

Project: Build Data Mart in SQL

Oskar Wolf





Table of Contents

01 Introduction and Overview

02 Database Design

03 Populating Mock Data

04 Testing and Results



01

Introduction and Overview

Scope and Purpose



Introduction

Welcome to my presentation on the database design and implementation of the **B**ooking and Accommodation System. In this session, I will delve into the architecture and structure of the database that underpins the seamless functioning of the system. This database serves as the backbone of an application, supporting various operations related to bookings, accommodations, users, and more.





Purpose and Scope:

This presentation aims to offer a comprehensive understanding of the design and implementation of the Booking and Accommodation System's database. Let's dive into the core components of the database, including entities, attributes, relationships, and how data is logically organized.



Importance of Database Design:

Effective database design is a critical factor in the success of any software application. It guarantees data accuracy, efficient storage, and seamless information retrieval. Within the Booking and Accommodation System, a well-designed database enables the streamlined management of bookings, accommodations, users, reviews, and other essential elements. It significantly contributes to an enhanced user experience by ensuring quick and precise data access.





02

Database Design

Using DbSchema,
Dbeaver, mySQL, and
Create.sql



User Details Group

User Table

The "users" table is designed to store user information. It consists of columns such as "userid," serving as an auto-incremented primary key, "name" for the user's name, "email" for the user's email address, "password" for their password, "profilepicture" for a URL to their profile picture, "phonenummer" for the user's phone number, "gender" for their gender, "nationality" for their nationality, and "age" to indicate the user's age. This table is intended to hold essential user data with appropriate constraints on certain columns.

```
CREATE TABLE users (
  userid          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name            VARCHAR(100) NOT NULL,
  email           VARCHAR(500) NOT NULL,
  password        VARCHAR(500) NOT NULL,
  profilepicture  VARCHAR(500),
  phonenummer     VARCHAR(500) NOT NULL,
  gender          CHAR NOT NULL,
  nationality      VARCHAR(50) NOT NULL,
  age            INT NOT NULL
);
```

Host Table

The "hosts" table is established to manage host-related information. It includes columns such as "hostid," an auto-incremented unique identifier for hosts, "aboutme" to store textual descriptions about the hosts, "verificationstatus" to track their verification status, "joindate" for recording the date of joining, and "userid" as a reference to link with the user who is a host.

```
CREATE TABLE hosts (
  hostid          INT NOT NULL AUTO_INCREMENT,
  aboutme         TEXT NOT NULL,
  verificationstatus INT NOT NULL,
  joindate        DATETIME NOT NULL,
  userid          INT NOT NULL,
  CONSTRAINT `pk hosts` PRIMARY KEY ( hostid, userid )
);
```

Guest Table

The "guests" table is designed to manage guest-related data. It includes columns like "guestid," an auto-incremented unique identifier for guests, "userid" to establish a connection with the corresponding user, and "joindate" to record the date of joining as a guest.

```
CREATE TABLE guests (
  guestid         INT NOT NULL AUTO_INCREMENT,
  userid          INT NOT NULL,
  joindate        DATETIME NOT NULL,
  CONSTRAINT `pk guests` PRIMARY KEY ( guestid, userid )
);
```


Languages Table

The Languages table stores preferred languages of guests, including LanguageID and LanguageName. It provides a standardized list of languages for guests to choose from, ensuring consistency and facilitating language-based features within the platform. The table enables efficient language filtering and matching for accommodations, enhancing the user experience.

Verification Status Table

The Verifications table stores host verification statuses, including VerificationID and VerificationStatus. It ensures host credibility and trustworthiness by verifying their identity or relevant information through methods such as email, phone, ID, or social media verification. The table enables displaying verified badges on host profiles and implementing security measures to enhance user trust.

Host Bank Table

The Host Bank Accounts table is integral to the financial system, storing host-linked bank account details for receiving payments. It includes fields like Host_ID, Bank_Name, Account_Number, Account_Type, Routing_Number, and Account_Holder_Name. By enabling hosts to receive earnings directly, it ensures smooth transactions and compliance. This table enhances trust, streamlines payments, and aids financial reporting.

UserAccommodation Table

This table represents the relationship between users and accommodations, indicating which users have a connection to specific accommodations. The UserAccommodation table allows for a many-to-many relationship between users and accommodations, as a user can be associated with multiple accommodations, and an accommodation can be associated with multiple users.

```
CREATE TABLE Languages (
  `LanguageID` INT NOT NULL AUTO_INCREMENT,
  `LanguageName` VARCHAR(100),
  `UserID` INT NOT NULL,
  CONSTRAINT `pk Languages` PRIMARY KEY ( `LanguageID`, `UserID` )
);
```

```
CREATE TABLE Verifications (
  `VerificationID` INT NOT NULL AUTO_INCREMENT,
  `VerificationStatus` BOOLEAN NOT NULL,
  `HostID` INT NOT NULL,
  CONSTRAINT `pk Verifications` PRIMARY KEY ( `VerificationID`, `HostID` ),
  CONSTRAINT `fk Verifications Hosts` FOREIGN KEY ( `HostID` ) REFERENCES hosts( hostid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

