Course Name: Object Oriented and Functional Programming with Python
Course Code: DLBDSOOFPP01
Student Name: Jedidiah Prince Amos
Matriculation No: 32009079

Portfolio
Build a Data Mart in SQL: Concept Phase

Abstract

This project revolves around the development of a database for managing Airbnb operations. The introduction provides context, defining Airbnb as an online marketplace connecting hosts offering accommodations with global travelers. The project's aim is outlined, emphasizing the systematic creation of a data mart through sections dedicated to tools, requirements specification, and entity relationship modeling (ERM).

The tools selected include Lucidchart for ERM creation and MySQL for its flexibility in modeling complex relationships. The requirements specification delves into user roles, data, functions, and calculations. User roles like Guest, Host, Admin, and Customer Support are defined with their associated permissions. Data requirements encompass entities, relationships, integrity constraints, and the use of foreign keys. Functional requirements outline user operations and roles, alongside calculations and automation considerations.

The ERM development in Section 4 visualizes relationships and attributes, with a focus on Property, Amenities, Property Comment, and Review entities. The data dictionary details each table functionality and relations.

In conclusion, the project outlines a structured plan for developing the Airbnb database. The identified tools, Lucidchart, and MySQL, are crucial for efficient development. The roles and functions of users are clearly defined, contributing to a robust database design. The ERM and data dictionary solidify the understanding of relationships and attribute types, ensuring accuracy and consistency.

1. Introduction

1.1. Airbnb Background Knowledge

According to Jean (2023), airbnb is an online marketplace that connects people who want to rent out their property with people who are looking for accommodations, typically for short stays. Airbnb connects local hosts offering spare rooms or homes with global travelers, providing a convenient platform for rentals with secure payment handling and additional support.

1.2. Aim

The aim of this project is to build a database for storing and processing information regarding the Airbnb use case. Creating a data mart is a systematic process, starting from identifying business needs to implementation and testing. This project is split into sections to systematically create the database. Section 2 identifies the tools and resources needed to begin development of the database. Section 3 determine the requirement specifications for each of the user roles as well as the required data and functions for the database. Section 4 focuses on the development of the entity relationship model (ERM), describing the relationship between entities and their attributes as well as providing a data dictionary.

2. Tools & Resources

- **Lucidchart:** Lucidchart is a cloud-based diagramming and visual communication tool that allows users to create various types of diagrams, including flowcharts, mind maps, organizational charts, and ERM. Lucidchart is chosen to create the ERM due to its ease of access and use, as well as the ability to convert the ERM directly into SQL code that greatly simplifies workload.

- **MySQL:** MySQL is chosen for modelling Airbnb database due to its versatility, scalability, and reliability. As a relational database management system (RDBMS), MySQL supports the structured organization of data, making it well-suited for modeling complex relationships within Airbnb's data, such as user profiles, property listings, and booking transactions. Its robust performance and open-source nature make it align with business considerations. Additionally, MySQL integrates seamlessly with various programming languages and frameworks, providing flexibility.

3. Requirements Specification

3.1. Roles & Performance

Guest	Host	Admin
Book Accommodations	Create and manage property listings	Manage user accounts and permissions
Communicate expectations	Preparation of Airbnb (engineering/ electrician / water certified)	Different level of access

Provide information (Address, Credit card, picture, etc.)	House maintenance	Monitor and maintain the platform/ database
Abide by host rules	Communicate with guests	
Pays host	Set pricing	
Provide feedback and review		

3.2. Data & Functions

Data Requirements:

1. *Entities and Attributes:* Identify the entities that need to be represented in the database and define their respective attributes. For instance, an entity would be guests where it entails the attributes of AddressID, ProfilePicture, UserID etc.

2. *Relationships:* It is important to determine how entities are related to each other and their relationship cardinality.

3. *Data Integrity Constraints:* Specify any constraints on the data to ensure accuracy and consistency. This could include rules like unique constraints (e.g., each user has a unique email) or foreign key constraints.

4. *Use of Foreign Keys:* Helps maintain data consistency and makes database easier to navigate.

5. *Referential Integrity:* Relating to the use of a table's primary key in other tables as a foreign key. Must ensure that changes to data in one record is related to other records.

Functional Requirements:

Functions:

- A user would be able to perform the following operations on the database: booking accommodations, writing reviews, search for properties, rating host and property, writing property comments, contacting customer support and doing payments.

User roles and level of access:

- Specify the roles that users can play in the system and what actions each role is allowed to perform. For instance, the system roles are guest, host, customer support and admin. Each role is associated with specific permissions, determining what actions users in that role can or cannot perform. For example, a guest would perform the booking operations, doing payments and searching for property while a customer support role would deal with customer interaction. Having different user roles help control access and ensures that users only have the necessary privileges for their roles, contributing to security and data protection.

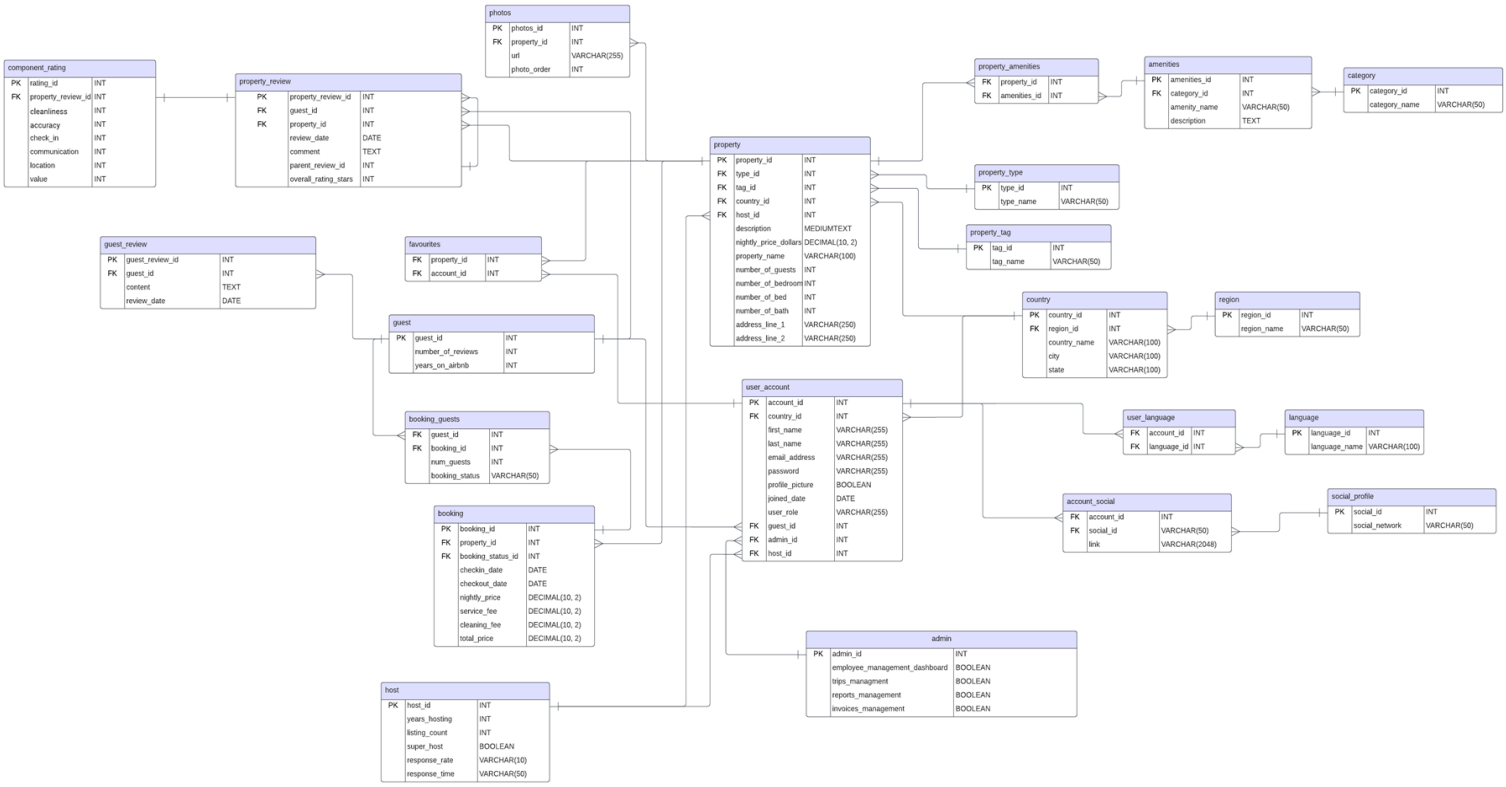
Calculations and automation:

- Determine if there are any processes within the system that need to be automated or need certain calculations. For example, automating processes for sending confirmation emails or calculating the total cost of a booking.

4. Entity Relationship Model (ERM)

The following ERM is created in martin (crowfoot) notation:

Figure 1: Airbnb ERM



4.1. Data Dictionary

Guest: Represents guests with attributes like number of reviews and years on Airbnb.

Property Tag, Region, Country: Classification and location entities for properties.

Host: Represents hosts with attributes like years hosting, listing count, and super host status.

Property Type: Classification of property types.

Property: Main entity representing properties with detailed attributes like nightly price, number of guests, bedrooms, etc. Foreign keys linking to Property Type, Property Tag, Country, and Host.

Booking: Represents booking details with attributes like check-in/out dates, prices, etc. Linked to Property.

Booking Guests: Records guests associated with bookings.

Category: Represents categories for amenities.

Admin, Language, Social Profile: Administrative, language, and social profile entities
.
Guest Review, Amenities: Guest reviews and property amenities with descriptions.

User Account: Represents user accounts with attributes like name, email, role, etc. Linked to Guest, Host, Admin, Country, and Social Profile.

Favourites, User Language, Account Social: Entities for user preferences, languages, and social profiles.

