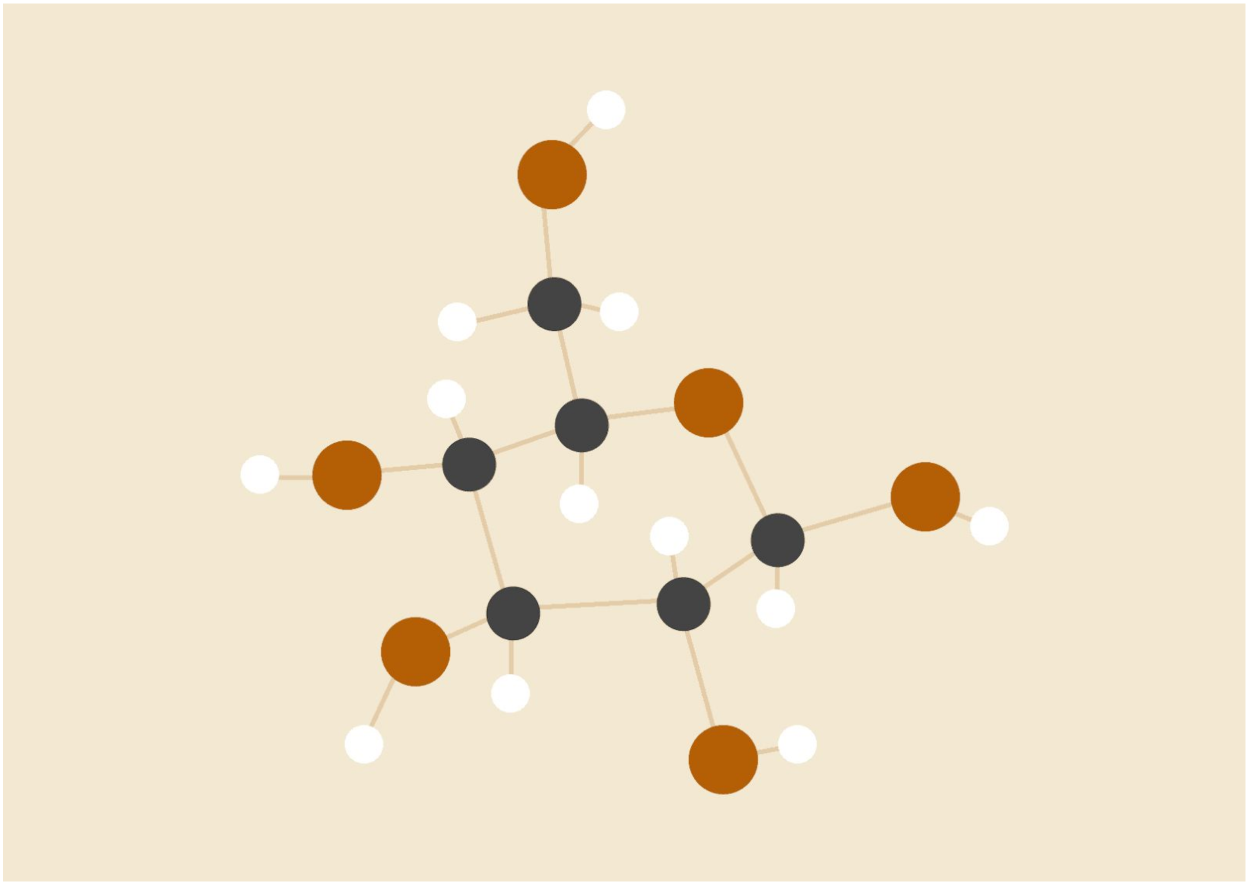
Hotels Database

Database for hotel reservation system



by.**Mohammed Salahadin & Abdulrahman Tawfiq**

F180388,F180292

Computer Science Department

INTRODUCTION

Hotels reservation full system in Kurdistan, this database will store the data about hotels.

This database stores multiple users accounts and each user have the ability to add multiple hotels and each hotel will have multiple rooms with options and users have the ability to:

- Add a hotel and specify the hotel information:
    - Hotel information: eg. Rating, address, contact info, etc.
    - Hotel amenities.
    - Hotel facilities.
    - Hotel Policies.
- Add Rooms to the registered hotel and specify the following
    - Room type:eg.Double
    - RoomName: eg. Deluxe Double
    - Room Status:Available,Reserved, Under construction
- Add Rates:
    - Specify room specific type of rooms.
    - Specify rooms prices for specific dates:
    - Specify available rate types and their prices.
    - Specify where the room is available or not on a specific date.
- Add Rates Types: each hotel has different rate types, therefore we give an option for the hotels to provide their own rate type to use it when determining their Prices.
- Add Supported payment methods.

It have the ability to store multiple guests accounts and each guest will have the ability to do the following:

- Search and View the list of hotels and the services that they provide
- View the available rooms and prices for each hotel.
- Reserve a room or multiple rooms.
- See the hotels pictures and the amenities that they provide.