```
CREATE TABLE HostBank (
  `HostBankAccountID` INT NOT NULL AUTO_INCREMENT,
  `HostID` INT NOT NULL,
  `AccountNumber` BIGINT NOT NULL,
  `BankName` VARCHAR(100) NOT NULL,
  `AccountHolderName` VARCHAR(100),
  `RoutingNumber` VARCHAR(500) NOT NULL,
  CONSTRAINT `pk Host Bank Accounts` PRIMARY KEY ( `HostBankAccountID`, `HostID` ),
  CONSTRAINT `fk Host Bank Accounts Hosts` FOREIGN KEY ( `HostID` ) REFERENCES hosts( hostid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

```
CREATE TABLE useraccommodation (
  useraccommodationid INT NOT NULL AUTO_INCREMENT,
  userid INT NOT NULL,
  accomid INT NOT NULL,
  CONSTRAINT `pk useraccommodation` PRIMARY KEY ( useraccommodationid, accomid )
);
```

Accommodation Details Group

Accom Table

The "Accom" table stores information about the available accommodations. It includes attributes such as accommodation title, description, location, amenities, and rules. The table is linked to the "Hosts" table through a foreign key relationship, indicating which host is responsible for the accommodation. It also has relationships with other entities such as bookings, reviews, photos, tags, and facilities, allowing for comprehensive management of the accommodation listings.

```
● CREATE TABLE accom (
  accomid INT NOT NULL AUTO_INCREMENT ,
  hostid INT NOT NULL,
  title VARCHAR(500) NOT NULL,
  description TEXT(0) NOT NULL,
  createdate DATETIME,
  CONSTRAINT `pk accommodations` PRIMARY KEY (accomid, hostid),
  CONSTRAINT `fk accommodations hosts` FOREIGN KEY (hostid) REFERENCES hosts(hostid) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

SpecialOffers Table

The Special Offers table is a core element of the accommodation system, overseeing discounts and deals for guests. It records specifics like Offer_ID, Accommodation_ID, Start_Date, End_Date, Discount_Percentage, and Terms, empowering hosts to attract guests with special rates. This streamlines managing offers, enhancing bookings and customer satisfaction, while also enabling the system to monitor the effectiveness of promotions, ultimately optimizing revenue and guest experiences.

```
● CREATE TABLE specialoffers (
  offerid INT NOT NULL AUTO_INCREMENT ,
  accomid INT NOT NULL ,
  offertitle VARCHAR(500) NOT NULL ,
  offerdescription VARCHAR(500) ,
  offerdiscount DECIMAL NOT NULL ,
  offerexpirationdate DATETIME NOT NULL ,
  hostid INT NOT NULL ,
  CONSTRAINT `pk special offers` PRIMARY KEY ( offerid, accomid, hostid ),
  CONSTRAINT `fk special offers accommodations` FOREIGN KEY ( accomid ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk special offers hosts` FOREIGN KEY ( hostid ) REFERENCES hosts( hostid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomCity (Link Table)

The AccomCity table stores the relationship between accommodations and cities, enabling easy categorization and search based on specific cities.

```
● CREATE TABLE accomcity (
  accomcityid INT NOT NULL AUTO_INCREMENT ,
  accomid INT NOT NULL,
  cityid INT NOT NULL,
  CONSTRAINT `pk accommodationcity` PRIMARY KEY (accomcityid, accomid, cityid),
  CONSTRAINT `fk accommodationcity accommodations` FOREIGN KEY (accomid) REFERENCES accom(accomid) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk accommodationcity cities` FOREIGN KEY (cityid) REFERENCES Cities(CityID) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```


AccomFacilities (Link Table)

The AccomFacilities table serves as a link table to establish the relationship between accommodations and facilities. It enables a many-to-many relationship, where an accommodation can have multiple facilities, and a facility can be associated with multiple accommodations.

```
● CREATE TABLE accomfacilities (
  accomfacilityid INT NOT NULL AUTO_INCREMENT ,
  accomid INT NOT NULL ,
  facilityid INT NOT NULL ,
  CONSTRAINT 'pk accommodationfacilities' PRIMARY KEY ( accomfacilityid, accomid, facilityid ),
  CONSTRAINT 'fk accommodationfacilities accommodations' FOREIGN KEY ( accomid ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT 'fk accommodationfacilities facilities' FOREIGN KEY ( facilityid ) REFERENCES facilities( facilityid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomPhotos (Link Table)

The AccomPhotos table links accommodations with their respective photos, allowing for a many-to-many relationship. It facilitates efficient storage and retrieval of photos, enhancing the visual representation of accommodations.

```
● CREATE TABLE accomphotos (
  accomphotoid INT NOT NULL AUTO_INCREMENT ,
  photoid INT NOT NULL ,
  accomid INT NOT NULL ,
  CONSTRAINT 'pk accomphotos' PRIMARY KEY ( accomphotoid, accomid ),
  CONSTRAINT 'fk accomphotos accom' FOREIGN KEY ( accomid ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT 'fk accomphotos photos' FOREIGN KEY ( photoid ) REFERENCES photos( photoid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomTag (Link Table)

The AccommodationTag table links accommodations and tags in a many-to-many relationship. It categorizes accommodations based on relevant tags, aiding in organization and classification.labels.