Component Rating, Property Review: Represents ratings and reviews for properties, including cleanliness, accuracy, etc. Linked to Property and Guest.

Property Amenities, Photos: Additional details like property amenities and photos.

5. Conclusion

The aim of this project is to develop an Airbnb database that would entail various tables, relationships, data types and functionality. In section 2, Lucidchart and MySQL were identified as crucial tools needed to progress with the development of the Airbnb database. In section 3, roles such as guest, host, admin and customer support were given different functional purposes. A guest role entail booking and payment operations as oppose to a host role which is supervising the property. Admin roles are database centric with different level of restriction, while a customer support focuses primarily on dealing with customer queries and complaints. An ERM is developed in section 4, showcasing all the various relationships between entities. Overall, this concept fulfills its aim of serving as a guide plan for the next phase of developing the database.

Data Mart

PORTFOLIO: DEVELOPMENT PHASE

Build a Data Mart in SQL

Credentials:

Course Name: Project: Build a Data Mart in SQL
Course Code: DLBDSPBDM01
Student Name: Jedidiah Prince Amos
Matriculation No: 32009079

CONTENT

1. Introduction

1. Airbnb Background Knowledge

- Airbnb is an online marketplace connecting hosts with guests for short-term rentals.
- It facilitates rentals of spare rooms or homes globally, offering secure payment handling and support.

2. Aim of the Project

- The aim is to develop a database for managing Airbnb-related information.
- The project follows a systematic approach, beginning with identifying business needs and ending with implementation and testing.

Database Design & Implementation Summary

- Developed a database management system for the Airbnb use case.
- Created an Entity-Relationship Model (ER Model) to guide database structure.
- Executed SQL statements for table creation and tested each table with varying SQL queries.
- Accompanying the submission is detailed documentation of all SQL commands for reference.
- Included screenshots of executed SQL queries and results in MySQL Workbench.
- Incorporated feedback for the database refinement.
- Result: Robust foundation for efficient management of Airbnb data.

2. Tools & Resources

MySQL:

- Relational Database Management System (RDBMS).
- Chosen for modeling the Airbnb database.
- Versatile, scalable, and reliable.
- Robust performance and open-source nature.
- Seamless integration with programming languages.

Lucidchart:

- Cloud-based diagramming tool.
- Used for creating the Entity-Relationship Model (ERM).
- Simplifies complex data visualization.
- Direct conversion of ERM to SQL code.

3. Requirement Specifications

Roles & Performance

Guest	Host	Admin
Book Accommodations	Create and manage property listings	Manage user accounts and permissions
Communicate expectations	Preparation of Airbnb (engineering/ electrician / water certified)	Different level of access
Provide information (Address, Credit card, picture, etc.)	House maintenance	Monitor and maintain the platform/ database
Abide by host rules	Communicate with guests	
Pays host	Set pricing	
Provide feedback and review		

Data & Functional Requirements

Data Requirements:

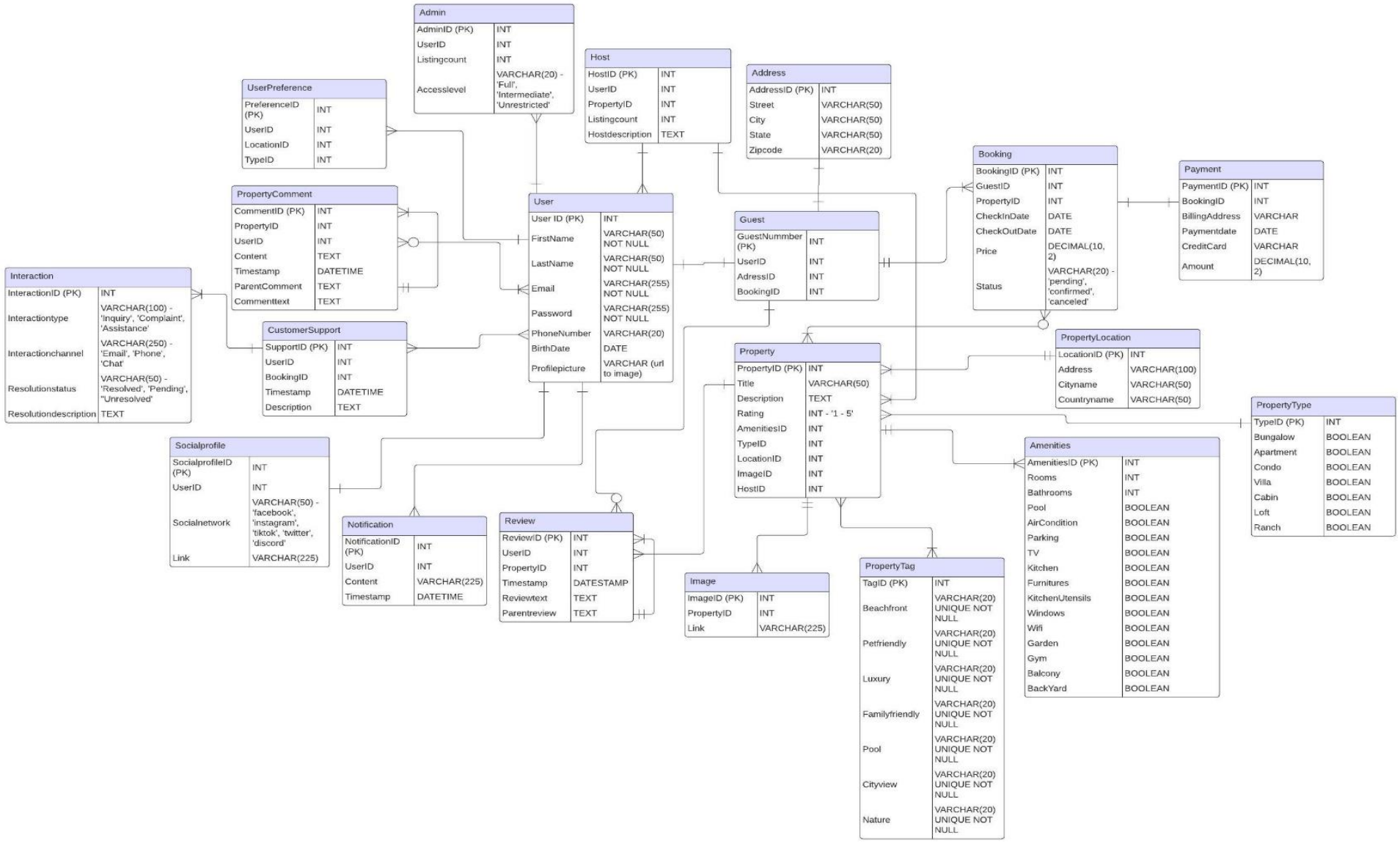
- Entities and Attributes identification.
- Relationship determination and cardinality.
- Data Integrity Constraints specification.
- Use of Foreign Keys for consistency.
- Ensuring Referential Integrity across tables.

Functional Requirements:

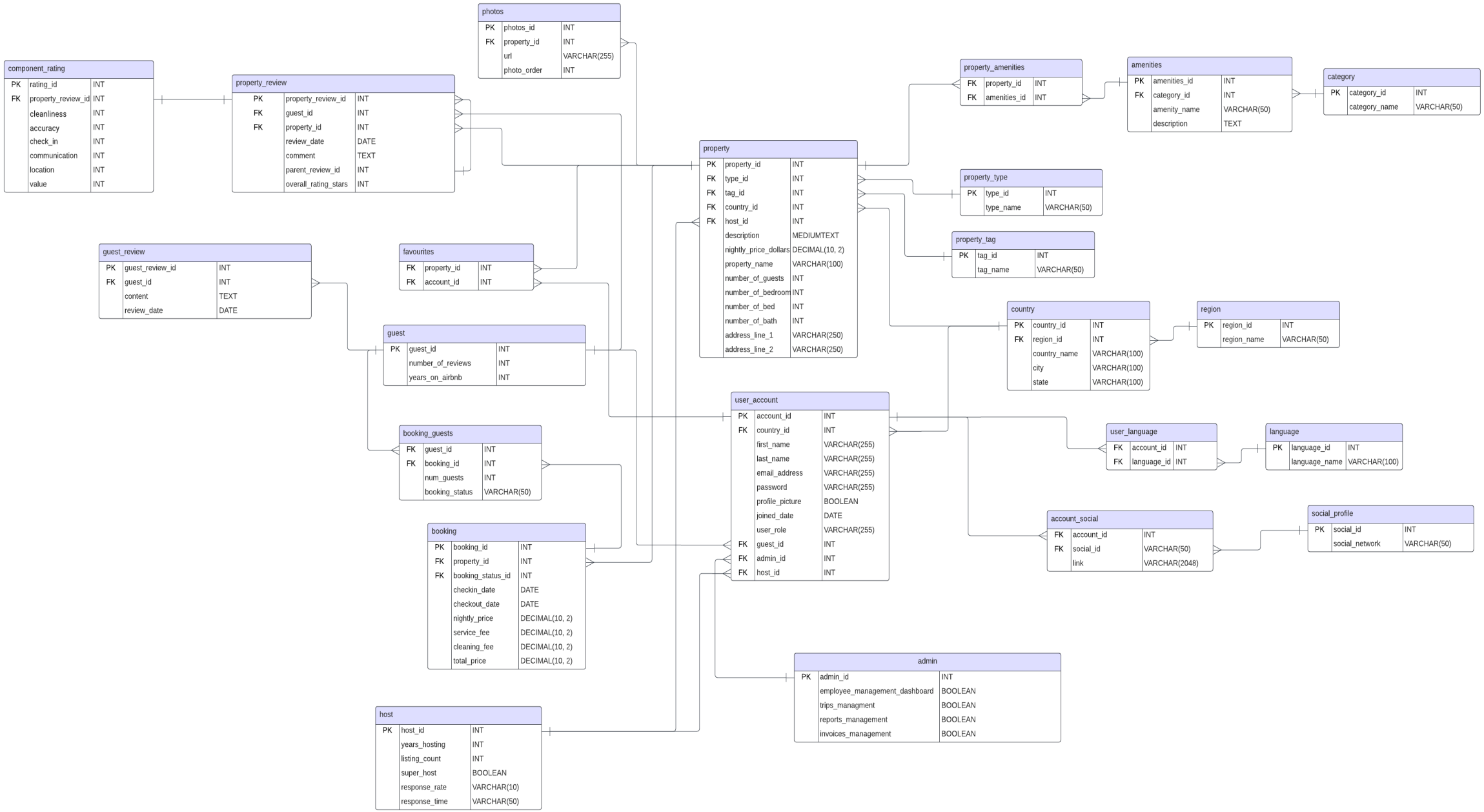
- **Functions**:
  - Booking accommodations.
  - Writing reviews.
  - Property search.
  - Host and property rating.
  - Property comments.
  - Customer support.
  - Payments.
- **User Roles & Access**:
  - Roles: Guest, Host, Customer Support, Admin.
  - Role-specific permissions for actions.
- **Calculations & Automation**:
  - Automation for confirmation emails.
  - Calculation of booking costs.