- Book and pay either online or through one of the available payments systems, or pay at hotel according to the hotel policy
- View hotels locations according to their coordinates on throw google ma

It have the ability to store multiple companies accounts each company will have same options as the guests permissions plus:

- 5% discount on each reservation.
- Free cancelation service

The database contains a table for string multiple admins each admin will have the ability to do the following:

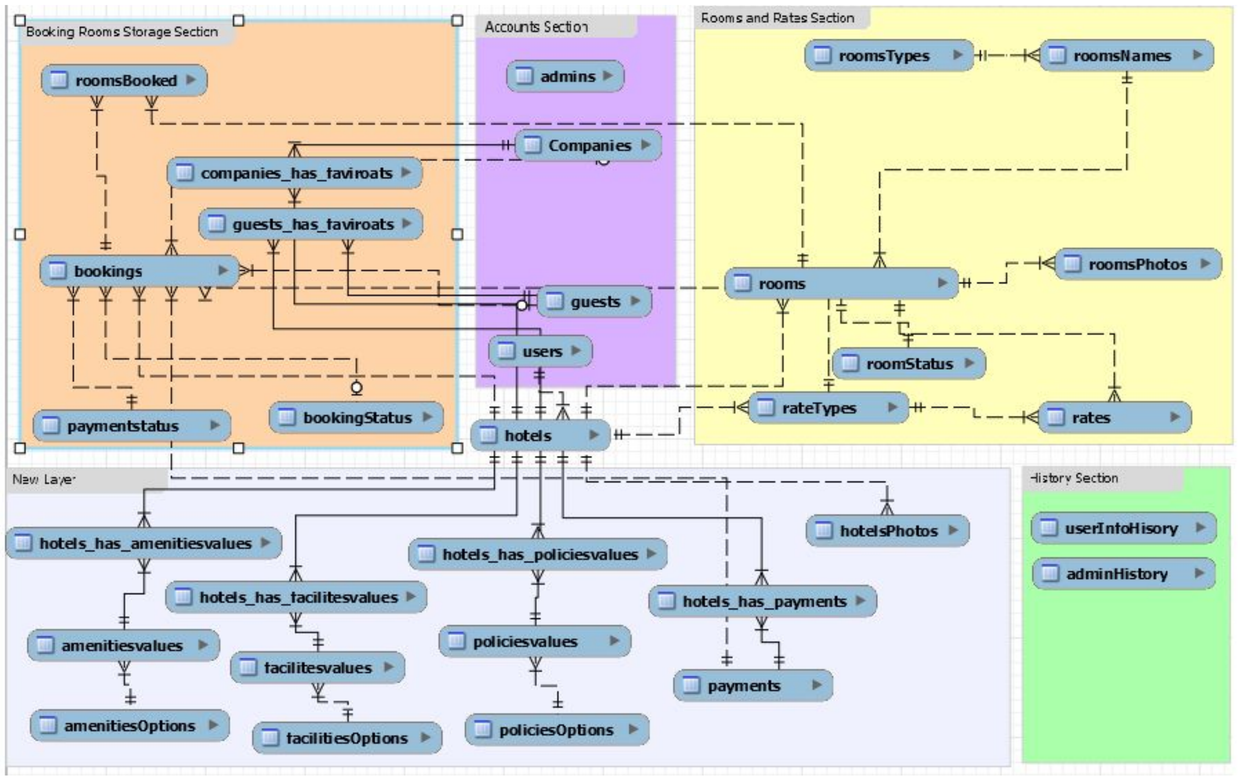
- View all the registered Users, Guests, Companies accounts in the database.
- Add, Remove, Active, deactivate users, guests and companies accounts.
- Add, Remove, modify
    - Facilities options
    - amenities Options
    - Available Payment Options
    - Rooms types and rooms names.

This Database will present the backend of the website and the main structure of the website, it will be used for storing, Processing and fetching out all the required information from and into the website.

DIAGRAM:

The diagram consists of several sections. Each section holds several tables to handle specific tasks.

1. **Guest and Users Accounts Section:** this section stores the users and guests registration information, users are representing the hotels owners and they will be able to have multiple hotels in the hotels table.
2. **Booking rooms Storage Section:** this section is used for storing booked rooms for the booking status and specify the booked rooms types.
3. **Rooms and Rates Section:** Here we store each hotel room's types and specify their kind, storing the state of the rooms as well the prices and rates.
4. **Hotel Options Section:** For storing each hotel amenities, facilities and policies.
5. **Tables History Section:** For storing the changes that occurred on the Accounts information table.



HOW IT WORKS

Users are able to register and have an account, the user's information will be stored inside users table each user will have the ability to insert and have more than one hotel therefore the relationship between users and hotels table is one to many, each hotel will be stored inside the hotels table and will have it's primary key and each hotel will be linked to several tables and they are:

1. **Rooms:** to store multiple rooms types of the hotel, therefore the relationship is one to many
2. **Rate Types:** to store the available rates for the hotel eg.room with breakfast, room with swimming pool etc, in order to specify the price of each rate when the customer reserves.
3. **Bookings:** this table will store all the bookings that have been made by guests or companies to specify which hotel has been reserved.
4. **hotels_has_aminites:** to store the hotel's amenities information to show it to the guest when he reserves the hotel, in the amenities table there will be some stored options available for each hotel to choose from. Eg. provide extra beds, Air conditioning, Bath, Electric kettle, Balcony, Etc.
5. **hotels_has _facilities:** to store the hotels facilities to show it to the guest when he reserves the hotel, in the facilities table there will be some stored services available and they are optional for each hotel to choose from, Eg.Internet, Parking, Breakfast, Restaurant,Swimmingpool, Sauna, Waterpark, etc.
6. **Hotels_has_policies:** to Store the hotel policies Eg.Cancellations policies, Check-in time, pets, etc.
7. **Hotels_has_supportPayment:** to store the supported payment systems that guest can pay while booking a room.

