Housing Cloud - Analytics Engineer Technical Interview

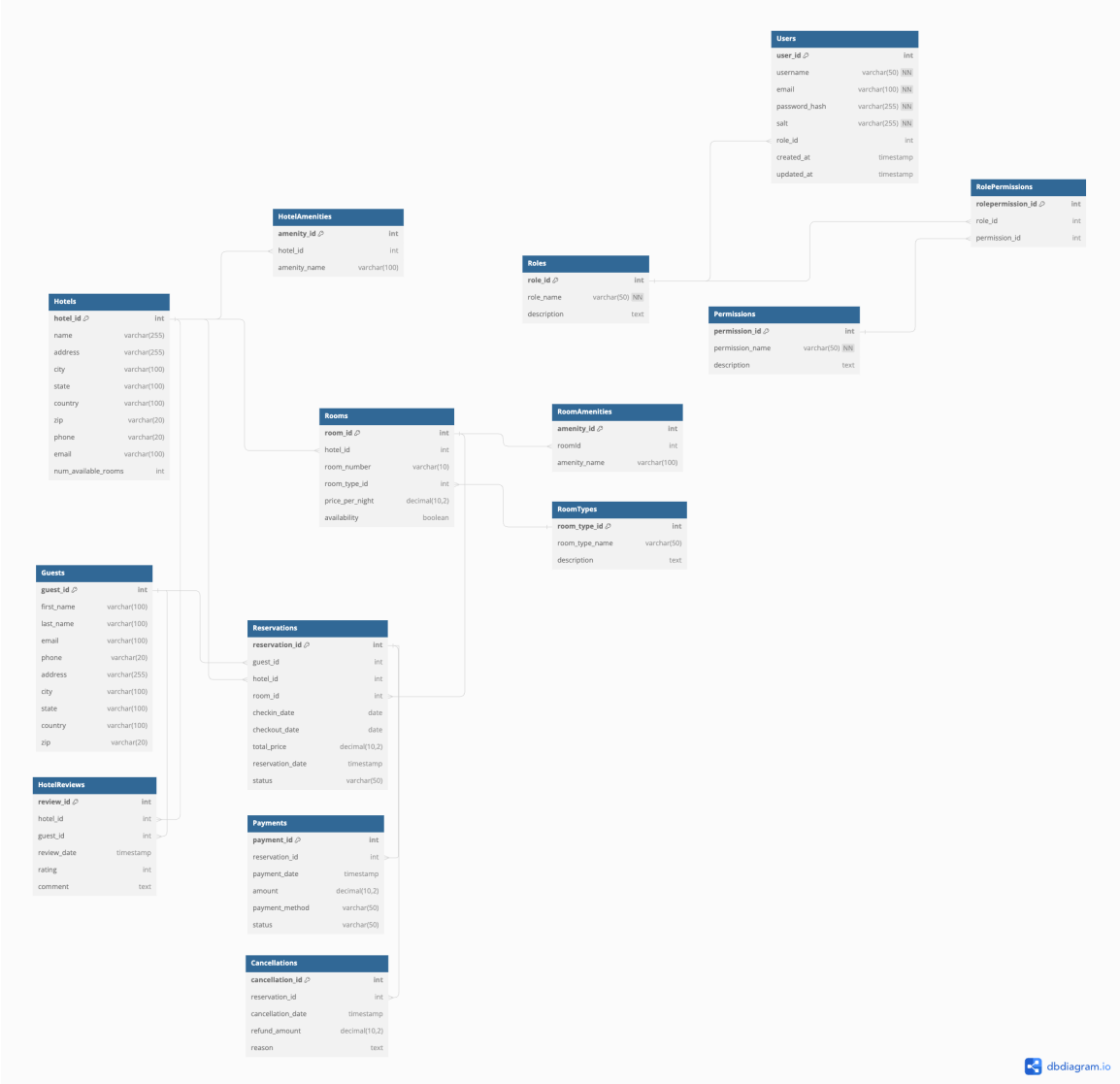
By: Endi Hajdari **Date:** Monday, June 24, 2024

Question 1: 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.