4. Entity Relationship Model

Old ERM


Admin
AdminID (PK) INT
UserID INT
Listingcount INT
Accesslevel VARCHAR(20) - 'Full', 'Intermediate', 'Unrestricted'
Host
HostID (PK) INT
UserID INT
PropertyID INT
Listingcount INT
Hostdescription TEXT
Address
AddressID (PK) INT
Street VARCHAR(50)
City VARCHAR(50)
State VARCHAR(50)
Zipcode VARCHAR(20)
UserPreference
PreferenceID (PK) INT
UserID INT
LocationID INT
TypeID INT
PropertyComment
CommentID (PK) INT
PropertyID INT
UserID INT
Content TEXT
Timestamp DATETIME
ParentComment TEXT
Commenttext TEXT
User
User ID (PK) INT
FirstName VARCHAR(50) NOT NULL
LastName VARCHAR(50) NOT NULL
Email VARCHAR(255) NOT NULL
Password VARCHAR(255) NOT NULL
PhoneNumber VARCHAR(20)
BirthDate DATE
Profilepicture VARCHAR (url to image)
Guest
GuestNummber (PK) INT
UserID INT
AdressID INT
BookingID INT
Booking
BookingID (PK) INT
GuestID INT
PropertyID INT
CheckInDate DATE
CheckOutDate DATE
Price DECIMAL(10, 2)
Status VARCHAR(20) - 'pending', 'confirmed', 'canceled'
Payment
PaymentID (PK) INT
BookingID INT
BillingAddress VARCHAR
Paymentdate DATE
CreditCard VARCHAR
Amount DECIMAL(10, 2)
Interaction
InteractionID (PK) INT
Interactiontype VARCHAR(100) - 'Inquiry', 'Complaint', 'Assistance'
Interactionchannel VARCHAR(250) - 'Email', 'Phone', 'Chat'
Resolutionstatus VARCHAR(50) - 'Resolved', 'Pending', "Unresolved'
Resolutiondescription TEXT
CustomerSupport
SupportID (PK) INT
UserID INT
BookingID INT
Timestamp DATETIME
Description TEXT
Property
PropertyID (PK) INT
Title VARCHAR(50)
Description TEXT
Rating INT - '1 - 5'
AmenitiesID INT
TypeID INT
LocationID INT
ImageID INT
HostID INT
PropertyLocation
LocationID (PK) INT
Address VARCHAR(100)
Cityname VARCHAR(50)
Countryname VARCHAR(50)
PropertyType
TypeID (PK) INT
Bungalow BOOLEAN
Apartment BOOLEAN
Condo BOOLEAN
Villa BOOLEAN
Cabin BOOLEAN
Loft BOOLEAN
Ranch BOOLEAN
Socialprofile
SocialprofileID (PK) INT
UserID INT
Socialnetwork VARCHAR(50) - 'facebook', 'instagram', 'tiktok', 'twitter', 'discord'
Link VARCHAR(225)
Notification
NotificationID (PK) INT
UserID INT
Content VARCHAR(225)
Timestamp DATETIME
Review
ReviewID (PK) INT
UserID INT
PropertyID INT
Timestamp DATESTAMP
Reviewtext TEXT
Parentreview TEXT
Image
ImageID (PK) INT
PropertyID INT
Link VARCHAR(225)
PropertyTag
TagID (PK) INT
Beachfront VARCHAR(20) UNIQUE NOT NULL
Petfriendly VARCHAR(20) UNIQUE NOT NULL
Luxury VARCHAR(20) UNIQUE NOT NULL
Familyfriendly VARCHAR(20) UNIQUE NOT NULL
Pool VARCHAR(20) UNIQUE NOT NULL
Cityview VARCHAR(20) UNIQUE NOT NULL
Nature VARCHAR(20) UNIQUE NOT NULL
Amenities
AmenitiesID (PK) INT
Rooms INT
Bathrooms INT
Pool BOOLEAN
AirCondition BOOLEAN
Parking BOOLEAN
TV BOOLEAN
Kitchen BOOLEAN
Furnitures BOOLEAN
KitchenUtensils BOOLEAN
Windows BOOLEAN
Wifi BOOLEAN
Garden BOOLEAN
Gym BOOLEAN
Balcony BOOLEAN
BackYard BOOLEAN

New ERM


photos
PK photos_id INT
FK property_id INT
url VARCHAR(255)
photo_order INT
component_rating
PK rating_id INT
FK property_review_id INT
cleanliness INT
accuracy INT
check_in INT
communication INT
location INT
value INT
property_review
PK property_review_id INT
FK guest_id INT
FK property_id INT
review_date DATE
comment TEXT
parent_review_id INT
overall_rating_stars INT
property_amenities
FK property_id INT
FK amenities_id INT
amenities
PK amenities_id INT
FK category_id INT
amenity_name VARCHAR(50)
description TEXT
category
PK category_id INT
category_name VARCHAR(50)
property
PK property_id INT
FK type_id INT
FK tag_id INT
FK country_id INT
FK host_id INT
description MEDIUMTEXT
nightly_price_dollars DECIMAL(10, 2)
property_name VARCHAR(100)
number_of_guests INT
number_of_bedroom INT
number_of_bed INT
number_of_bath INT
address_line_1 VARCHAR(250)
address_line_2 VARCHAR(250)
property_type
PK type_id INT
type_name VARCHAR(50)
property_tag
PK tag_id INT
tag_name VARCHAR(50)
guest_review
PK guest_review_id INT
FK guest_id INT
content TEXT
review_date DATE
favourites
FK property_id INT
FK account_id INT
guest
PK guest_id INT
number_of_reviews INT
years_on_airbnb INT
country
PK country_id INT
FK region_id INT
country_name VARCHAR(100)
city VARCHAR(100)
state VARCHAR(100)
region
PK region_id INT
region_name VARCHAR(50)
user_account
PK account_id INT
FK country_id INT
first_name VARCHAR(255)
last_name VARCHAR(255)
email_address VARCHAR(255)
password VARCHAR(255)
profile_picture BOOLEAN
joined_date DATE
user_role VARCHAR(255)
FK guest_id INT
FK admin_id INT
FK host_id INT
booking_guests
FK guest_id INT
FK booking_id INT
num_guests INT
booking_status VARCHAR(50)
user_language
FK account_id INT
FK language_id INT
language
PK language_id INT
language_name VARCHAR(100)
account_social
FK account_id INT
FK social_id VARCHAR(50)
link VARCHAR(2048)
social_profile
PK social_id INT
social_network VARCHAR(50)
booking
PK booking_id INT
FK property_id INT
FK booking_status_id INT
checkin_date DATE
checkout_date DATE
nightly_price DECIMAL(10, 2)
service_fee DECIMAL(10, 2)
cleaning_fee DECIMAL(10, 2)
total_price DECIMAL(10, 2)
admin
PK admin_id INT
employee_management_dashboard BOOLEAN
trips_managment BOOLEAN
reports_management BOOLEAN
invoices_management BOOLEAN
host
PK host_id INT
years_hosting INT
listing_count INT
super_host BOOLEAN
response_rate VARCHAR(10)
response_time VARCHAR(50)

Data dictionary

Guest:

- Represents guests with attributes like number of reviews and years on Airbnb.

Property Tag, Region, Country:

- Classification and location entities for properties.

Host:

- Represents hosts with attributes like years hosting, listing count, and super host status.

Property Type:

- Classification of property types.

Property:

- Main entity representing properties with detailed attributes like nightly price, number of guests, bedrooms, etc.
- Foreign keys linking to Property Type, Property Tag, Country, and Host.

Booking:

- Represents booking details with attributes like check-in/out dates, prices, etc.
- Linked to Property.

Booking Guests:

- Records guests associated with bookings.

Category:

- Represents categories for amenities.

Admin, Language, Social Profile:

- Administrative, language, and social profile entities

.

Guest Review, Amenities:

- Guest reviews and property amenities with descriptions.

User Account:

- Represents user accounts with attributes like name, email, role, etc.
- Linked to Guest, Host, Admin, Country, and Social Profile.

Favourites, User Language, Account Social:

- Entities for user preferences, languages, and social profiles.

Component Rating, Property Review:

- Represents ratings and reviews for properties, including cleanliness, accuracy, etc.
- Linked to Property and Guest.