They then add rooms with their types and kinds, after that they will be able to specify which rooms are available on a specific date and specify how much it will cost in several rates in the rate table.

Guests can view the available rooms from the rate table according to their request of guest numbers the output will be a list of the rooms

TABLES

Accounts Section Tables

- Admin:to store admins accounts they are able to control the whole database.
- Users : to store hotels owners accounts and their information.
- Guests: to store guests accounts and their information.
- Companies: to store registered companies accounts for reserving for groups.

Hotels table: to store the hotels which are added by the users and full information.

Hotel Options Section Tables

- AmenitiesOptions: for storing the standard amenities types options that hotels will use.
- amenitiesList: Stores a list of amenities for each amenity type in amenities options.
- hotels_has_amenitiesList: for soring the amenities for each hotel.
- FacilitiesOptions: for storing the standard facilities options which hotels are going to use.
- facilitesList: Stores a list of facilities for each facility type in facilitiesOptions Table.
- hotels_has_facilitesList: for storing the facilities for each hotel.
- PoliciesOptions: to store the standard Policies to be assigned to each hotel.
- policiesList: to store the values of each policy type.
- hotels_has_policiesList:for string each hotel policies.
- SupportPayment:for storing the supported payments in Iraq that hotels are going to use for getting paid from guests.
- hotels_has_payments: for storing the supported payment methods for each hotel.
- hotelsPhotos:for storing each hotel photo's paths.

Booking Rooms & Storage Section Tables

- Bookings: to store all bookings which have been done by guests and companies.
- RoomsBooked: declare which rooms have been reserved along with hotels (For

storing both booked hotel ID and the room ID.

- BookingStatus: to store the statues types for the booking.
- Paymentstatus: for storing the status of the payment of the customer if it’s paid, not paid, fail to read credit card etc.
- Companies_has_faviroats: to store the favorite hotels added by companies.
- guests_has_faviroats : to store the favorite hotels added by guests.

Note: when new bookings have been added the payment status by default is ‘1’ which stands for not paid.

Rooms and Rates Section tables:

- Rooms: for holding and storing all the rooms added by each hotel.
- RoomsPhotos:for storing rooms photos paths.
- RoomsTypes: for storing the standard rooms types.
- RoomsNames: for storing the rooms kinds for each room type.
- RoomsStatus: to store the statues types determined by admin.
- Rates: for storing available rates for each room.
- RateTypes: for storing the standard rates determined by the users.

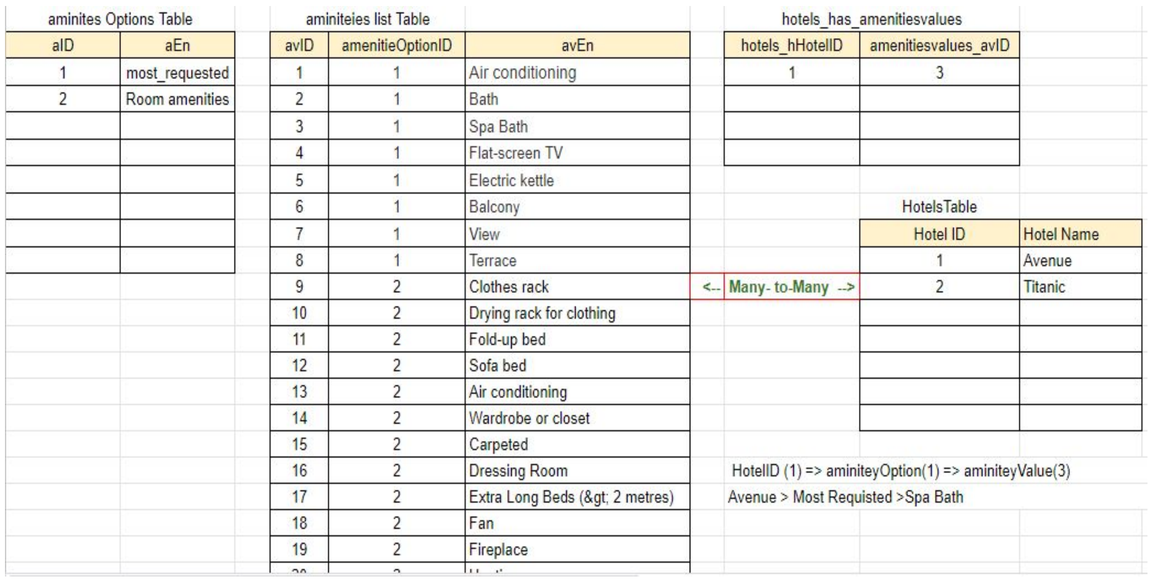
History Section tables:

- UserInfoHistory: for storing the history of changes happened on the user account.
- AdminHistory: for storing the history of the operations that have been made by admins.