```
● CREATE TABLE accomtag (
  accomtagid INT NOT NULL AUTO_INCREMENT ,
  accomid INT NOT NULL ,
  tagid INT NOT NULL ,
  CONSTRAINT 'pk accomodationtag' PRIMARY KEY ( accomtagid, accomid, tagid ),
  CONSTRAINT 'fk accomodationtag accommodations' FOREIGN KEY ( accomid ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT 'fk accomodationtag tags' FOREIGN KEY ( tagid ) REFERENCES 'Tags' ( 'TagID' ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomAmenity (Link Table)

This table serves as a link table to establish the relationship between accommodations and amenities. It allows for a many-to-many relationship, as an accommodation can have multiple amenities, and an amenity can be associated with multiple accommodations.

```
● CREATE TABLE AccomAmenity(
  'AccomAmenityID' INT NOT NULL AUTO_INCREMENT ,
  'AccomID' INT NOT NULL ,
  'AmenityID' INT NOT NULL ,
  CONSTRAINT 'pk AccommodationAmenity' PRIMARY KEY ( 'AccomAmenityID', 'AccomID', 'AmenityID' ),
  CONSTRAINT 'fk AccommodationAmenity Accomodations' FOREIGN KEY ( 'AccomID' ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT 'fk AccommodationAmenity Amenities' FOREIGN KEY ( 'AmenityID' ) REFERENCES Amenities( 'AmenityID' ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomCountry (Link Table)

The AccomCountry table establishes the connection between accommodations and countries, allowing for efficient categorization and search based on specific countries.

```
● CREATE TABLE AccomCountry (
  'AccomCountryID' INT NOT NULL AUTO_INCREMENT ,
  'AccomID' INT NOT NULL ,
  'CountryID' INT NOT NULL ,
  CONSTRAINT 'pk AccomodationCity 0' PRIMARY KEY ( 'AccomCountryID', 'AccomID', 'CountryID' ),
  CONSTRAINT 'fk AccomodationCountry Accomodations' FOREIGN KEY ( 'AccomID' ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT 'fk AccomodationCountry Countries' FOREIGN KEY ( 'CountryID' ) REFERENCES Countries( 'CountryID' ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomLocation (Link Table)

The AccomLocation table links accommodations with their respective locations, allowing for efficient search and filtering based on geographical proximity.

```
CREATE TABLE AccomLocation (
  AccomLocationID INT NOT NULL AUTO_INCREMENT ,
  AccomID INT NOT NULL ,
  LocationID INT NOT NULL ,
  CONSTRAINT pk AccommodationLocation PRIMARY KEY ( AccomLocationID, AccomID, LocationID ),
  CONSTRAINT fk AccommodationLocation Accommodations FOREIGN KEY ( AccomID ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT fk AccommodationLocation Locations FOREIGN KEY ( LocationID ) REFERENCES locations( locationid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

AccomRule (Link Table)

This table is used to link specific rules to each accommodation, indicating which rules are applicable to a particular accommodation. The AccommodationRule table helps in organizing and managing the rules associated with each accommodation.

```
CREATE TABLE AccomRule (
  AccomRuleID INT NOT NULL AUTO_INCREMENT ,
  AccomID INT NOT NULL ,
  RuleID INT NOT NULL ,
  CONSTRAINT pk AccommodationRule PRIMARY KEY ( AccomRuleID, AccomID, RuleID ),
  CONSTRAINT fk AccommodationRule Accommodations FOREIGN KEY ( AccomID ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT fk AccommodationRule Rules FOREIGN KEY ( RuleID ) REFERENCES rules( ruleid ) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Amenities Table

The Amenities table contains a list of amenities that are offered in accommodations. Each amenity is identified by a unique AmenityID and is described by an AmenityName. This table helps users to easily find accommodations that provide specific amenities, such as Wi-Fi, parking, pool, or gym facilities.

```
CREATE TABLE Amenities (
  AmenityID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  AmenityName VARCHAR(100)
);
```

Cities Table

The Cities table stores city information, including CityID, CityName, CountryID, and Population. This setup allows for accommodations to be categorized by city, facilitating location-based searches and analysis. The table's design ensures efficient organization of geographic data for better management and user experience.

```
CREATE TABLE Cities (
  CityID INT NOT NULL AUTO_INCREMENT ,
  CityName VARCHAR(100) NOT NULL,
  CountryID INT NOT NULL,
  Population INT NOT NULL,
  CONSTRAINT pk Cities PRIMARY KEY (CityID, CountryID),
  CONSTRAINT fk Cities Countries FOREIGN KEY (CountryID) REFERENCES Countries(CountryID) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Countries Table

The Countries table holds data about countries, featuring attributes like CountryID, CountryName, and CountryCode. This data aids in various operations, such as categorizing accommodations and presenting relevant information based on different countries.

```
CREATE TABLE Countries (
  CountryID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  CountryName VARCHAR(100),
  Code CHAR(3) NOT NULL
);
```




Tags Table

The Tags table stores unique tags associated with accommodations. It helps categorize and identify accommodations based on attributes, making search and filtering easier for users.

```
CREATE TABLE Tags (
  TagID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  TagName VARCHAR(100) NOT NULL
);
```



Facilities Table

The Facilities table stores unique facilities available in accommodations, identified by FacilityID and described by FacilityName. It helps users identify accommodations with desired amenities, enhancing their selection process.