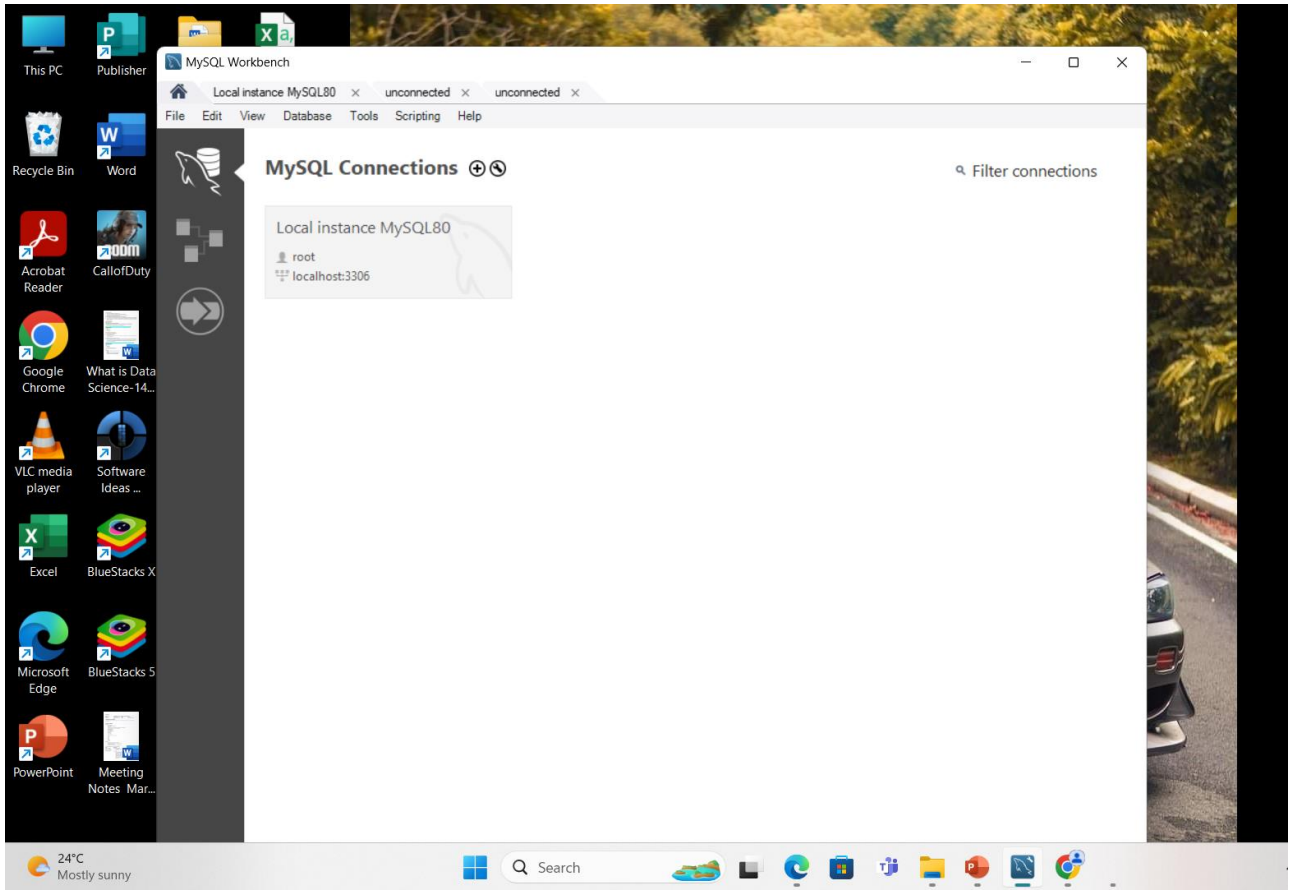
Property Amenities, Photos:

- Additional details like property amenities and photos.

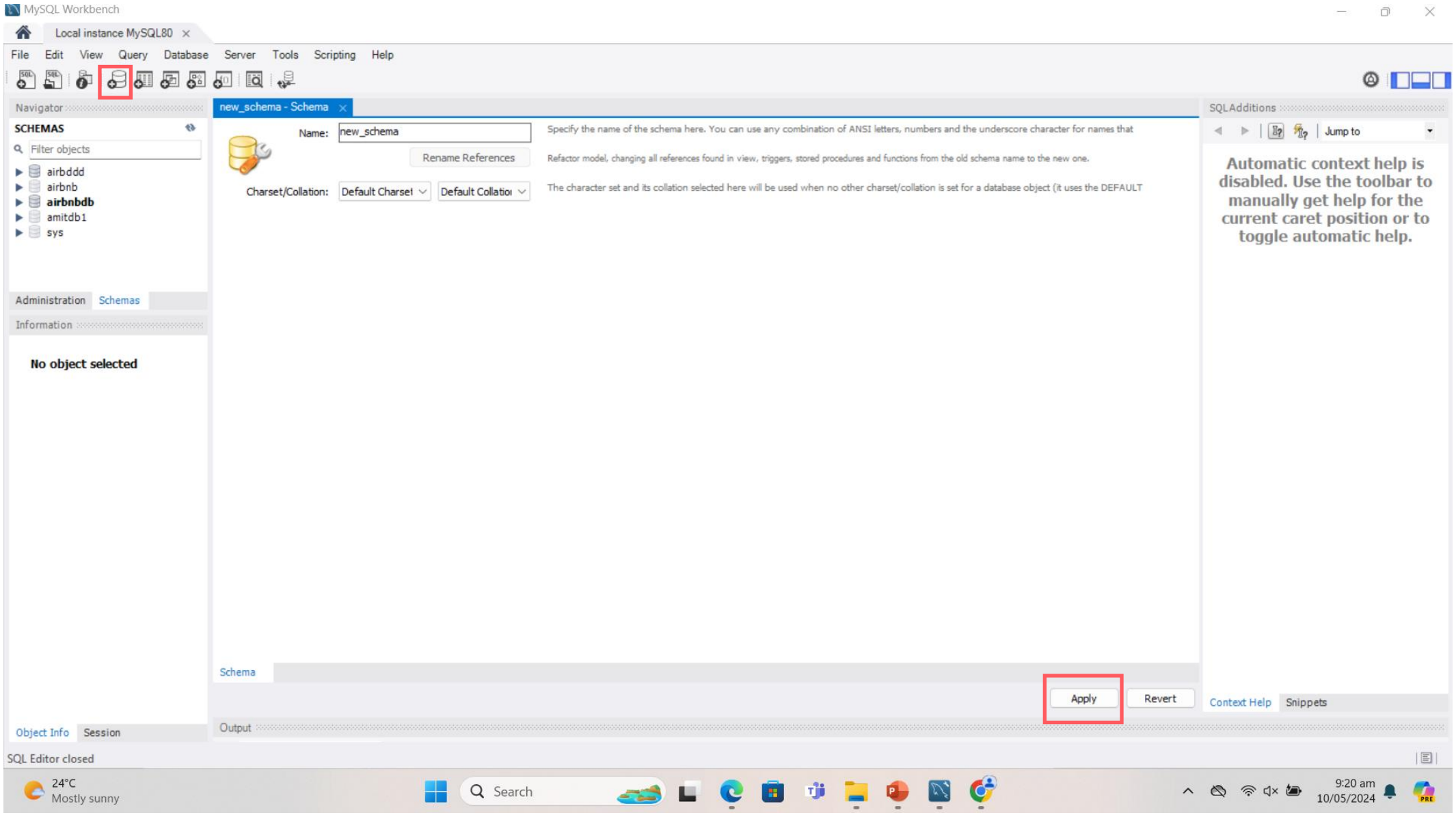
5. Importing the database to MySQL

1. Install MySQL.

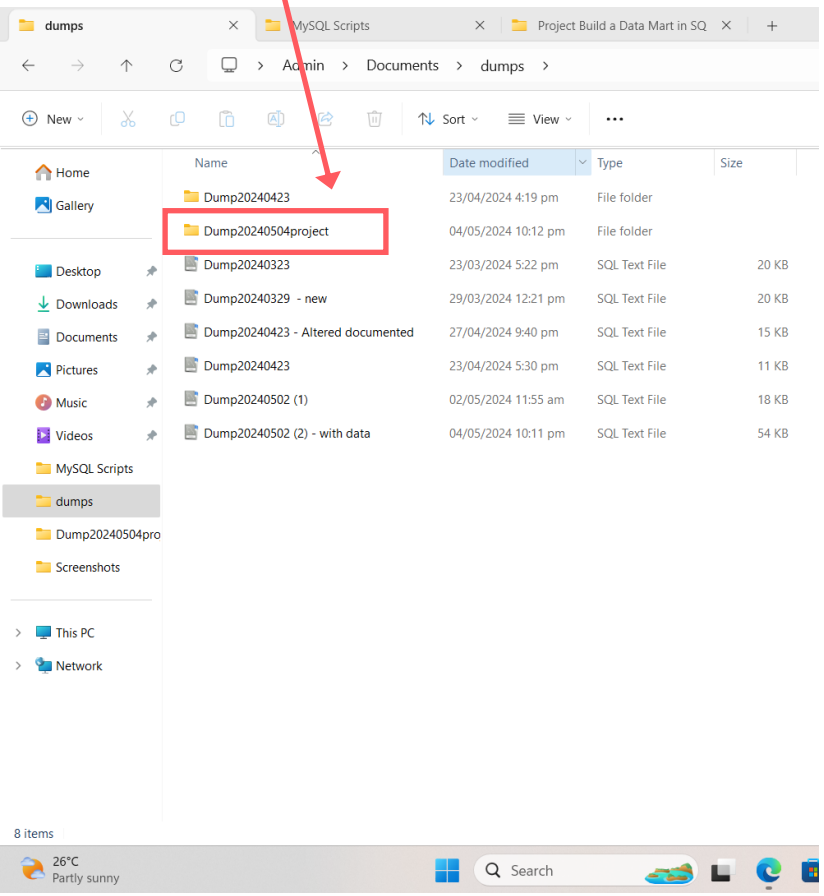
2. Open MySQL Workbench and create a connection.