RELATIONSHIPS

Amenities Relationship with hotel:

Declaration:



aminites Options Table

aID	aEn
1	most_requested
2	Room amenities

aminiteies list Table

avID	amenitieOptionID	avEn
1	1	Air conditioning
2	1	Bath
3	1	Spa Bath
4	1	Flat-screen TV
5	1	Electric kettle
6	1	Balcony
7	1	View
8	1	Terrace
9	2	Clothes rack
10	2	Drying rack for clothing
11	2	Fold-up bed
12	2	Sofa bed
13	2	Air conditioning
14	2	Wardrobe or closet
15	2	Carpeted
16	2	Dressing Room
17	2	Extra Long Beds (&gt; 2 metres)
18	2	Fan
19	2	Fireplace

hotels_has_amenitiesvalues

hotels_hHotelID	amenitiesvalues_avID
1	3

<-- Many- to-Many -->

HotelsTable

Hotel ID	Hotel Name
1	Avenue
2	Titanic

HotelID (1) => aminiteyOption(1) => aminiteyValue(3)

Avenue > Most Requisted >Spa Bath

Note: The other relationships are declared in the diagram.

QUERIES

The queries which are going to be executed by Guests, Companies, Users and Admins.

Queries which will be executed by **admins:**

1. Adding New Admin:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`admins` (`aFullName`, `aUserName`,
   `aPassword`, `aEmail`) VALUES ('Abdulrahman.T Jalal', 'Abdulrahman', '12345',
   'atj@email.com');
   ```

2. Adding Rooms Types:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`roomstypes` (`rtRoomTypeID`, `rtRoomTypeEn`,
   `rRoomTypeAr`, `rRoomTypeKu`, `trActive`) VALUES ('1', 'Single', 'مفردة', 'يه كي', '1'
   ');
   ```

3. Adding room names (kinds) for single room type:

   ```
   INSERT INTO `m&a_hotels`.`roomsnames`
   (`rnRoomNameID`,`rnRoomNameEn`, `rnRoomNameAr`, `rnRoomTypeID`)
   VALUES ('1','Budget Single Room', 'غرفة اقتصادية مفردة', '1');
   ```

   ```
   INSERT INTO `m&a_hotels`.`roomsnames`
   (`rnRoomNameID`,`rnRoomNameEn`, `rnRoomNameAr`, `rnRoomTypeID`)
   VALUES ('2','Deluxe Single Room', 'غرفة ديلوكس مفردة', '1');
   ```

4. Adding amenities option:

   Using SQl:

   ```
   INSERT INTO `m&a_hotels`.`amenitiesoptions` (`aID`, `aEn`, `aAr`, `aKu`)
   VALUES ('1', 'most_requested', 'الأكثر طلبا', '');
   ```

5. Adding amenities for each amenity type into ameniteList table:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`amenitiesvalues` (`avID`, `amenitieOptionID`, `avEn`) VALUES ('1', '1', 'Air conditioning');
   ```

   ```
   INSERT INTO `m&a_hotels`.`amenitiesvalues` (`avID`, `amenitieOptionID`, `avEn`) VALUES ('2', '1', 'Bath');
   ```

   ```
   INSERT INTO `m&a_hotels`.`amenitiesvalues` (`avID`, `amenitieOptionID`, `avEn`) VALUES ('3', '1', 'Spa Bath');
   ```

6. Adding facilities options types:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`facilitiesoptions` (`fFaciliteID`, `fFaciliteNameEn`, `fFaciliteNameAr`) VALUES ('1', 'Internet', 'خدمة الأنترنت');
   ```

7. Adding facilities choices for each Option into facilitiesList Table:

   ```
   INSERT INTO `m&a_hotels`.`facilitesvalues` (`fID`, `fFacilitesOptionsID`, `fOptionsEn`, `fOptionsAr`) VALUES ('1', '1', 'Not Available', 'غير متاح');
   ```

   ```
   INSERT INTO `m&a_hotels`.`facilitesvalues` (`fID`, `fFacilitesOptionsID`, `fOptionsEn`, `fOptionsAr`) VALUES ('2', '1', 'Yes, Free', 'نعم ، مجاناً');
   ```

   ```
   INSERT INTO `m&a_hotels`.`facilitesvalues` (`fID`, `fFacilitesOptionsID`, `fOptionsEn`, `fOptionsAr`) VALUES ('3', '1', 'Yes, Paid', 'نعم ، مدفوع');
   ```

8. Adding policies options:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`policiesoptions` (`pPoliceID`, `pPoliceNameEn`, `pPoliceNameAr`) VALUES ('1', 'days in advance can guests cancel free of charge', 'عدد الأيام التي يمكن للضيف الغاء الحجز بدون رسوم');
   ```

9. Adding policies values for each policy option:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`policiesvalues` (`pID`, `pPoliciesID`, `pOptionEn`, `pOptionAr`) VALUES ('1', '1', 'Day of arrival (18:00)', '18:00) يوم الوصول)');
   ```

   ```
   INSERT INTO `m&a_hotels`.`policiesvalues` (`pID`, `pPoliciesID`, `pOptionEn`, `pOptionAr`) VALUES ('2', '1', '1 day', '1 يوم');
   ```

   ```
   INSERT INTO `m&a_hotels`.`policiesvalues` (`pID`, `pPoliciesID`, `pOptionEn`, `pOptionAr`) VALUES ('3', '1', '2 day', 'يومان');
   ```

10. Adding Payment types:

    Using SQL:

    ```
    INSERT INTO `m&a_hotels`.`payments` (`pID`, `pName`, `pType`) VALUES ('1', 'Fast pay', 'Online');
    ```

    ```
    INSERT INTO `m&a_hotels`.`payments` (`pID`, `pName`, `pType`) VALUES ('2', 'Asia Hawala', 'Online');
    ```

11. Adding booking status list:

    Using SQL:

    ```
    INSERT INTO `m&a_hotels`.`bookingstatus` (`bsBookingStatusID`, `bsStatus`, `bsDescription`, `bsActive`) VALUES ('1', 'Pending', 'whating for the hotel to confirm', '1');
    ```

    ```
    INSERT INTO `m&a_hotels`.`bookingstatus` (`bsBookingStatusID`, `bsStatus`, `bsDescription`, `bsActive`) VALUES ('2', 'Canceled-hotel', 'Canceld by hotel', '1');
    ```

12. Adding room status options:

    Using SQL:

    ```
    INSERT INTO `m&a_hotels`.`roomstatus` (`rsRoomStatusID`, `rsRoomStatus`, `rsDescription`, `rsActive`) VALUES ('1', 'Reserved', 'all rooms of this type are reserved', '1');
    ```

    ```
    INSERT INTO `m&a_hotels`.`roomstatus` (`rsRoomStatusID`, `rsRoomStatus`, `rsDescription`, `rsActive`) VALUES ('2', 'Available', 'There are Available rooms for reserving', '1');
    ```

13. Get a Report of all guests names, their booking ID, types of rooms, Dates of bookings, Number of nights they stay and number of passengers:

    ```
    SELECT g.gFullName as 'Guest full Name', b.bBookingID as 'Booking ID', rt.rtRoomTypeEn as 'Room Type', rn.rnRoomNameEn as 'Room Kind'
    , b.bDateFrom as 'From Date', b.bDateTo as 'To Date', b.bNumOfNights as 'Number of nights to stay',
     b.bNumOfGuests as 'Number of adults', b.bNumOfChildren as 'Number of children' FROM bookings b
    JOIN roomsbooked rb ON b.bBookingID = rb.rbBookingID
    JOIN rooms r ON rb.rbRoomID = r.rRoomID
    JOIN roomsnames rn ON r.rRoomNameID = rn.rnRoomNameID
    JOIN roomstypes rt ON rt.rtRoomTypeID = rn.rnRoomTypeID
    JOIN guests g ON g.gGuestID = b.bGuestID
    WHERE b.bHotelID = 1
    AND b.bDateFrom >= '2020-05-19'
     AND b.bDateFrom <= '2020-05-31';
    ```

Queries which will be executed by **Users**

Users Queries:

1. For adding New user

    Using SQL:

    ```
    INSERT INTO `m&a_hotels`.`users` (`uUserName`, `uUserEmail`, `uUserPassword`, `uFullName`, `uUserPhoneNumber`, `uStatus`) VALUES ( 'Salim_Kareem', 'salim@salim.com', '11234', 'Salim Kareem Hama', '07701101010', '1');
    ```

2. For adding new Hotel *user should be added:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`hotels` (`hUserID`, `hHotelName`, `hHotelRating`,
   `hAddress2`, `hCity`, `hState`, `hZipCode`, `hMainPhoneNumber`, `hNormalNumber`,
   `hCompanyMailAddress`, `hWebsiteAddress`, `hLogoPath`, `hActive`, `hAddress`,
   `hMapsLocation`) VALUES ('1', 'Avenue', '4', 'Sarchnar', 'suly', 'suly', '40005',
   '07701111111', '05311111', 'avenue@hotel.com', 'avenue.com',
   '../logos/hotelID/logo.png', '1', 'sarchnar',
   '@35.614260699999996,45.3491321,16z');
   ```

3. To add room * will use different data from different tables, Hotel should be created in order to add a room to it.

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`rooms` (`rRoomID`, `rHotelID`, `rRoomStatusID`,
   `rnRoomNameID`, `rFloor`, `rRoomsNumbers`, `rDescription`,
   `rCustomName`, `rSmokingPolicy`, `rRoomSize`, `rLowestPrice`,
   `rOfferLower`, `rDisscount`) VALUES ('1', '1', '1', '2', '1', '56', 'description', 'custom
   Name', '1', '30', '78', '1', '10');
   ```

4. Adding rates types for hotel:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`ratetypes` (`rtRateTypeID`, `rtRateType`,
   `rtDescription`, `rtSortOrder`, `rtActive`, `hotels_hHotelID`) VALUES ('1',
   'Breakfast', 'Price including Breakfast', '1', '1', '1')
   ```

5. Add amenities to the created hotel:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`hotels_has_amenitiesvalues` (`hotels_hHotelID`,
   `amenitiesvalues_avID`) VALUES ('1', '2');
   ```

6. Add Facilities to the hotel:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`hotels_has_facilitesvalues` (`hotels_hHotelID`,
   `facilitesvalues_fID`) VALUES ('1', '2');
   ```

7. Add Policies to the hotel:

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`hotels_has_policiesvalues` (`hotels_hHotelID`,
   `policiesvalues_pID`) VALUES ('1', '2');
   ```

8. Adding Photos paths :

   Using SQL:

   ```
   INSERT INTO `m&a_hotels`.`hotelsphotos` (`hpID`, `hotels_hHotelID`,
   `hpPath`) VALUES ('1', '1', '../photos/hotels/hotel_ID/photos/ph1.jpg');
   ```

   ```
   INSERT INTO `m&a_hotels`.`hotelsphotos` (`hpID`, `hotels_hHotelID`,
   `hpPath`) VALUES ('2', '1', '../photos/hotels/hotel_ID/photos/ph2.jpg');
   ```