▼ Database Schema Diagram and SQL Code

▼ Database Schema Diagram

Database schema diagram was created using **dbdiagram.io.**



▼ DBML and SQL Code

This code was generated using **dbdiagram.io.**

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

// Hotels Table
Table Hotels {
    hotel_id int [pk, increment]
```

```
    name varchar(255)
    address varchar(255)
    city varchar(100)
    state varchar(100)
    country varchar(100)
    zip varchar(20)
    phone varchar(20)
    email varchar(100)
    num_available_rooms int
}

// Rooms Table
Table Rooms {
    room_id int [pk, increment]
    hotel_id int [ref: > Hotels.hotel_id]
    room_number varchar(10)
    room_type_id int [ref: > RoomTypes.room_type_id]
    price_per_night decimal(10, 2)
    availability boolean [default: true]
}

// RoomTypes Table
Table RoomTypes {
    room_type_id int [pk, increment]
    room_type_name varchar(50)
    description text
}

// RoomAmenities Table
Table RoomAmenities {
    amenity_id int [pk, increment]
    roomId int [ref: > Rooms.room_id]
    amenity_name varchar(100)
}

// HotelAmenities Table
```

```
Table HotelAmenities {
    amenity_id int [pk, increment]
    hotel_id int [ref: > Hotels.hotel_id]
    amenity_name varchar(100)
}

// Guests Table
Table Guests {
    guest_id int [pk, increment]
    first_name varchar(100)
    last_name varchar(100)
    email varchar(100) [unique]
    phone varchar(20)
    address varchar(255)
    city varchar(100)
    state varchar(100)
    country varchar(100)
    zip varchar(20)
}

// Reservations Table
Table Reservations {
    reservation_id int [pk, increment]
    guest_id int [ref: > Guests.guest_id]
    hotel_id int [ref: > Hotels.hotel_id]
    room_id int [ref: > Rooms.room_id]
    checkin_date date
    checkout_date date
    total_price decimal(10, 2)
    reservation_date timestamp [default: `current_timestamp
    status varchar(50) [default: 'Booked']
}

// Payments Table
Table Payments {
    payment_id int [pk, increment]
```

```
    reservation_id int [ref: > Reservations.reservation_id]
    payment_date timestamp [default: `current_timestamp`]
    amount decimal(10, 2)
    payment_method varchar(50)
    status varchar(50) [default: 'Completed']
}

// Cancellations Table
Table Cancellations {
    cancellation_id int [pk, increment]
    reservation_id int [ref: > Reservations.reservation_id]
    cancellation_date timestamp [default: `current_timestar
    refund_amount decimal(10, 2)
    reason text
}

// HotelReviews Table
Table HotelReviews {
    review_id int [pk, increment]
    hotel_id int [ref: > Hotels.hotel_id]
    guest_id int [ref: > Guests.guest_id]
    review_date timestamp [default: `current_timestamp`]
    rating int [note: 'Rating from 0 to 100']
    comment text
}

// Users Table
Table Users {
    user_id int [pk, increment]
    username varchar(50) [unique, not null]
    email varchar(100) [unique, not null]
    password_hash varchar(255) [not null]
    salt varchar(255) [not null]
    role_id int [ref: > Roles.role_id]
    created_at timestamp [default: `current_timestamp`]
    updated_at timestamp
```

```
}

// Roles Table
Table Roles {
    role_id int [pk, increment]
    role_name varchar(50) [unique, not null]
    description text
}

// Permissions Table
Table Permissions {
    permission_id int [pk, increment]
    permission_name varchar(50) [unique, not null]
    description text
}

// RolePermissions Table
Table RolePermissions {
    rolepermission_id int [pk, increment]
    role_id int [ref: > Roles.role_id]
    permission_id int [ref: > Permissions.permission_id]
}
```

```
// MySQL Code generated from DBML
CREATE TABLE "Hotels" (
  "hotel_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY K
  "name" varchar(255),
  "address" varchar(255),
  "city" varchar(100),
  "state" varchar(100),
  "country" varchar(100),
  "zip" varchar(20),
  "phone" varchar(20),
  "email" varchar(100),
  "num_available_rooms" int
```

```
);

CREATE TABLE "Rooms" (
  "room_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KE
  "hotel_id" int,
  "room_number" varchar(10),
  "room_type_id" int,
  "price_per_night" decimal(10,2),
  "availability" boolean DEFAULT true
);

CREATE TABLE "RoomTypes" (
  "room_type_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMA
  "room_type_name" varchar(50),
  "description" text
);

CREATE TABLE "RoomAmenities" (
  "amenity_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY
  "roomId" int,
  "amenity_name" varchar(100)
);

CREATE TABLE "HotelAmenities" (
  "amenity_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY
  "hotel_id" int,
  "amenity_name" varchar(100)
);

CREATE TABLE "Guests" (
  "guest_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY K
  "first_name" varchar(100),
  "last_name" varchar(100),
  "email" varchar(100) UNIQUE,
  "phone" varchar(20),
  "address" varchar(255),
```