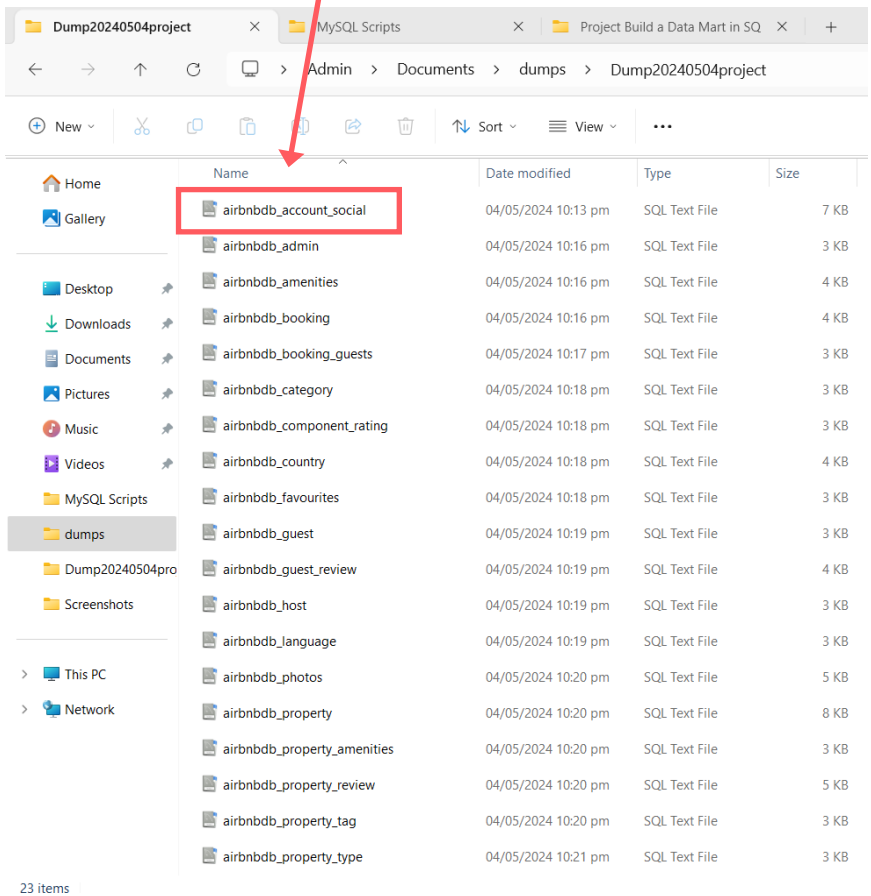
3. Create new schema / database then select “apply”.

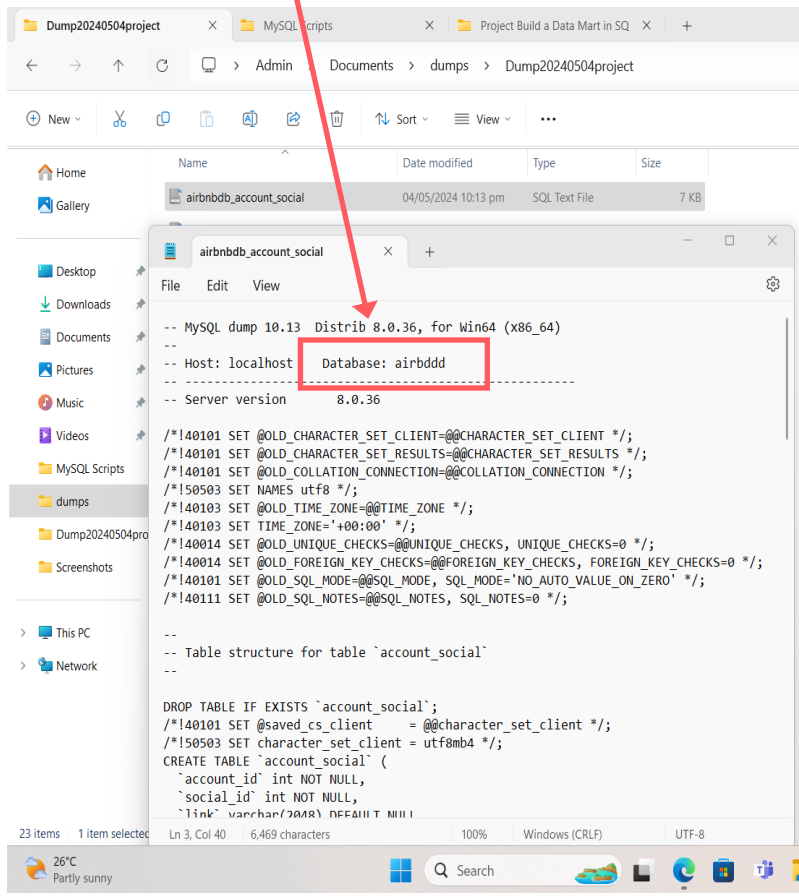


4. In “File explorer”, open the folder that contains the database structure and data to import.

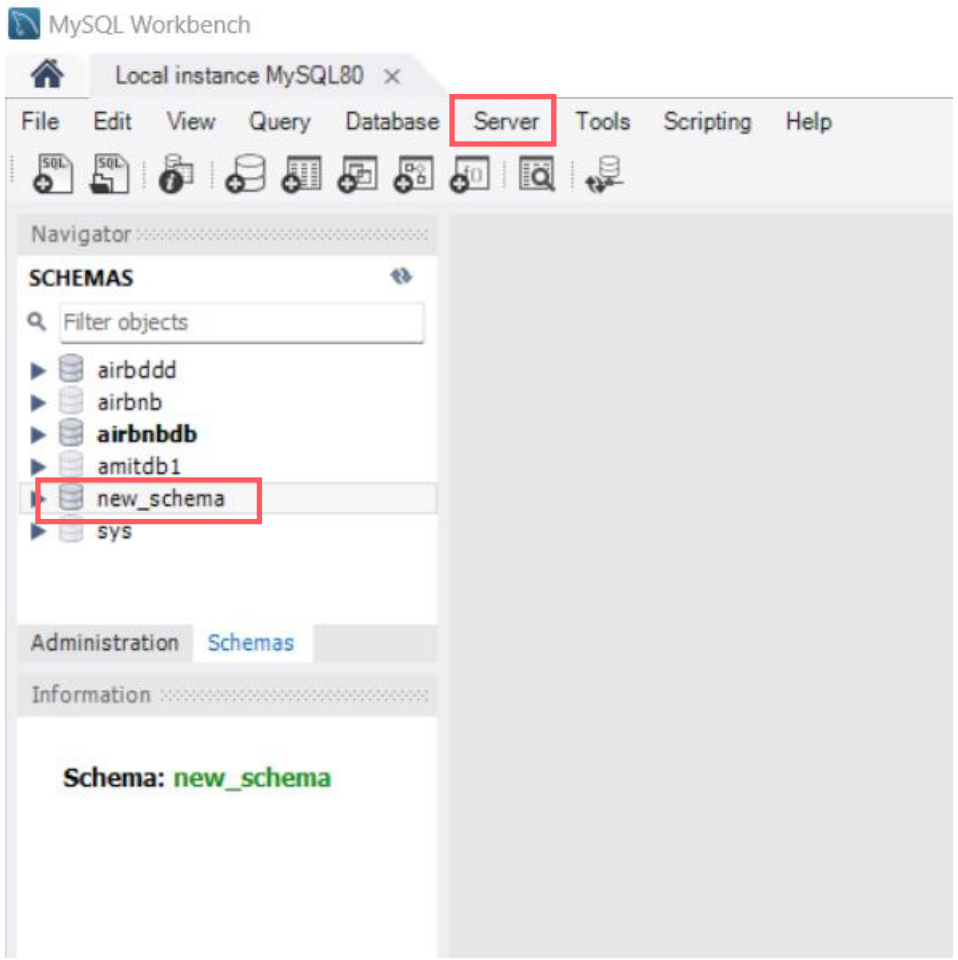


5. Edit each script file in Notepad and change the name of the database to the newly created database in workbench, “new_schema”.