   ```
   INSERT INTO `m&a_hotels`.`hotelsphotos` (`hpID`, `hotels_hHotelID`,
   `hpPath`) VALUES ('3', '1', '../photos/hotels/hotel_ID/photos/ph2.jpg');
   ```

9. To view the list of the hotels this user have:

   Using SQL:

   ```
   SELECT * FROM `m&a_hotels`.hotels WHERE hHotelID = 1;
   ```

   Algebra:

   σ {hHotelID = 1}(hotels)

10. To view a list of the rooms types for the hotel that have been created :

    Using SQL:

    ```
    SELECT rt.rtRoomTypeEn, rn.rnRoomNameEn FROM `m&a_hotels`.rooms `r`
    JOIN roomsnames `rn` ON r.rRoomNameID = rn.rnRoomNameID
    JOIN roomstypes `rt` ON rn.rnRoomTypeID = rt.rtRoomTypeID
    WHERE r.rHotelID = "1";
    ```

    Using Algebra:

    π rt.rtRoomTypeEn, rn.rnRoomNameEn σ r.rHotelID = 1 ρ r rooms $\bowtie$ r.rRoomNameID = rn.rnRoomNameID ρ rn roomsnames $\bowtie$ rn.rnRoomNameID = rt.rtRoomTypeID ρ rt roomstypes

11. Adding room to sell (including Price and date and rate).

    Using SQL:

    ```
    INSERT INTO `m&a_hotels`.`rates` (`rRateID`, `rRoomsID`, `rRateTypeID`,
    `rRate`, `rDate`) VALUES ('1', '1', '1', '46', '2020-05-27');
    INSERT INTO `m&a_hotels`.`rates` (`rRateID`, `rRoomsID`, `rRateTypeID`,
    `rRate`, `rDate`) VALUES ('2', '1', '1', '45', '2020-05-28');
    ```

Queries which will be executed by **Guests & Companies:**

1. Created accounts:

   Insert New guest Account:

   Using SQL:

```
INSERT INTO `m&a_hotels`.`guests` (`gGuestID`, `gFullName`,
`gUserName`, `gPassword`, `gAddress`, `gAddress2`, `gCity`, `gState`,
`gZipCode`, `gCountry`, `gPhoneNumber`, `gEmail`, `gGender`,
`gStatus`) VALUES ('1', 'Mohammed salahadin Hazim',
'MohammedsHazim', '12345', 'Qularaesee', 'sarchnar', 'sulaymaniyah',
'sulaymaniyah', '40004', 'Iraq', '07708138928', 'm@m.com', 'male', '1');
```

   Insert New Company Account:

```
INSERT INTO `m&a_hotels`.`companies` (`cCompanyID`,
`cOwnerFullName`, `cUserName`, `cPassword`, `cAddress`,
`cAddress2`, `cCity`, `cState`, `cZipCode`, `cPhoneNumber`, `cEmail`,
`cStatus`) VALUES ('1', 'Abdulrahman Tawffeq Jalal', 'ATJALAL', '12345',
'Qanat', 'QaziMuhammrd', 'suly', 'suly', '40005', '07701111111',
'a@email.com', '1');
```

2. View all hotels and their rooms in specific city eg.sulaymaniyah:

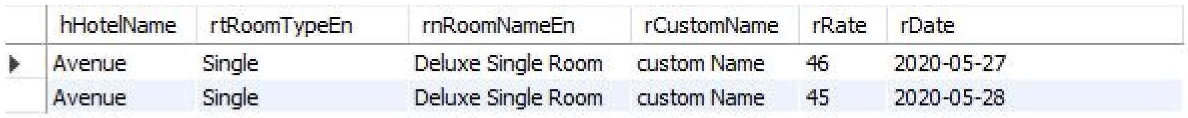
   Using SQL:

```
SELECT h.hHotelName,rt.rtRoomTypeEn,rn.rnRoomNameEn, r.rCustomName,
rs.rRate, rs.rDate FROM `m&a_hotels`.hotels `h`
JOIN rooms `r` ON h.hHotelID = r.rHotelID
JOIN roomsnames `rn` ON r.rRoomNameID = rn.rnRoomNameID
JOIN roomstypes `rt` ON rn.rnRoomTypeID = rt.rtRoomTypeID
JOIN rates `rs` ON r.rRoomID = rs.rRoomsID
WHERE h.hCity LIKE "suly"
```

Using Algebra:

π h.hHotelName,rt.rtRoomTypeEn,rn.rnRoomNameEn, r.rCustomName, rs.rRate, rs.rDate σ h.hCity LIKE ‘suly’ ρ h hotels $\bowtie$ h.hHotelID = r.rHotelID ρ r rooms $\bowtie$ r.rRoomNameID = rn.rnRoomNameID ρ rn roomsnames $\bowtie$ rn.rnRoomTypeID = rt.rtRoomTypeID ρ rt roomstypes $\bowtie$ r.rRoomID = rs.rRoomsID ρ rs rates

Result:



hHotelName	rtRoomTypeEn	rnRoomNameEn	rCustomName	rRate	rDate
Avenue	Single	Deluxe Single Room	custom Name	46	2020-05-27
Avenue	Single	Deluxe Single Room	custom Name	45	2020-05-28

3. Adding a hotel to the favorites list:

   Using SQL:

   INSERT INTO `m&a_hotels`.`guests_has_faviroats` (`guests_gGuestID`, `hotels_hHotelID`) VALUES ('1', '3');