```
CREATE TABLE facilities (
  facilityid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  facilityname VARCHAR(100)
);
```



Location Table

The Locations table maintains distinct accommodation locations with a unique Location_ID. It encompasses details like Address and Latitude/Longitude, enabling precise searches and tailored location suggestions for platform users.

```
CREATE TABLE locations (
  locationid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  locationname VARCHAR(100),
  longitude DECIMAL(8, 4) NOT NULL,
  latitude DECIMAL(8, 4) NOT NULL,
  address VARCHAR(100) NOT NULL
);
```



Photos Table

The Photos table manages accommodation images, featuring attributes like PhotoID and PhotoLink. It empowers hosts to display property visuals, helping guests evaluate and choose accommodations effectively.

```
CREATE TABLE photos (
  photoid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  photolink VARCHAR(500)
);
```



Rules Table

The Rules table stores accommodation rules, identified by a unique RuleID and described by a RuleDescription. It helps hosts communicate expectations and ensures a pleasant stay for guests by outlining policies and regulations.

```
CREATE TABLE rules (
  ruleid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  ruledescription TEXT(0)
);
```

Booking And Financial Group

Bookings Table

The Bookings table stores key details of guest reservations, including BookingID, GuestID, AccomID, inDate, outDate, TotalPrice, and Status. It aids in reservation management, payment processing, and communication between guests and hosts. The table ensures accurate billing, availability tracking, and supports revenue analysis. It's vital for efficient booking processes, enhancing guest experience and host interaction.

```
● CREATE TABLE Bookings (
  `BookingID`      INT NOT NULL AUTO_INCREMENT ,
  `GuestID`        INT NOT NULL ,
  `AccomID`        INT NOT NULL ,
  `inDate`         DATETIME NOT NULL ,
  `outDate`        DATETIME NOT NULL ,
  `TotalPrice`     DECIMAL NOT NULL ,
  `Status`         CHAR(1) NOT NULL ,
  CONSTRAINT `pk Bookings` PRIMARY KEY ( `BookingID`, `GuestID`, `AccomID` )
);
```

Payments Table

The Payments table holds vital payment details for bookings, including PayID, BookingID, PayMethod, Amount, PayDate, and PayStatus. It ensures secure transactions, tracks payments, and manages refunds. This table helps generate receipts, supports financial reporting, and ensures reliable reimbursements, enhancing Airbnb's payment system for both guests and hosts.

```
● CREATE TABLE Payments (
  `PayID`          INT NOT NULL AUTO_INCREMENT ,
  `BookingID`      INT NOT NULL ,
  `Amount`         DECIMAL ,
  `PayDate`        DATETIME NOT NULL ,
  `PayMethod`      VARCHAR(50) ,
  `PayStatus`      BOOLEAN ,
  CONSTRAINT `pk Payments` PRIMARY KEY ( `PayID`, `BookingID` )
);
```

Booking Extensions Table

The Booking Extensions table records extension requests for bookings, featuring ExtensionID, BookingID, ExtensionDate, and NewOutDate. It aids guests in requesting extensions and assists hosts in managing prolonged stays, enabling clear communication. This table works with Payments and Cancellations tables to streamline payments and handle booking changes, contributing to an improved booking process.

```
● CREATE TABLE bookingextensions (
  extensionid      INT NOT NULL AUTO_INCREMENT ,
  bookingid        INT NOT NULL ,
  newoutdate       DATETIME NOT NULL ,
  extensiondate    DATETIME NOT NULL ,
  CONSTRAINT `pk booking extensions` PRIMARY KEY ( extensionid, bookingid )
);
```


Cancellations Table

The Cancellations table manages booking cancellations, recording CancellationID, BookingID, CancellationDate, and CancellationReason. It aids in analyzing trends and managing refunds, promoting transparency and responsibility in cancellations.

```
CREATE TABLE cancellations (
  cancellationid INT NOT NULL AUTO_INCREMENT ,
  bookingid INT NOT NULL ,
  cancellationdate DATETIME NOT NULL ,
  cancellationreason VARCHAR(500) NOT NULL ,
  CONSTRAINT `pk cancellations` PRIMARY KEY ( cancellationid, bookingid )
);
```

Commissions Table

The Commissions table holds data about earned commissions, featuring CommissionID, BookingID, GuestCommission, HostCommission, and TotalCommission. This enables precise calculations, financial reporting, and payouts. It's essential for financial management, revenue tracking, and commission processing within the platform.

```
CREATE TABLE commissions (
  commisionid INT NOT NULL AUTO_INCREMENT ,
  bookingid INT NOT NULL ,
  guestcommision DECIMAL NOT NULL ,
  hostcommision DECIMAL NOT NULL ,
  totalcommision DECIMAL NOT NULL ,
  CONSTRAINT `pk commissions` PRIMARY KEY ( commisionid, bookingid )
);
```

Social Group

SocialMedia Table

The SocialMedia table stores user-linked social media details, with SocialMediaID, UserID, SocialMediaType, and ProfileLink. It encourages social engagement by allowing users to display their social presence on the platform.