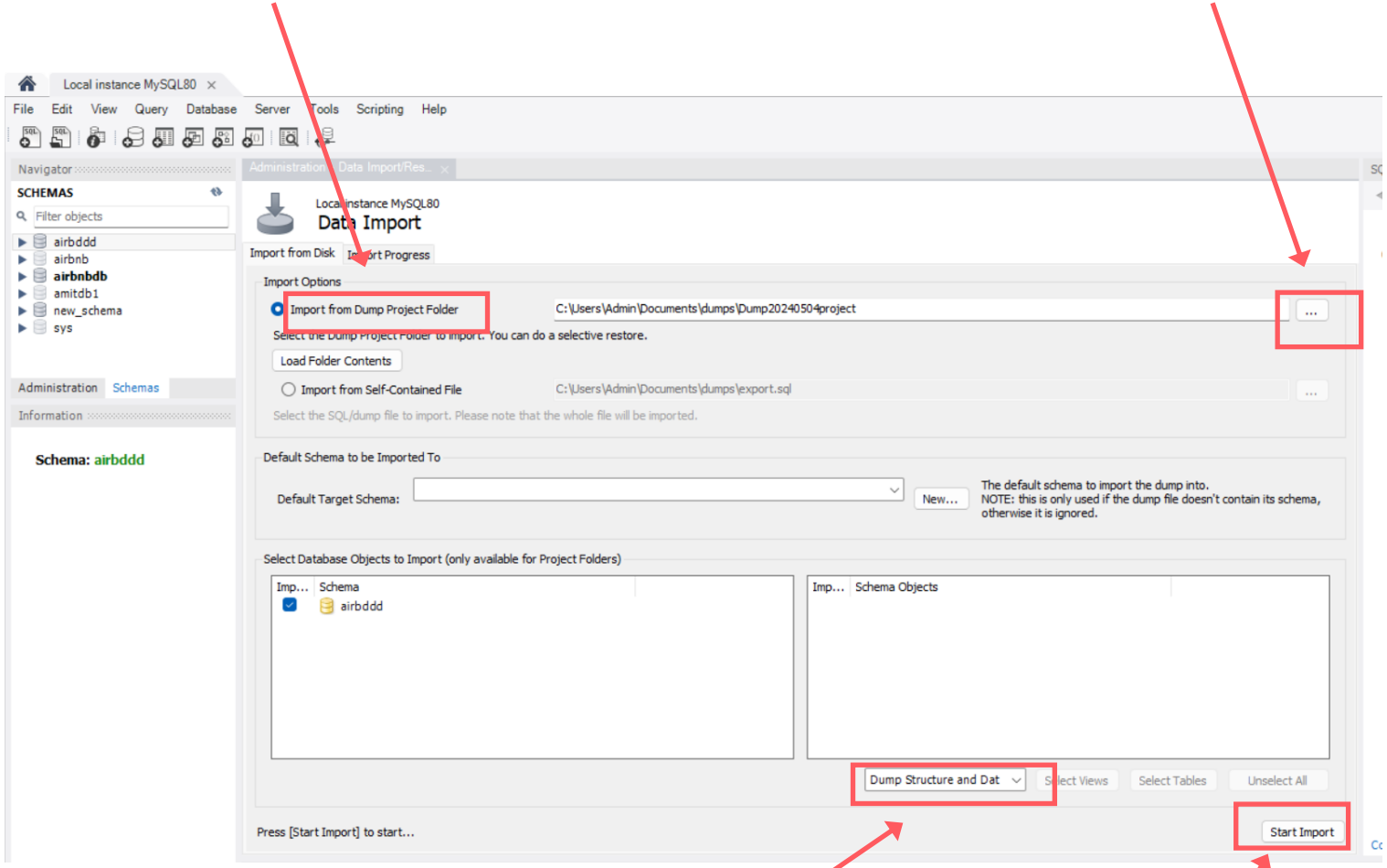


6. Back in Workbench, select “server” in the toolbar and choose “Data import”.



7. The following appears, then select “Import from dump project folder”.

8. Choose the folder containing the structure and data of the database by clicking “…”.



9. Ensure “Dump Structure and Data” is selected

10. Finally select “start import”

6. Table Statements

Table: Property

Table Creation

```sql
CREATE TABLE property (
  property_id int NOT NULL AUTO_INCREMENT,
  type_id int DEFAULT NULL,
  tag_id int DEFAULT NULL,
  host_id int DEFAULT NULL,
  country_id int DEFAULT NULL,
  `description` mediumtext,
  nightly_price_dollars decimal(10,2) NOT NULL,
  property_name varchar(100) NOT NULL,
  number_of_guests int NOT NULL,
  number_of_bedrooms int NOT NULL,
  number_of_bed int NOT NULL,
  number_of_bath int NOT NULL,
  address_line_1 varchar(250) NOT NULL,
  adrress_line_2 varchar(45) NOT NULL,
  PRIMARY KEY (property_id),
  KEY FK__property_type_id (type_id),
  KEY FK_property_host_id (host_id),
  KEY FK_property_country_id (country_id),
  KEY FK_property_tag_id_idx (tag_id),
  CONSTRAINT FK__property_type_id FOREIGN KEY (type_id) REFERENCES property_type (type_id),
  CONSTRAINT FK_property_country_id FOREIGN KEY (country_id) REFERENCES country (country_id),
  CONSTRAINT FK_property_host_id FOREIGN KEY (host_id) REFERENCES `host` (host_id),
  CONSTRAINT FK_property_tag_id FOREIGN KEY (tag_id) REFERENCES property_tag (tag_id)
);
```

- The table structure reflects the finalized design for MySQL dumps.
- However, it requires creating referenced tables before applying constraints to maintain data integrity.

Test Case

1. SQL Statement: Lines 170 to 185 are executed initially, followed by the addition of lines 186 to 193 within the "Alter Table" statement as depicted below. This sequence applies to all other tables within the database.

```sql
ALTER TABLE table1
ADD CONSTRAINT FK_constraint_name FOREIGN KEY (column_id)
REFERENCES table2(column_id);
```

2. Once set up, we can now use queries like "Insert" and "Select" for testing. Here's an example of an "Insert" statement and its result..

```sql
INSERT INTO `airbddd`.`property`
(`property_id`, `type_id`, `tag_id`, `host_id`, `country_id`,
`description`, `nightly_price_dollars`, `property_name`, `number_of_guests`,
`number_of_bedrooms`, `number_of_bed`, `number_of_bath`, `address_line_1`, `adrress_line_2`)
VALUES ('41', '3', '12', '14', '40',
'Industrial-style loft in trendy neighborhood', '400.00', 'Trendy Loft', '4',
'1', '2', '1', '333 Beach Drive', 'Unit 51');
```

3. Below you can see the table structure and the new row created from the “Insert” statement.

property_id	type_id	tag_id	host_id	country_id	description	nightly_price_dollars	property_name	number_of_guests
31	3	3	25	31	Bright and airy apartment in downtown area	130.00	Bright Downtown Apar...	2
32	4	4	26	32	Elegant condo with garden views	280.00	Garden View Condo	2
33	1	5	27	33	Rustic cabin with mountain views	170.00	Mountain Cabin	4
34	2	6	28	34	Modern loft with downtown skyline views	240.00	Downtown Skyline Loft	2
35	3	7	29	35	Beachfront house with private pool	550.00	Beach House Retreat	8
36	4	8	30	36	Charming apartment in historic building	110.00	Historic Apartment	2
37	1	9	31	37	Modern condo with scenic river views	320.00	River View Condo	2
38	2	10	32	38	Cozy cottage with lakefront access	160.00	Lakefront Cottage	4
39	3	11	13	39	Industrial-style loft in trendy neighborhood	180.00	Trendy Loft	2
40	4	12	14	40	Spacious condo with panoramic mountain views	400.00	Mountain View Condo	6
41	3	12	14	40	Industrial-style loft in trendy neighborhood	400.00	Trendy Loft	4

number_of_bedrooms	number_of_bed	number_of_bath	address_line_1	adrress_line_2
1	1	1	888 City Street	Unit 401
1	1	1	999 Garden Avenue	Suite 102
2	2	1	111 Mountain Road	
1	1	1	222 Downtown Boulevard	Floor 5
4	4	3	333 Beach Drive	
1	1	1	444 Historic Street	Unit 201
1	1	1	555 Riverside Avenue	Suite 301
2	2	1	666 Lakeview Lane	
1	1	1	777 Trendy Street	Floor 2
3	3	2	888 Mountain Road	
1	2	1	333 Beach Drive	Unit 51

Table: Account Social

Table Creation

```sql
CREATE TABLE `account_social` (
    `account_id` int NOT NULL,
    `social_id` int NOT NULL,
    `link` varchar(2048) DEFAULT NULL,
    PRIMARY KEY (`account_id`,`social_id`),
    KEY `FK_account_social_social_id` (`social_id`),
    CONSTRAINT `FK_account_social_account_id` FOREIGN KEY (`account_id`) REFERENCES `user_account` (`account_id`),
    CONSTRAINT `FK_account_social_social_id` FOREIGN KEY (`social_id`) REFERENCES `social_profile` (`social_id`)
);
```

- **Role:**
- A junction table.
- facilitates efficient retrieval and management of social profile data linked to specific user accounts.

- **Column:**
- *account_id:* Identifies the account.
- *social_id:* Identifies the social profile
- *link:* Holds the URL link associated with the social profile. Link column can hold up to 2048 characters.
- *Primary Key*: Combination of account_id and social_id.

- **Indexes**:
- *FK_account_social_social_id*: Index for the social_id column.

- **Foreign Key Constraints**:
- *FK_account_social_account_id*: Links account_id to the account_id column in the 'user_account' table.
- *FK_account_social_social_id*: Links social_id to the social_id column in the 'social_profile' table.

Test Case

1. Below is a table filled with data containing over 111 rows. However, due to space constraints, we'll provide a condensed version here.

```sql
SELECT * FROM airbddd.account_social;
```

account_id	social_id	link
1	1	https://www.example.com/page
1	2	https://www.example.com/page
2	3	https://www.example.com/page
3	4	https://www.example.com/page
4	5	https://www.example.com/page
4	6	https://www.example.com/page
4	7	https://www.example.com/page
5	1	https://www.example.com/page
5	2	https://www.example.com/page
6	3	https://www.example.com/page
7	4	https://www.example.com/page
7	5	https://www.example.com/page
7	6	https://www.example.com/page
8	1	https://www.example.com/page
8	7	https://www.example.com/page
9	2	https://www.example.com/page
9	3	https://www.example.com/page
10	4	https://www.example.com/page
11	5	https://www.example.com/page
12	6	https://www.example.com/page
13	7	https://www.example.com/page
14	1	https://www.example.com/page
15	2	https://www.example.com/page

2. let us select all rows with account_id = 4.

```sql
SELECT * FROM airbddd.account_social WHERE account_id = 4;
```

account_id	social_id	link
4	5	https://www.example.com/page
4	6	https://www.example.com/page
4	7	https://www.example.com/page
NULL	NULL	NULL

Table: Admin

Table Creation

```
CREATE TABLE `admin` (
    `admin_id` int NOT NULL AUTO_INCREMENT,
    `employee_management_dashboard` tinyint(1) DEFAULT NULL,
    `trips_management` tinyint(1) DEFAULT NULL,
    `reports_management` tinyint(1) DEFAULT NULL,
    `invoices_management` tinyint(1) DEFAULT NULL,
    PRIMARY KEY (`admin_id`)
);
```

- **Role:** Stores administrative privileges for users.

- **Column:**

- *admin_id:* is the primary key.
- Other columns denote various administrative privileges.
- Mysql doesn't have boolean datatype, tinyint is used as an alternative where 1 = YES and 0 = NO.