4. Reserving a room:

   Using SQL:

   Using procedure

   -- adding bookings

   -- Using addReservation procedure

   -- (hotelID INT, RoomID INT, guestID INT, rateTypeID INT, dateFrom DATE, dateTo DATE,numOfRooms INT,psNum INT,ChildrenNum INT, bPaymentID INT)

   CALL addReservation ('1','1','3','1','2020-06-9','2020-06-11', '1', '2', '1', '1');

   CALL addReservation ('2','2','4','1','2020-06-9','2020-06-11', '1', '2', '1', '1');

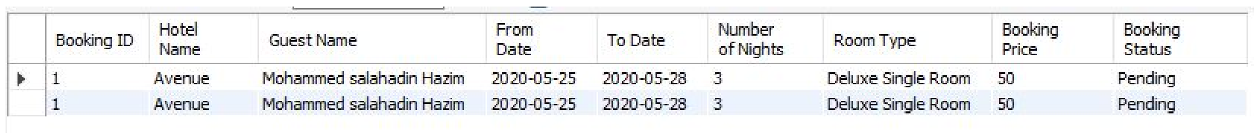
   CALL addReservation ('2','3','5','1','2020-06-9','2020-06-11', '1', '2', '1', '2');

   CALL addReservation ('2','4','4','1','2020-06-9','2020-06-11', '1', '2', '1', '2');

5. View bookings for the current guest (guest ID:1):
   Using SQL:

```
SELECT b.bBookingID as 'Booking ID', h.hHotelName as 'Hotel Name', g.gFullName
as 'Guest Name', b.bDateFrom as 'From Date',
b.bDateTo as 'To Date', b.bNumOfNights as 'Number of Nights',
rn.rnRoomNameEn as 'Room Type', rb.rbRate as 'Booking Price',
bs.bsStatus as 'Booking Status' FROM bookings b
JOIN roomsbooked rb ON b.bBookingID = rb.rbBookingID
JOIN rooms r ON r.rRoomID = rb.rbRoomID
JOIN roomsnames rn ON r.rRoomNameID = rn.rnRoomNameID
JOIN rates rs ON rs.rRoomsID = r.rRoomID
JOIN hotels h ON b.bHotelID = h.hHotelID
JOIN guests g ON b.bGuestID = g.gGuestID
JOIN bookingstatus bs ON bs.bsBookingStatusID = b.bBookingStatusID
WHERE b.bGuestID = 1;
```

   Eg. Output:



Booking ID	Hotel Name	Guest Name	From Date	To Date	Number of Nights	Room Type	Booking Price	Booking Status
1	Avenue	Mohammed salahadin Hazim	2020-05-25	2020-05-28	3	Deluxe Single Room	50	Pending
1	Avenue	Mohammed salahadin Hazim	2020-05-25	2020-05-28	3	Deluxe Single Room	50	Pending

FUNCTIONS

Find numbers of bookings for a specific hotel:

```
-- Get Number of bookings for a specific hotel using it's ID
CREATE FUNCTION get_hotel_bookings(hotelID int)
RETURNS INT deterministic
return (SELECT COUNT(b.bHotelID) FROM bookings b WHERE b.bHotelID =
hotelID);
```

Get number of all bookings for all the hotels specific city:

```
-- Get number of all bookings for all the hotels specific city and specific state:
DROP FUNCTION IF EXISTS get_bookings;
CREATE FUNCTION get_bookings(city VARCHAR(45), bStatus VARCHAR(45))
RETURNS INT deterministic
RETURN (SELECT COUNT(b.bHotelID) FROM bookings b
JOIN hotels h ON b.bHotelID = h.hHotelID
JOIN bookingStatus bs ON b.bBookingStatusID = bs.bsBookingStatusID
WHERE h.hCity LIKE concat('%',city,'%') and bs.bsStatus LIKE
concat('%',bStatus,'%')
);
-- for test: SELECT get_bookings('suly', 'Canceled-hotel');
```

TRIGGERS

1.When inserting new row to the admin,user,guest and companies accounts change the user letters from Uppercase to lowercase

```
-- for Change users to lower
DROP TRIGGER IF EXISTS userToUpper;
CREATE TRIGGER userToUpper
BEFORE INSERT ON users
FOR EACH ROW
SET NEW.uUserName =lower(new.uUserName);


-- for Change Admins to lowercase
DROP TRIGGER IF EXISTS adminToUpper;
CREATE TRIGGER adminToUpper
BEFORE INSERT ON admins
FOR EACH ROW
SET NEW.aUserName =lower(new.aUserName);


-- for Change guests to Lowercase
DROP TRIGGER IF EXISTS guestToUpper;
CREATE TRIGGER guestToUpper
BEFORE INSERT ON guests
FOR EACH ROW
SET NEW.gUserName =lower(new.gUserName);
```

2. Insert the number of nights on each booking according to the reservations dates:

```
DROP TRIGGER IF EXISTS insertNights;
CREATE TRIGGER insertNights
AFTER INSERT ON bookings
FOR EACH ROW
```

```
SET NEW.bNumOfNights = DATEDIFF(NEW.bDateTo, NEW.bDateFrom);
```