```
CREATE TABLE SocialMedia (
  `SocialMediaID` INT NOT NULL AUTO_INCREMENT ,
  `UserID` INT NOT NULL ,
  `SocialMediaType` VARCHAR(500) ,
  `ProfileLink` VARCHAR(500) ,
  CONSTRAINT `pk Social Media` PRIMARY KEY ( `SocialMediaID`, `UserID` )
);
```

Messaging Table

The Messaging table logs user conversations with MessageID, SenderID, RecipientID, MessageContent, and MessageDate. It supports real-time communication, private chats, and preserves message history for improved user interactions.

```
CREATE TABLE Messaging (
  `MessageID` INT NOT NULL AUTO_INCREMENT ,
  `SenderID` INT NOT NULL ,
  `ReceiverID` INT NOT NULL ,
  `MessageContent` TEXT(0) NOT NULL ,
  `MessageDate` DATETIME NOT NULL ,
  CONSTRAINT `pk Messaging` PRIMARY KEY ( `MessageID`, `SenderID`, `ReceiverID` )
);
```

Notifications Table

The Notifications table holds user notifications with NotificationID, UserID, NotificationMessage, and NotificationDate. It delivers updates, supports user preferences, and improves communication and engagement.

```
CREATE TABLE notifications (
  notificationid INT NOT NULL AUTO_INCREMENT ,
  userid INT NOT NULL ,
  notificationmessage TEXT(0) NOT NULL ,
  notificationdate DATETIME NOT NULL ,
  CONSTRAINT `pk notifications` PRIMARY KEY ( notificationid, userid )
);
```

Reviews, Ratings and Favorites Group

Reviews Table

The Reviews table logs accommodation reviews with ReviewID, AccomID, GuestID, Rating, and ReviewDate. It aids decisions, ratings, and satisfaction analysis, contributing to better user experiences and informed bookings.

UserRatings Table

The UserRatings table records user ratings with UserRatingID, RatedUserID, RatingUserID, RatingValue, and RatingType. It supports user feedback, building trust in the community by allowing evaluations of interactions.

Ratings Table

The Ratings table stores ratings with RatingID, AccomID, AccomRating, and HostID. It enables detailed evaluations, averages ratings, and provides personalized recommendations based on specific criteria.

ReviewsComments Table

The ReviewComments table holds comment data with ReviewCommentID, CommentText, ReviewID, and CommentDate.

Favorites Table

The Favorites table notes user-favorite accommodations with FavouriteID, GuestID, AccomID, and DateAdded. It helps users save and compare preferred options for convenient future bookings.

```
CREATE TABLE reviews (  
    reviewid INT NOT NULL AUTO_INCREMENT,  
    guestid INT NOT NULL,  
    accomid INT NOT NULL,  
    rating DECIMAL(3, 1) NOT NULL,  
    reviewdate DATETIME NOT NULL,  
    CONSTRAINT `pk reviews` PRIMARY KEY (reviewid, guestid, accomid)  
);
```

```
CREATE TABLE UserRatings (  
    `UserRatingID` INT NOT NULL AUTO_INCREMENT ,  
    `RatedUserID` INT NOT NULL ,  
    `RatingType` VARCHAR(5) NOT NULL ,  
    `RatingValue` DECIMAL(3,1) NOT NULL ,  
    `RatingUserID` INT NOT NULL ,  
    CONSTRAINT `pk User Ratings` PRIMARY KEY ( `UserRatingID`, `RatingUserID` )  
);
```

```
CREATE TABLE Ratings (  
    `RatingID` INT NOT NULL AUTO_INCREMENT ,  
    `AccomID` INT NOT NULL ,  
    `HostID` INT NOT NULL ,  
    `AccomRating` DECIMAL(4,2) NOT NULL ,  
    CONSTRAINT `pk Ratings` PRIMARY KEY ( `RatingID`, `AccomID`, `HostID` ),  
    CONSTRAINT `fk Ratings Accomodations` FOREIGN KEY ( `AccomID` ) REFERENCES accom( accomid ) ON DELETE NO ACTION ON UPDATE NO ACTION,  
    CONSTRAINT `fk Ratings Hosts` FOREIGN KEY ( `HostID` ) REFERENCES hosts( hostid ) ON DELETE NO ACTION ON UPDATE NO ACTION  
);
```

```
CREATE TABLE ReviewsComments (  
    `ReviewCommentID` INT NOT NULL AUTO_INCREMENT ,  
    `ReviewID` INT NOT NULL ,  
    `CommentText` TEXT(0) NOT NULL ,  
    `CommentDate` DATETIME NOT NULL ,  
    CONSTRAINT `pk Reviews Comments` PRIMARY KEY ( `ReviewCommentID`, `ReviewID` )  
);
```

```
CREATE TABLE Favorites (  
    `FavoritesID` INT NOT NULL AUTO_INCREMENT ,  
    `GuestID` INT NOT NULL ,  
    `AccomID` INT NOT NULL ,  
    `DateAdded` DATETIME NOT NULL ,  
    CONSTRAINT `pk Favourites` PRIMARY KEY ( `FavoritesID`, `GuestID`, `AccomID` )  
);
```




03

Populating Mock Data

Using Dbeaver, mySQL
and Insert.sql



Inserted Data

To showcase the functionality of our Booking and Accommodation System's database, I've populated various tables with mock data, ensuring that each table contains 20 or more entries. Let's take a look at some examples:

User Data and Profiles:

- **Users Table:** User information such as names, email addresses, passwords, profile pictures, contact numbers, genders, nationalities, and ages.
- **Guests and Hosts Tables:** Guest and host data, including join dates, verification status, and user IDs.
- **Languages and SocialMedia Tables:** User preferred languages and social media profiles.