Test Case

1. Selecting all data from the table.

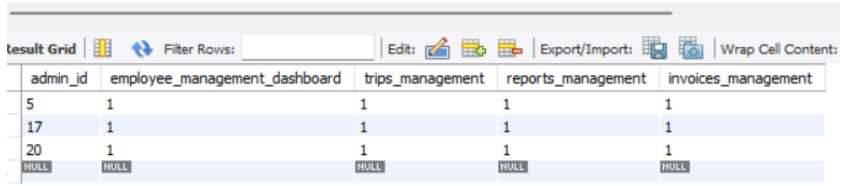
```
SELECT * FROM airbddd.admin;
```

admin_id	employee_management_dashboard	trips_management	reports_management	invoices_management
1	1	1	1	0
2	0	0	1	1
3	1	0	1	0
4	0	1	0	1
5	1	1	1	1
6	1	1	1	0
7	1	1	0	0
8	1	0	0	0
9	0	0	0	1
10	0	0	1	1
11	0	1	1	1
12	0	1	0	0
13	0	0	1	0
14	1	1	0	0
15	0	0	1	1
16	1	0	0	1
17	1	1	1	1
18	0	1	1	1
19	1	1	0	1
20	1	1	1	1

Admins can have varying administrative privileges

2. Some admins can have all of the privileges

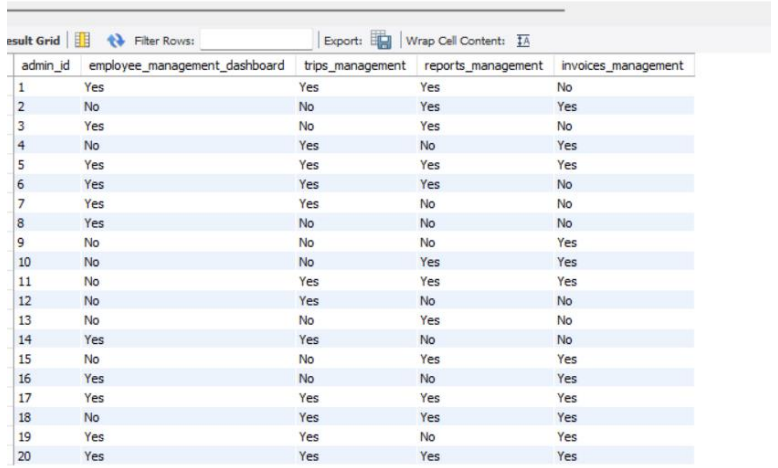
```
SELECT * FROM airbddd.admin
WHERE employee_management_dashboard =1
AND trips_management = 1
AND reports_management = 1
AND invoices_management = 1;
```



admin_id	employee_management_dashboard	trips_management	reports_management	invoices_management
5	1	1	1	1
17	1	1	1	1
20	1	1	1	1
NULL	NULL	NULL	NULL	NULL

3. Returning BOOLEAN “Yes” and “No” based on the TINYINT values.

```
SELECT
    admin_id,
    IF(employee_management_dashboard = 1, 'Yes', 'No') AS employee_management_dashboard,
    IF(trips_management = 1, 'Yes', 'No') AS trips_management,
    IF(reports_management = 1, 'Yes', 'No') AS reports_management,
    IF(invoices_management = 1, 'Yes', 'No') AS invoices_management
FROM airbnbdb.admin;
```



admin_id	employee_management_dashboard	trips_management	reports_management	invoices_management
1	Yes	Yes	Yes	No
2	No	No	Yes	Yes
3	Yes	No	Yes	No
4	No	Yes	No	Yes
5	Yes	Yes	Yes	Yes
6	Yes	Yes	Yes	No
7	Yes	Yes	No	No
8	Yes	No	No	No
9	No	No	No	Yes
10	No	No	Yes	Yes
11	No	Yes	Yes	Yes
12	No	Yes	No	No
13	No	No	Yes	No
14	Yes	Yes	No	No
15	No	No	Yes	Yes
16	Yes	No	No	Yes
17	Yes	Yes	Yes	Yes
18	No	Yes	Yes	Yes
19	Yes	Yes	No	Yes
20	Yes	Yes	Yes	Yes

Table: Amenities

Table Creation

```sql
CREATE TABLE `amenities` (
    `amenities_id` int NOT NULL AUTO_INCREMENT,
    `category_id` int NOT NULL,
    `amenity_name` varchar(50) NOT NULL,
    `description` mediumtext,
    PRIMARY KEY (`amenities_id`),
    KEY `FK_category_id` (`category_id`),
    CONSTRAINT `FK_category_id` FOREIGN KEY (`category_id`) REFERENCES `category` (`category_id`)
);
```

- **Role**: Stores information about various amenities.

- **Columns:**
- *amenities_id:* Primary key, auto-incremented integer. Unique identifier for each amenity.
- *category_id*: Identifies the category to which the amenity belongs.
- *description:* Mediumtext. Description of the amenity.

- **Foreign Key Constraint:**
- *FK_category_id*: Links category_id to the category_id column in the 'category' table.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.amenities;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

amenities_id	category_id	amenity_name	description
15	6	Smoke alarm	NULL
16	7	Wifi	NULL
17	8	Microwave	NULL
18	8	Dishes and sil...	Bowls, chopsticks, plates, cups, etc.
19	8	Mini fridge	NULL
20	8	Hot water kettle	NULL
21	8	Coffee maker...	NULL
22	8	Barbecue ute...	Grill, charcoal, bamboo skewers/iron skew...
23	8	coffee	NULL
24	9	BBQ grill: char...	NULL
25	10	Free parking ...	NULL
26	10	Free street p...	NULL
27	10	Single level h...	No stairs in home
28	11	Luggage drop...	For guests' convenience when they have ...
29	11	Breakfast	Breakfast is provided
30	12	Kitchen	NULL
31	12	TV	NULL
32	12	Washer	NULL
33	12	Dryer	NULL
34	12	Air conditioning	NULL
35	12	Carbon mono...	This place may not have carbon monoxide...
36	12	Heating	NULL
NULL	NULL	NULL	NULL

2. Rows are up to 36.

3. A simple SELECT statement with a WHERE clause to filter out the rows where the description column is not null

```sql
SELECT * FROM airbddd.amenities WHERE description IS NOT NULL;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

amenities_id	category_id	amenity_name	description
9	3	Essentials	Towels, bed sheets, soap, and toilet paper
18	8	Dishes and silverware	Bowls, chopsticks, plates, cups, etc.
22	8	Barbecue utensils	Grill, charcoal, bamboo skewers/iron skewers, etc.
27	10	Single level home	No stairs in home
28	11	Luggage dropoff allowed	For guests' convenience when they have early ...
29	11	Breakfast	Breakfast is provided
35	12	Carbon monoxide alarm	This place may not have carbon monoxide dete...
NULL	NULL	NULL	NULL

Table: Booking

Table Creation

```sql
CREATE TABLE `booking` (
  `booking_id` int NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `checkin_date` date DEFAULT NULL,
  `checkout_date` date DEFAULT NULL,
  `nightly_price_dollars` decimal(10,2) DEFAULT NULL,
  `total_nightly_price_dollars` decimal(10,2) DEFAULT NULL,
  `service_fee_dollars` decimal(10,2) DEFAULT NULL,
  `cleaning_fee_dollars` decimal(10,2) DEFAULT NULL,
  `total_price_dollars` decimal(10,2) DEFAULT NULL,
  PRIMARY KEY (`booking_id`),
  KEY `FK_property_id` (`property_id`),
  CONSTRAINT `FK_property_id` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
);
```

- **Role:**
- Facilitates the storage and retrieval of booking data related to specific properties.

- **Columns:**.
- *nightly_price_dollars:* Price per night for the booking.
- *total_nightly_price_dollars:* Total price for all nights of the booking.
- *total_price_dollars:* Total price including all fees for the booking.

- **Foreign Key Constraint:**
- *FK_property_id*: Links the "property_id" column to the "property_id" column in the "property" table, ensuring referential integrity.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.booking;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

booking_id	property_id	checkin_date	checkout_date	nightly_price_dollars	total_nightly_price_dollars	service_fee_dollars	cleaning_fee_dollars	total_price_dollars
1	1	2024-04-01	2024-04-05	500.00	2000.00	20.00	10.00	2030.00
2	1	2024-04-06	2024-05-06	500.00	15000.00	25.00	12.50	15037.50
3	2	2024-04-05	2024-04-10	75.00	375.00	18.00	0.00	393.00
4	2	2024-04-11	2024-05-05	75.00	1800.00	22.00	11.00	1833.00
5	2	2024-05-06	2024-05-15	75.00	675.00	21.00	10.50	706.50
6	3	2024-04-11	2024-04-15	300.00	1200.00	23.00	11.50	1234.50
7	4	2024-04-13	2024-05-13	100.00	3000.00	19.00	9.50	3028.50
8	5	2024-04-15	2024-05-15	120.00	3600.00	25.00	12.50	3637.50
9	6	2024-04-17	2024-04-20	150.00	450.00	26.00	13.00	489.00
10	7	2024-04-19	2024-04-29	90.00	900.00	17.00	8.50	925.50
11	8	2024-04-21	2024-05-21	250.00	7500.00	27.00	13.50	7540.50
12	9	2024-04-23	2024-04-25	180.00	360.00	28.00	0.00	388.00
13	10	2024-04-25	2024-04-28	280.00	840.00	29.00	14.50	883.50
14	11	2024-04-27	2024-05-06	200.00	1800.00	30.00	0.00	1830.00
15	12	2024-04-29	2024-05-01	150.00	300.00	31.00	15.50	346.50
16	13	2024-05-01	2024-05-10	120.00	1080.00	32.00	16.00	1128.00
17	14	2024-05-03	2024-05-06	180.00	540.00	33.00	16.50	589.50
18	15	2024-05-05	2024-05-28	220.00	5060.00	34.00	0.00	5094.00
19	16	2024-05-07	2024-06-09	100.00	3300.00	35.00	17.50	3352.50
20	17	2024-05-09	2024-06-01	160.00	3680.00	36.00	0.00	3716.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. This statement ensures the respective column is populated with accurate information based on the duration of stay and nightly price.