```
  "city" varchar(100),
  "state" varchar(100),
  "country" varchar(100),
  "zip" varchar(20)
);

CREATE TABLE "Reservations" (
  "reservation_id" INT GENERATED BY DEFAULT AS IDENTITY PRI
  "guest_id" int,
  "hotel_id" int,
  "room_id" int,
  "checkin_date" date,
  "checkout_date" date,
  "total_price" decimal(10,2),
  "reservation_date" timestamp DEFAULT (current_timestamp),
  "status" varchar(50) DEFAULT 'Booked'
);

CREATE TABLE "Payments" (
  "payment_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY
  "reservation_id" int,
  "payment_date" timestamp DEFAULT (current_timestamp),
  "amount" decimal(10,2),
  "payment_method" varchar(50),
  "status" varchar(50) DEFAULT 'Completed'
);

CREATE TABLE "Cancellations" (
  "cancellation_id" INT GENERATED BY DEFAULT AS IDENTITY PR
  "reservation_id" int,
  "cancellation_date" timestamp DEFAULT (current_timestamp)
  "refund_amount" decimal(10,2),
  "reason" text
);

CREATE TABLE "HotelReviews" (
```

```
  "review_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY
  "hotel_id" int,
  "guest_id" int,
  "review_date" timestamp DEFAULT (current_timestamp),
  "rating" int,
  "comment" text
);

CREATE TABLE "Users" (
  "user_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KE
  "username" varchar(50) UNIQUE NOT NULL,
  "email" varchar(100) UNIQUE NOT NULL,
  "password_hash" varchar(255) NOT NULL,
  "salt" varchar(255) NOT NULL,
  "role_id" int,
  "created_at" timestamp DEFAULT (current_timestamp),
  "updated_at" timestamp
);

CREATE TABLE "Roles" (
  "role_id" INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KE
  "role_name" varchar(50) UNIQUE NOT NULL,
  "description" text
);

CREATE TABLE "Permissions" (
  "permission_id" INT GENERATED BY DEFAULT AS IDENTITY PRI
  "permission_name" varchar(50) UNIQUE NOT NULL,
  "description" text
);

CREATE TABLE "RolePermissions" (
  "rolepermission_id" INT GENERATED BY DEFAULT AS IDENTITY
  "role_id" int,
  "permission_id" int
);
```

