Q1. Design a database schema that can handle the operations of a room reservation system for a global hotel chain. Ensure the schema is scalable, maintainable, and supports necessary business operations.

This database schema for a global chain's room reservation system is divided into two parts:

Step 1: Entity Relationship Diagram (ERD)

In this step, we define various entities such as Hotel, Rooms, Employees, etc., that are crucial to our database. We also illustrate the relationships between these entities in the ER Diagram.

Step 2: Create SQL Statements

Here, I converted the previously created ERD diagram into SQL CREATE statements, specifying each entity's primary key, foreign key, and their respective relationships.

Step 1: ERD Diagram

In this step, I have created a list of entities to be included in the schema:

- **Hotel**: Captures attributes about each hotel branch - like hotel id, city, state, country, contact information and so on.
- **Room**: Captures attributes about all the rooms like room type, capacity, layout, room rate etc.
- **Employee**: Captures attributes about all employees working for the hotel chain like first name, last name, email, phone number etc.
- **Customer**: Captures attributes about customers reserving rooms in a hotel/hotels like name, address and first_reservation_date.
- **Reservation**: Captures reservation details made by a customer like reservation id, check-in / check-out dates, total cost etc..
- **Payment**: Captures payment details for a particular reservation ID like date, amount, method of payment and currency type.
- **Room Service Requests**: Captures room service requests made by customers during their stay.
- **Holiday Pricing**: Captures holiday surge pricing information for each hotel.

This diagram shows the types of relationships between the entities, such as:

- **One-to-One Relationship**: One room can be reserved by only one reservation and vice versa.
- **One-to-Many Relationship**: One customer can make multiple reservations.

Notation:

- 0..* - 0 to many
- 1..* - 1 to many
- 1..1 - 1 and only 1

Certain assumptions were made while establishing the relationships between entities in this ER diagram which are as follows -

- A certain employee can only work in one of the hotel branches in the global chain.

- If the reservation check in - check out time is the same, multiple rooms in the same hotel can be booked under the same reservation id. For eg: A bigger family of 5-10 traveling together might need multiple rooms and would ideally want to book it under the same id to make things easier while checking in.

- If the check in - check out time does not overlap, every time the same room is booked by multiple parties - the reservation id is unique.

- A reservation_id for a room/rooms in a hotel can have multiple payment_ids considering that one of the payment methods failed while attempting to complete the transaction online so the database records all the different methods used to complete the payment.

Step 2: SQL Queries

In this step, we create tables for each entity mentioned in the ER diagram from the previous step. Each entity is represented as a separate table. We have defined primary keys for each entity and referenced foreign keys to their respective tables.

Notes -

* We had to add a new table *Reservation_Room_Details* to capture the **many to many relationship** between Reservation and Room since one room can have multiple reservations at different periods of time which leads to multiple reservation ids and one reservation id can link to multiple rooms booked during the same time in the case of a group checkout.

QUESTION 2:

Using the provided CSV files. Please create a CSV file that contains the following headers:

Column Name	Data Type	Notes
personId	String UUID (unique)	
name	String	Full name
email	String (unique)	
dob	Date string	
address1	String (optional)	An address should only be saved if it can form a full address string
address2	String (optional)	
city	String (optional)	
state	String (optional)	
zip	String (optional)	
majorIds	Comma separated string	
bedId	String	

Please include a description of any data cleaning policies that you applied.

Preliminary Data Preprocessing - There are **no null values** in the data set

COLUMN	**NOTES**
personId	This column is directly taken from the *persons_data* table (Column A)

name	Using the **CONCAT** function, I combined the **firstName** and **lastName** columns (B and C) from the *persons_data* table to obtain the full name of the student
email	This column is directly taken from the *persons_data* table (Column D)
dob	This column is directly taken from the *persons_data table (Column E)*

address1	This column is directly taken from the *persons_data* table (Column F)
address2	This column is the **concatenated** version of address1 and zip code
city	Using the **FIND** function, find the first and second occurrence of the comma in the address column in *persons_data* to extract the string between the commas, which represents the city

state	We find the second instance of comma using the **FIND** function in Excel; the text after this index is the state column
zip code	Since the zip code column was blank, I populated this field with random 5-digit numbers using the **RANDBETWEEN** function in Excel
majorIds	In the *majors_data* table, each major (e.g., Comp Sci) has multiple major_ids, and each student has multiple majors To retrieve all the major_ids related to a particular student, I delimited the majors from the majors column in *persons_data* (column G). Using an array formula with **TEXTJOIN**, I fetched all the major ids for each major and created comma-separated strings. Finally, I combined all the strings using the **CONCAT** function to obtain comma-separated strings in the majorIds column. Some minor data trimming was required as each major id was separated by a comma and a space.

bedId	I combined the buildingName, roomName, and bedName from the *inventory_data* table using the **CONCAT** function. Then, I matched it to the person_id in the *occupancy_data* table using **VLOOKUP** Now, we have the bed_id for each person matched with their person_id and can populate the bed_id column in the new table

Completed by -

Ballori Ghosh
06.25.24