```
● INSERT INTO users (name, email, password, profilepicture, phonenumber, gender, nationality, age)
VALUES
('John Doe', 'john@example.com', 'password123', 'https://example.com/profile.jpg', '123-456-7890', 'M', 'USA', 30),
('Jane Smith', 'jane@example.com', 'securepwd456', 'https://example.com/jane.jpg', '987-654-3210', 'F', 'Canada', 28),
● INSERT INTO guests (userid, joindate)
VALUES
(1, '2023-08-01'),
(2, '2023-08-02'),
● INSERT INTO hosts (aboutme, verificationstatus, joindate, userid)
VALUES
('Experienced host offering great accommodations.', 1, '2023-08-01', 4),
('Friendly host excited to meet new guests!', 0, '2023-08-02', 5),
● INSERT INTO Languages (LanguageName, UserID)
VALUES
('English', 1),
('French', 2),
● INSERT INTO SocialMedia (UserID, SocialMediaType, ProfileLink)
VALUES
(1, 'Twitter', 'https://twitter.com/user1'),
(2, 'Instagram', 'https://www.instagram.com/user2'),
```

User Preferences and Connections:

- **UserRatings Table:** User ratings.
- **Favorites, Verifications, and UserAccommodation Tables:** User favorites, verification statuses, and user-accommodation associations.

```
● INSERT INTO UserRatings (RatedUserID, RatingType, RatingValue, RatingUserID)
VALUES
(1, 'Host', 4.8, 2),
(2, 'Guest', 4.5, 3),
● INSERT INTO Favorites (GuestID, AccomID, DateAdded)
VALUES
(1, 1, '2023-08-05'),
(2, 2, '2023-08-06'),
● INSERT INTO useraccommodation (accomid, userid)
VALUES
(1, 1),
(2, 2),
● INSERT INTO Verifications (VerificationStatus, HostID)
VALUES
(TRUE, 1),
```



Accommodation Details:

- Accommodations Table: Accommodation details like titles, descriptions, creation dates, and corresponding host IDs.
- Facilities and Tags Tables: Facility and tag names associated with accommodations.
- AccomFacilities and AccomTag Tables: Accommodations linked to facilities and tags.
- Amenities and AccomAmenity Tables: Amenity names linked to accommodations.
- AccomCountry and AccomCity Tables: Accommodations linked to countries and cities.
- Rules and AccomRule Tables: Rules linked to accommodations.
- SpecialOffers Table: Special offers associated with accommodations, including offer titles, descriptions, discounts, expiration dates, and host IDs.
- Photos and AccomPhotos Tables: Photo links associated with accommodations.

Location Data:

- Countries and Cities Tables: Data entries representing different countries and cities, each with their respective codes and population figures.
- Locations Table: Accommodation locations with longitude, latitude, and addresses.

```
●INSERT INTO accom (hostid, title, description, createdate)
VALUES
(1, 'Cozy Downtown Apartment', 'A comfortable apartment in the heart of the city.', '2023-08-01'),
(2, 'Seaside Villa with Stunning Views', 'Relax and enjoy the beachfront in this beautiful villa.', '2023-08-02'),
```

```
●INSERT INTO facilities (facilityname) ●INSERT INTO accomfacilities (accomid, facilityid)
VALUES VALUES
('Gym'), (1, 1),
('Parking Garage'), (1, 2),
```

```
●INSERT INTO Tags (tagname) ●INSERT INTO accomtag (accomid, tagid)
VALUES VALUES
('Family-Friendly'), (1, 1),
('Luxury'), (2, 2),
```

```
●INSERT INTO Amenities (AmenityName) ●INSERT INTO AccomAmenity (AccomID, AmenityID)
VALUES VALUES
('Gym'), (1, 1),
('Spa'), (2, 2),
```

```
●INSERT INTO rules (ruledescription) ●INSERT INTO AccomRule (AccomID, RuleID)
VALUES VALUES
('No smoking allowed.'), (1, 1),
('Pets are welcome.'), (2, 2),
```

```
●INSERT INTO specialoffers (accomid, offertitle, offerdescription, offerdiscount, offerexpirationdate, hostid)
VALUES
(1, 'Summer Getaway', 'Enjoy 20% off for bookings in August.', 20, '2023-08-31', 1),
(2, 'Weekend Retreat', 'Book 3 nights, get the 4th night free!', 25, '2023-09-15', 2),
```

```
●INSERT INTO photos (photolink) ●INSERT INTO accomphotos (photoid, accomid)
VALUES VALUES
('https://example.com/photo1.jpg'), (1, 1),
('https://example.com/photo2.jpg'), (2, 2),
```

```
●INSERT INTO locations (locationname, longitude, latitude, address)●INSERT INTO AccomLocation (AccomLocationID, AccomID, LocationID)
VALUES VALUES
('Downtown Area', -73.9857, 40.7488, '123 Main St, City'), (1, 1, 1),
('Beachfront', -118.2437, 34.0522, '456 Ocean Ave, Coast'), (2, 2, 2),
```