3.Recorde the users and admins update info history:

```
-- recorde of the users update info history
DROP TRIGGER IF EXISTS userInfoHistory;
CREATE TRIGGER  userInfoHistory
AFTER UPDATE ON users
FOR EACH ROW
INSERT INTO
userinfohisory(tnUserName,tnPassword,tnEmail,tnFullName,tnPhoneNumber,tnState,tnAtTime)
VALUES(old.uUserName, old.uUserEmail, old.uUserPassword, old.uFullName,
old.uUserPhoneNumber, old.uStatus, NOW());


-- recorde of the Admin update info history
DROP TRIGGER IF EXISTS adminInfoHistory;
CREATE TRIGGER  adminInfoHistory
AFTER UPDATE ON admins
FOR EACH ROW
INSERT INTO
adminhistory(ahFullName,ahUserName,ahPassword,ahEmail,ahAtTime)
VALUES(old.aFullName, old.aUserName, old.aPassword, old.aEmail, NOW());
show triggers
```

PROCEDURES

Get a list of all registered accounts with their types in the database

```
DROP PROCEDURE IF EXISTS allAccounts;

DELIMITER //

CREATE PROCEDURE allAccounts()

BEGIN

SELECT aFullName, 'Admin' as `User Type`  FROM admins union

SELECT uUserName, 'User' as `User Type`  FROM users union

SELECT gFullName, 'Guest' as `User Type` FROM guests union

SELECT cOwnerFullName, 'Company' as `User Type` FROM companies;

END //

DELIMITER ;

-- for testing: call allAccounts;
```

Procedure for decreasing from the number of rooms each time we add new reservation:

```
-- To decrase the number of rooms each time a customer reserves
DROP PROCEDURE IF EXISTS decreserooms;
DELIMITER $$
CREATE PROCEDURE decreserooms(IN dateStart DATE, IN dateEnd DATE,IN roomID INT,
IN numOfRooms INT)
BEGIN
        WHILE dateStart <= dateEnd DO
           UPDATE `m&a_hotels`.`rates` SET `rARoomsNum` = `rARoomsNum` -
        numOfRooms
           WHERE (`rRoomsID` = roomID and `rDate` = dateStart);
```

```
    SET dateStart = date_add(dateStart, INTERVAL 1 DAY);
 END WHILE;
END$$
DELIMITER ;
-- CALL decreserooms('2021-01-01','2022-12-31',1, 5);
-- will be called from inside addReservation procedure
```

Procedure for inserting new reservations to the database

Note (the booking status is pending by default)

```
DROP PROCEDURE IF EXISTS addReservation;
-- Procedure for inserting new reservations to the database Note (the booking
status is pending by default
DELIMITER $$
CREATE PROCEDURE addReservation (IN hotelID INT,IN RoomID INT, IN guestID
INT,IN rateTypeID INT, IN dateFrom DATE, IN dateTo DATE,IN numOfRooms INT,
IN psNum INT, IN ChildrenNum INT, ,bPaymentID INT)
BEGIN
    DECLARE bookingID INT;
    INSERT INTO `m&a_hotels`.`bookings` (`bHotelID`,
    `bGuestID`,`bRateTypeID`, `bDateFrom`, `bDateTo`,  `bNumOfGuests`,
    `bNumOfChildren`, `bPaymentID`) VALUES (hotelID, guestID,rateTypeID,
    dateFrom, dateTo, psNum, ChildrenNum, bPaymentID);
    -- to get the booking id from the bookings table to insert it to the
    roomsBooked table
    SET bookingID = LAST_INSERT_ID();
    INSERT INTO `m&a_hotels`.`roomsbooked` (`rbBookingID`, `rbRoomID`,
    `rbNumOfRooms`) VALUES (bookingID, RoomID, numOfRooms);
    -- to decrase the reserved rooms from the
    CALL decreserooms(dateFrom,dateTo,RoomID, numOfRooms);
END$$
DELIMITER ;

-- For testing the procedure:
--  CALL addReservation ('1','1','1','1','2020-05-29','2020-06-01', '2', '3', '1');
```

VIEWS

- Create a view to view the list of hotels who have available rooms from now and in the future:

  ```
  CREATE OR REPLACE VIEW available_hotels AS
  SELECT * FROM hotels h
  JOIN rooms r ON h.hHotelID = r.rHotelID
  JOIN rates rs ON r.rRoomID = rs.rRoomsID
  WHERE rs.rARoomsNum > 0 AND rs.rDate >= now()
  ```

  Note: in order for this view to work you have to make sure you have bookings in rates table for feature dates.

- Create View to show the number of hotels in each city:

  Using sql:

  ```
  CREATE VIEW `no_of_hotels_in_each_city` AS
    SELECT
      `hotels`.`hCity` AS `City`,
     COUNT(`hotels`.`hCity`) AS `Number of hotels`
   FROM
      `hotels`
    GROUP BY `hotels`.`hCity`
  ```

CONCLUSION

This Database has the ability to store the data for Kurdistan region hotels and it’s capable of reserving and storing customers accounts for more than 20,000 customers.

The outcome of this database is fully managed data and it is:

- Easy to fitch and select using the join techniques.
- Takes less storage to store huge data.
- Takes less time to execute the queries.
- Effective multiple tasks in one command.
- Manageable easily.
- Flexible data input and output.
- Easy maintenance.
- The ability to link more tables or columns to the tables later.
- Using the third form technique for storing the data.

Notes:

1. I haven’t used tables for each city. If I do so the diagram will be very complicated to understand and we have only a few cities in Kurdistan region.
2. I have reduced the codes as much as I could due to the limited time and to make it take less time for presenting and understanding.