```
COMMENT ON COLUMN "HotelReviews"."rating" IS 'Rating from (

ALTER TABLE "Rooms" ADD FOREIGN KEY ("hotel_id") REFERENCES

ALTER TABLE "Rooms" ADD FOREIGN KEY ("room_type_id") REFERE

ALTER TABLE "RoomAmenities" ADD FOREIGN KEY ("roomId") REFE

ALTER TABLE "HotelAmenities" ADD FOREIGN KEY ("hotel_id") F

ALTER TABLE "Reservations" ADD FOREIGN KEY ("guest_id") REF

ALTER TABLE "Reservations" ADD FOREIGN KEY ("hotel_id") REF

ALTER TABLE "Reservations" ADD FOREIGN KEY ("room_id") REFE

ALTER TABLE "Payments" ADD FOREIGN KEY ("reservation_id") F

ALTER TABLE "Cancellations" ADD FOREIGN KEY ("reservation_i

ALTER TABLE "HotelReviews" ADD FOREIGN KEY ("hotel_id") REF

ALTER TABLE "HotelReviews" ADD FOREIGN KEY ("guest_id") REF

ALTER TABLE "Users" ADD FOREIGN KEY ("role_id") REFERENCES

ALTER TABLE "RolePermissions" ADD FOREIGN KEY ("role_id") F

ALTER TABLE "RolePermissions" ADD FOREIGN KEY ("permission_
```

▼ Explanation

▼ Entities and Attributes for the Hotel Chain Operations and Reservation System

1. **Hotels Table:** Stores information regarding the hotels that make-up the chain and contains the fields listed below.
    - `hotel_id`: Primary key uniquely identifying each hotel; int (auto-incremented)
    - `name`: Name of the hotel; varchar(255)
    - `address`: Street address of the hotel; varchar(255)
    - `city`: City where the hotel is located; varchar(100)
    - `state`: State or province where the hotel is located; varchar(100)
    - `country`: Country where the hotel is located; varchar(100)
    - `zip`: Postal code of the hotel's location; varchar(20)
    - `phone`: Contact phone number for the hotel; varchar(20)
    - `email`: Contact email address for the hotel; varchar(100)
    - `num_available_Rooms`: Number of rooms available in the hotel; int
2. **Rooms Table:** Stores basic information regarding each of the rooms in a given hotel.
    - `room_id`: Primary key uniquely identifying each room in each hotel; int (auto-incremented)
    - `hotel_id`: Foreign key referencing `Hotels.hotel_id`, identifying the hotel to which the room belongs; int
    - `room_num`: Unique identifier for a given room within a hotel; varchar(10)
    - `room_type_id`: Foreign key referencing `RoomTypes.room_type_id`, specifying the type of room; int
    - `price_per_night`: The price per night for a given room; decimal (10, 2)
    - `availability`: Field which indicates whether a room is currently available; boolean [default: True]
3. **RoomTypes Table:** Stores information about the different types of rooms available in the hotels.

- `room_type_id`: Primary key uniquely identifying each room type in each hotel; int (auto-incremented)
- `room_type_name`: The name or category of the room type (e.g., "Single Room"); varchar(50)
- `description`: Field which provides a detailed description of the room type (e.g., the size of the room); text (or varchar if the descriptions are not long)

4. **RoomAmenities Table:** Stores information about the amenities available in each room.
   - `amenity_id`: Primary key uniquely identifying each room amenity; int (auto-incremented)
   - `room_id`: Foreign key referencing `Rooms.room_id` which links the reservation to a specific room within a hotel; int
   - `amenity_name`: The name of the amenity (e.g., "Wifi"); varchar(100)
5. **HotelAmenities Table:** Stores information about the amenities available in each hotel.
   - `amenity_id`: Primary key uniquely identifying each room amenity; int (auto-incremented)
   - `hotel_id`: Foreign key referencing referencing `Hotels.hotel_id` which links the reservation to a specific hotel of the chain; int
   - `amenity_name`: The name of the amenity (e.g., "Fitness Centre"); varchar(100)
6. **Guests Table:** Stores detailed information about guests who have interacted with the hotel chain.
   - `guest_id`: Primary key uniquely identifying each guest; int (auto-incremented)
   - `first_name`: The first name of the guest; varchar(100)
   - `last_name`: The last name of the guest; varchar(100)
   - `email`: The email address of the guest; varchar(100) [unique constraint]

- `address`: The street address of the guest; varchar(255)
- `city`: The city where the guest resides; varchar(100)
- `state`: The state or province where the guest resides; varchar(100)
- `country`: The country where the guest resides; varchar(100)
- `zip`: The postal code of the guest's address; varchar(20)

7. **Reservations Table:** Stores detailed information about the room bookings made by guests.
    - `reservation_id`: Primary key uniquely identifying each reservation; int (auto-incremented)
    - `guest_id`: Foreign key referencing `Guests.guest_id` which links the reservation to a specific guest; int
    - `hotel_id`: Foreign key referencing referencing `Hotels.hotel_id` which links the reservation to a specific hotel of the chain; int
    - `room_id`: Foreign key referencing `Rooms.room_id` which links the reservation to a specific room within a hotel; int
    - `checkin_date`: The date when the guest is scheduled to check-in; date
    - `checkout_date`: The date when the guest is scheduled to check-out; date
    - `total_price`: The total price of the reservation (precise financial calculations, capturing the cost of the stay including taxes and any additional charges); decimal (10, 2)
    - `reservation_date`: The date and time when the reservation was made; timestamp [default value: current timestamp]
    - `status`: The current status of the reservation (e.g., “Booked”); varchar(50)
8. **Payments Table:** Stores information about payment transactions associated with reservations.

- `payment_id`: Primary key uniquely identifying each payment transaction; int (auto-incremented)
- `reservation_id`: Foreign key referencing `Reservations.reservation_id` which links the payment to a specific reservation; int
- `payment_date`: The date and time when the payment was made; timestamp
- `amount`: The amount paid in the transaction; decimal (10, 2)
- `payment_method`: The method used for the payment (e.g., "Credit Card"); varchar(50)
- `status`: The date when the guest is scheduled to check-out; date
- `total_price`: The total price of the reservation (precise financial calculations, capturing the cost of the stay including taxes and any additional charges); decimal (10, 2)