```
●INSERT INTO Cities (CityName, CountryID, Population) ●INSERT INTO accomcity (accomid, cityid)
VALUES VALUES
('New York', 1, 8500000), (1, 1),
('Los Angeles', 1, 3900000), (2, 2),
```

```
●INSERT INTO Countries (CountryName, Code) ●INSERT INTO AccomCountry (AccomID, CountryID)
VALUES VALUES
('United States', 'USA'), (1, 1),
('Canada', 'CAN'), (2, 2),
```

Review and Feedback Data:

- Reviews Table: Guest reviews for accommodations including accommodation IDs, guest IDs, review dates, and ratings.

Booking and Transaction Data:

- Bookings Table: Guest bookings with guest IDs, accommodation IDs, check-in and check-out dates, total prices, and booking statuses.
- BookingExtensions, Cancellations, and Commissions Tables: Data related to booking extensions, cancellations, and commissions.
- Payments Table: Payment details including booking IDs, payment amounts, dates, payment methods, and payment statuses.

Communication and Interaction:

- Notifications and Messaging Tables: Notification and messaging data including user IDs, message content, and dates.
- ReviewsComments and Ratings Tables: Guest reviews and ratings with comments, dates, and ratings.

```
● INSERT INTO reviews (accomid, guestid, reviewdate, rating)
VALUES
(1, 1, '2023-08-12', 5),
(2, 2, '2023-08-13', 4),
```

```
● INSERT INTO Bookings (GuestID, AccomID, inDate, outDate, TotalPrice, Status)
VALUES
(1, 1, '2023-08-10', '2023-08-15', 500.00, '1'),
(2, 2, '2023-09-01', '2023-09-05', 750.00, '0').

● INSERT INTO cancellations (bookingid, cancellationdate, cancellationreason)
VALUES
(1, '2023-08-08', 'Change of plans'),
(2, '2023-09-03', 'Emergency situation'),

● INSERT INTO bookingextensions (bookingid, newoutdate, extensiondate)
VALUES
(1, '2023-08-20', '2023-08-15'),
(2, '2023-09-08', '2023-09-05'),

● INSERT INTO commisions (bookingid, guestcommision, hostcommision, totalcommision)
VALUES
(1, 25.00, 50.00, 75.00),
(2, 30.00, 45.00, 75.00),

● INSERT INTO Payments (BookingID, Amount, PayDate, PayMethod, PayStatus)
VALUES
(1, 200.00, '2023-08-10', 'Credit Card', 1),
(2, 300.00, '2023-08-12', 'PayPal', 1),

● INSERT INTO HostBank (HostID, AccountNumber, BankName, RoutingNumber)
VALUES
(1, 1234567890, 'ABC Bank', '123456789'),
(2, 9876543210, 'XYZ Bank', '987654321'),
```

```
● INSERT INTO notifications (userid, notificationmessage, notificationdate)
VALUES
(1, 'New booking request received.', '2023-08-05 10:30:00'),
(2, 'Your special offer was accepted!', '2023-08-06 14:15:00'),

● INSERT INTO Messaging (SenderID, ReceiverID, MessageContent, MessageDate)
VALUES
(1, 2, 'Hello there!', '2023-08-05 08:30:00'),
(2, 1, 'Hi! How are you?', '2023-08-06 10:15:00'),

● INSERT INTO ReviewsComments (ReviewID, CommentText, CommentDate)
VALUES
(1, 'Great experience overall!', '2023-08-12 15:20:00'),
(2, 'Could be cleaner.', '2023-08-13 09:45:00'),

● INSERT INTO Ratings (AccomID, HostID, AccomRating)
VALUES
(1, 1, 4.5),
(2, 2, 3.8),
```





04

Testing and Results

Using Dbeaver, MySQL
and Testing.sql



Testing Queries and Results

Simple Data Retrieval:

```
-- Retrieve the names and email addresses of all users.
SELECT name, email
FROM users;
```

	ABC name	ABC email
1	John Doe	john@example.com
2	Jane Smith	jane@example.com
3	Michael Johnson	michael@example.com
4	Alice Brown	alice@example.com
5	Robert Lee	robert@example.com

Basic Filtering:

```
-- Get the titles and descriptions of accommodations where the host's verification status is true.
SELECT a.title, a.description
FROM accom a
JOIN hosts h ON a.hostid = h.hostid
WHERE h.verificationstatus = 1;
```

	ABC title	ABC description
1	Cozy Downtown Apartment	A comfortable apartment in the heart of the city.
2	Secluded Forest Retreat	Disconnect from the world in this hidden forest re
3	Modern Loft with Skyline View	Enjoy breathtaking city views from this contempo
4	Mountain Cabin Retreat	Escape to the mountains and unwind in this cozy

Sorting and Limiting:

```
-- List the names of guests, sorted alphabetically, along with their join dates, limited to the first 5 entries.
SELECT u.name AS guest_name, g.joindate
FROM guests g
JOIN users u ON g.userid = u.userid
ORDER BY guest_name ASC
LIMIT 5;
```

	ABC guest_name	🕒 joindate
1	Alice Brown	2023-08-04 00:00:00
2	Ava Johnson	2023-08-27 00:00:00
3	Ava Mitchell	2023-08-19 00:00:00
4	Daniel Brown	2023-08-22 00:00:00
5	Daniel Johnson	2023-08-07 00:00:00

Aggregation with Grouping:

```
-- Calculate the average rating of accommodations, grouping them by city.
SELECT c.CityName, AVG(rating) AS average_rating
FROM accom a
JOIN accomcity ac ON a.accomid = ac.accomid
JOIN Cities c ON ac.cityid = c.cityid
LEFT JOIN reviews r ON a.accomid = r.accomid
GROUP BY c.CityName
ORDER BY average_rating DESC;
```

	ABC CityName	📊 average_rating
1	Vancouver	5
2	Melbourne	5
3	Marseille	5
4	London	4,5
5	Sydney	4,5
6	New York	4,33333
7	Toronto	4,33333
8	Los Angeles	4
9	Manchester	4
10	Paris	3



Testing Queries and Results

Aggregation with Filtering:

```
-- Find the total number of accommodations hosted by each host who has more than one accommodation.
SELECT h.userid, COUNT(a.accomid) AS total_accommodations
FROM hosts h
JOIN accom a ON h.userid = a.hostid
GROUP BY h.userid
HAVING total_accommodations > 1;
```

	userid	total_accommodations
1	4	3
2	5	3
3	6	2
4	7	2
5	8	2

Joins for Detailed Information:

```
-- Retrieve the names of guests who have booked accommodations in New York City, along with the accommodation titles and booking dates.
SELECT u.name AS guest_name, a.title AS accommodation_title, b.indate AS booking_date
FROM users u
JOIN guests g ON u.userid = g.userid
JOIN Bookings b ON g.userid = b.guestid
JOIN accom a ON b.accomid = a.accomid
JOIN accomcity ac ON a.accomid = ac.accomid
JOIN Cities c ON ac.cityid = c.cityid
WHERE c.cityname = 'New York'
ORDER BY guest_name, booking_date;
```

	guest_name	accommodation_title	booking_date
1	Grace Lee	Cozy Downtown Apartment	2023-08-22 00:00:00
2	John Doe	Cozy Downtown Apartment	2023-08-10 00:00:00
3	William Taylor	Cozy Downtown Apartment	2023-12-10 00:00:00

Nested Subquery:

```
-- List the accommodations with a rating greater than the average rating of all accommodations.
SELECT a.title, a.description, r.accomrating
FROM accom a
JOIN Ratings r ON a.accomid = r.accomid
WHERE r.accomrating > (SELECT AVG(accomrating) FROM Ratings);
```

	title	description	accomrating
1	Cozy Downtown Apartment	A comfortable apartment in the heart of the city.	4,5
2	Urban Loft with Modern Desi	Experience city living in this stylish loft.	4,7
3	Luxurious Resort by the Oce	Indulge in luxury at this oceanfront resort.	4,9
4	Rustic Chalet in the Woods	Embrace nature in this charming chalet surround	4,3

Data Modification:

```
-- Update the price of all bookings with a total price higher than $500 to increase by 10%.
UPDATE Bookings
SET totalprice = totalprice * 1.1
WHERE totalprice > 500;
```

	BookingID	GuestID	AccomID	inDate	outDate	TotalPrice	Status
1	1	1	1	2023-08-10 00:00:00	2023-08-15 00:00:00	500	1
2	2	2	2	2023-09-01 00:00:00	2023-09-05 00:00:00	1 099	0
1	1	1	1	2023-08-10 00:00:00	2023-08-15 00:00:00	500	1
2	2	2	2	2023-09-01 00:00:00	2023-09-05 00:00:00	1 209	0



Testing Queries and Results

Correlated Subquery:

```
-- Display the accommodations and their titles that have been favorited by users who are also hosts.
SELECT a.accomid, a.title
FROM accom a
INNER JOIN Favorites f ON a.accomid = f.accomid
INNER JOIN guests g ON f.guestid = g.userid
INNER JOIN hosts h ON g.userid = h.hostid;
```

	123 accomid	ABC title
1		1 Cozy Downtown Apartment
2		2 Seaside Villa with Stunning V
3		3 Mountain Cabin Retreat

Complex Join and Aggregation:

```
-- Find the average commission earned by hosts from each country, considering both guest and host commission.
SELECT c.CountryName, AVG(cr.totalcommission) AS avg_commission
FROM commissions cr
JOIN Bookings b ON cr.bookingid = b.bookingid
JOIN accom a ON b.accomid = a.accomid
JOIN AccomCountry ac ON a.accomid = ac.accomid
JOIN Countries c ON ac.countryid = c.countryid
GROUP BY c.CountryName;
```

	ABC CountryName	123 avg_commission
1	United States	67,1071
2	Australia	69
3	Canada	66,8889
4	United Kingdom	65,5

Aggregation with Grouping and Date Functions:

```
-- Find the total earnings earned by the company from commissions, grouped by month.
SELECT YEAR(b.inDate) AS year, MONTH(b.inDate) AS month, SUM(c.totalcommission) AS total_earnings
FROM Bookings b
JOIN commissions c ON b.bookingid = c.bookingid
GROUP BY YEAR(b.inDate), MONTH(b.inDate)
ORDER BY year, month;
```

	123 year	123 month	123 total_earnings
1	2 023	8	385
2	2 023	9	517
3	2 023	10	462
4	2 023	11	480
5	2 023	12	580



THANKS