- `reservation_date`: The date and time when the reservation was made; timestamp [default value: current timestamp]
- `status`: The current status of the payment (e.g., "Successful"); varchar(50)

9. **Cancellations Table:** Stores detailed information regarding reservation cancellations.
    - `cancellation_id`: Primary key uniquely identifying each cancellation entry; int (auto-incremented)
    - `reservation_id`: Foreign key referencing `Reservations.reservation_id` which links the payment to a specific reservation; int
    - `cancellation_date`: The date and time when the cancellation was made; timestamp
    - `refund_amount`: The amount refunded to the guest due to the cancellation; decimal (10, 2)
    - `reason`: The reason for the guest cancelling their reservation; text
10. **HotelReviews Table:** Stores information surrounding reviews left by guests for the hotels.

- `review_id`: Primary key uniquely identifying each review left by guests; int (auto-incremented)
- `hotel_id`: Foreign key referencing referencing `Hotels.hotel_id` which links the reservation to a specific hotel of the chain; int
- `guest_id`: Foreign key referencing `Guests.guest_id` which links the reservation to a specific guest; int
- `review_date`: The date and time when the review was made; timestamp
- `rating`: The guest's rating of the hotel, on a scale from 0 to 100; int
- `additional_comment`: The additional comments submitted by the guest; text

> **Note:** I am not well-versed in the methods used to protect data, however recognize the importance of implementing security measures. I consulted this Stack post to aid in the creation of the following tables and fields: https://stackoverflow.com/questions/46016139/best-user-role-permissions-database-design-practice

11. **Users Table:** Stores information about the users who have access to the room reservation system.
    - `user_id`: Primary key uniquely identifying each user; int (auto-incremented)
    - `username`: The username of the user; varchar(50)
    - `email`: The email address of the user; varchar(100) [unique, not null constraint]
    - `password_hash`: The hashed version of the user's password; varchar(255) [not null constraint]
    - `salt`: The salt used in hashing the user's password; varchar(255) [not null constraint]
    - `role_id`: The specific role assigned to each user; int

- `created_at`: The date and time when the user account was created; timestamp
- `updated_at`: The date and time the user account was last updated; timestamp

12. **Roles Table:** Stores information about the different roles that users can have in the system.
    - `role_id`: Primary key uniquely identifying each role; int (auto-incremented)
    - `role_name`: The name of the role (e.g., "Admin"); varchar(50)
    - `description`: The description of the role (e.g., "Full accessibility"); text
13. **Permissions Table:** Stores information about the different permissions that can be assigned to roles.
    - `permission_id`: Primary key uniquely identifying each permission; int (auto-incremented)
    - `permission_name`: The name of the permission (e.g., "Read Only"); varchar(50)
    - `description`: The description of the permission (e.g., "Can only read existing records"); text
14. **RolePermissions Table:** Stores information about the different permissions that can be assigned to roles.
    - `role_permission_id`: Primary key uniquely identifying each role-permission; int (auto-incremented)
    - `permission_name`: Foreign key referencing `Roles.role_id` which inks the role-permission entry to a specific role; int
    - `permission_id`: Foreign key referencing `Permissions.permission_id` which links the role-permission entry to a specific permission; int

▼ Scalability, Maintainability, and Support of Business Operations

▼ Scalability and Maintainability

- The database schema above is **modular** in its design and is structured into separate tables (e.g., Hotels, Guests) that may be scaled independently. Additional hotels, rooms, or guests can be added without affecting other parts of the system.
- The database schema is **logically organized** with tables representing distinct entities making it easier to understand and simpler to maintain over time.
- The database schema is scalable as data volume increases as it is **properly indexed** and **primary/foreign key relationships** are defined.
- Use of **foreign key constraints** ensures data integrity, while relationships between entities are clearly defined and maintained. Use of `AUTO_INCREMENT` primary keys ensures that tables can grow without key conflicts.
- The database schema is **normalized** which allows for efficient storage and retrieval of data, even as the database grows.

▼ Support of Business Operations

Typical business operations of a room reservation system for a global hotel chain include booking management, guest management, room and amenity management, hotel operations, customer feedback and reviews, marketing and sales, and security.

- **Booking Management**: The schema supports the core operations of managing reservations, tracking room availability, and handling payments.
- **Guest Interaction**: Detailed guest information which enables the chain to facilitate personalized service and communication.
- **Review and Feedback**: The schema allows guests to provide feedback and ratings, which can be used for improving services and marketing of the hotel chain.
- **Security**: Implementation of user authentication, authorization (i.e., roles and permissions), and sensitive data handling (i.e., encryption) ensures some level of security within the system.

- **Analytics and Reporting**: Well-structured and logical schema enables data to be efficiently queried and aggregated for business intelligence purposes, thereby supporting decision-making and analysis.

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. If there are any rows that could not be included in the final sheet, please include them and indicate why/how it was not possible to include them.

▼ Final CSV Files

final_data1.csv

missing_data_rows.csv

▼ Python Code

Untitled1.ipynb

▼ Summary of Data Cleaning Policies Used