```sql
UPDATE booking
SET total_nightly_price_dollars = DATEDIFF(checkout_date, checkin_date) * nightly_price_dollars;
```

Table: Booking Guests

Table Creation

```sql
CREATE TABLE `booking_guests` (
  `booking_id` int NOT NULL,
  `guest_id` int NOT NULL,
  `num_guests` int DEFAULT NULL,
  `booking_status` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`booking_id`,`guest_id`),
  KEY `FK_guest_id` (`guest_id`),
  CONSTRAINT `FK_booking_id` FOREIGN KEY (`booking_id`) REFERENCES `booking` (`booking_id`),
  CONSTRAINT `FK_guest_id` FOREIGN KEY (`guest_id`) REFERENCES `guest` (`guest_id`)
);
```

- **Role:**

- A junction table that facilitates the association of guests with specific bookings and tracks relevant information such as the number of guests and booking status.

- **Composite Primary Key:**

- Combination of booking_id and guest_id ensures uniqueness for each guest entry in the table.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.booking_guests;
```

booking_id	guest_id	num_guests	booking_status
3	19	2	Pending
4	4	2	Pending
5	5	1	Expired
6	6	4	Cancelled
7	4	3	Confirmed
7	7	3	Expired
7	11	4	Pending
8	8	2	Rejected
9	9	6	Rejected
10	10	2	Confirmed
11	11	3	Confirmed
12	3	6	Confirmed
12	8	2	Confirmed
12	12	3	Cancelled
12	16	1	Rejected
13	13	1	Pending
14	14	1	Pending
15	15	2	Expired
16	16	2	Confirmed
17	17	4	Confirmed
18	18	4	Pending
19	19	2	Cancelled
20	20	3	Expired
NULL	NULL	NULL	NULL

2. Returning booking status that are either ‘confirmed’ or ‘rejected.

```sql
SELECT * FROM airbddd.booking_guests
WHERE booking_status IN ('confirmed', 'rejected');
```

booking_id	guest_id	num_guests	booking_status
1	20	5	Confirmed
7	4	3	Confirmed
8	8	2	Rejected
9	9	6	Rejected
10	10	2	Confirmed
11	11	3	Confirmed
12	3	6	Confirmed
12	8	2	Confirmed
12	16	1	Rejected
16	16	2	Confirmed
17	17	4	Confirmed
NULL	NULL	NULL	NULL

3. Selecting rows whose number of guests are greater than or equal to 4.

```sql
SELECT * FROM booking_guests
WHERE num_guests >= 4;
```

booking_id	guest_id	num_guests	booking_status
1	20	5	Confirmed
6	6	4	Cancelled
7	11	4	Pending
9	9	6	Rejected
12	3	6	Confirmed
17	17	4	Confirmed
18	18	4	Pending
NULL	NULL	NULL	NULL

Table: Category

Table Creation

```sql
CREATE TABLE `category` (
  `category_id` int NOT NULL AUTO_INCREMENT,
  `category_name` varchar(50) NOT NULL,
  PRIMARY KEY (`category_id`)
);
```

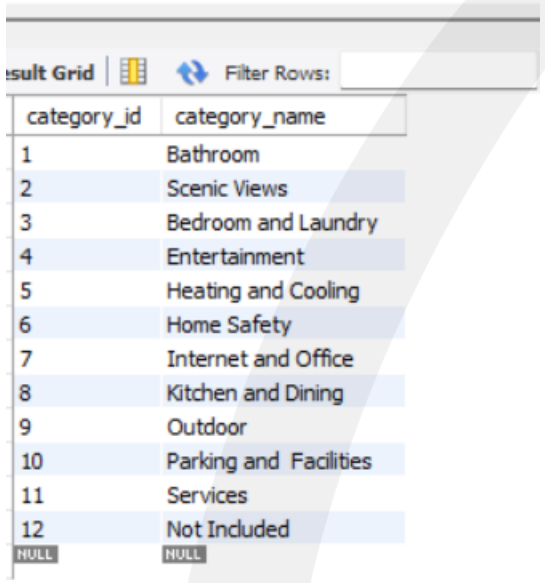
- **Role:**
- Provides a consistent way to organize amenities in the amenities table.

- **Columns:**
- *category_name:* Name or label of the category, providing a descriptive identifier for classification.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.category;
```



Result Grid | Filter Rows:

category_id	category_name
1	Bathroom
2	Scenic Views
3	Bedroom and Laundry
4	Entertainment
5	Heating and Cooling
6	Home Safety
7	Internet and Office
8	Kitchen and Dining
9	Outdoor
10	Parking and Facilities
11	Services
12	Not Included
NULL	NULL

The Airbnb website offers a curated selection of 12 categories, as determined by its filter settings.

Table: Component Rating

Table Creation

```sql
CREATE TABLE `component_rating` (
  `rating_id` int NOT NULL AUTO_INCREMENT,
  `property_review_id` int DEFAULT NULL,
  `cleanliness` int DEFAULT NULL,
  `accuracy` int DEFAULT NULL,
  `check_in` int DEFAULT NULL,
  `communication` int DEFAULT NULL,
  `location` int DEFAULT NULL,
  `value` int DEFAULT NULL,
  PRIMARY KEY (`rating_id`),
  KEY `FK_component_rating_property_review_id_idx` (`property_review_id`),
  CONSTRAINT `FK_component_rating_property_review_id`
  FOREIGN KEY (`property_review_id`) REFERENCES `property_review` (`property_review_id`)
);
```

- **Role:**
- The "component_rating" table stores detailed component ratings for property reviews.

- **Columns:**
- *property_review_id:* Identifies the property review to which the ratings belong.
- *accuracy:* Rating for accuracy of the property description.
- *check_in:* Rating for the check-in experience.
- *communication:* Rating for communication with the property owner or host.
- *location:* Rating for the location of the property.
- *value:* Rating for the value provided by the property.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.component_rating;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

rating_id	property_review_id	cleanliness	accuracy	check_in	communication	location	value
1	1	4	3	4	4	4	3
2	4	5	5	4	5	5	4
3	7	3	3	4	2	3	3
4	8	4	5	5	5	5	5
5	9	5	5	5	5	5	5
6	10	4	4	5	5	5	4
7	11	4	3	4	4	4	3
8	12	3	2	3	2	2	2
9	15	4	4	4	4	4	4
10	16	3	2	3	3	2	4
11	17	2	1	3	2	2	2
12	18	4	2	2	2	2	1
13	19	4	2	4	5	4	4
14	20	3	5	4	5	5	5
15	21	5	5	4	4	4	4
16	22	5	5	5	5	5	5
17	23	3	3	3	2	3	4
18	24	5	4	3	3	3	3
19	25	5	5	5	5	5	5
20	26	5	4	4	1	4	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Calculating the average ratings for each component.

```sql
SELECT
    AVG(cleanliness) AS avg_cleanliness,
    AVG(accuracy) AS avg_accuracy,
    AVG(check_in) AS avg_check_in,
    AVG(communication) AS avg_communication,
    AVG(location) AS avg_location,
    AVG(value) AS avg_value
FROM
    component_rating;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg_cleanliness	avg_accuracy	avg_check_in	avg_communication	avg_location	avg_value
4.0000	3.6000	3.9000	3.6500	3.8000	3.7000

Table: Country

Table Creation

```sql
CREATE TABLE `country` (
    `country_id` int NOT NULL AUTO_INCREMENT,
    `region_id` int DEFAULT NULL,
    `country_name` varchar(100) NOT NULL,
    `city` varchar(100) DEFAULT NULL,
    `state` varchar(100) NOT NULL,
    PRIMARY KEY (`country_id`),
    KEY `FK_region_id` (`region_id`),
    CONSTRAINT `FK_region_id` FOREIGN KEY (`region_id`)
    REFERENCES `region` (`region_id`)
);
```

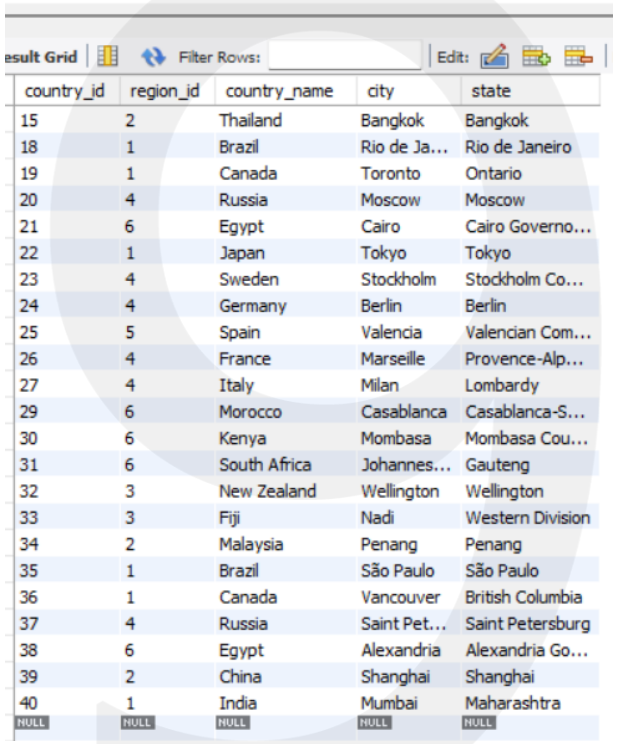
- **Role:**
- Facilitates the storage and retrieval of data pertaining to countries, including their names, cities, and states.

- **Foreign Key Constraint:**
- *FK_region_id:* Links the "region_id" column to the "region_id" column in the "region" table, ensuring referential integrity.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.country;
```

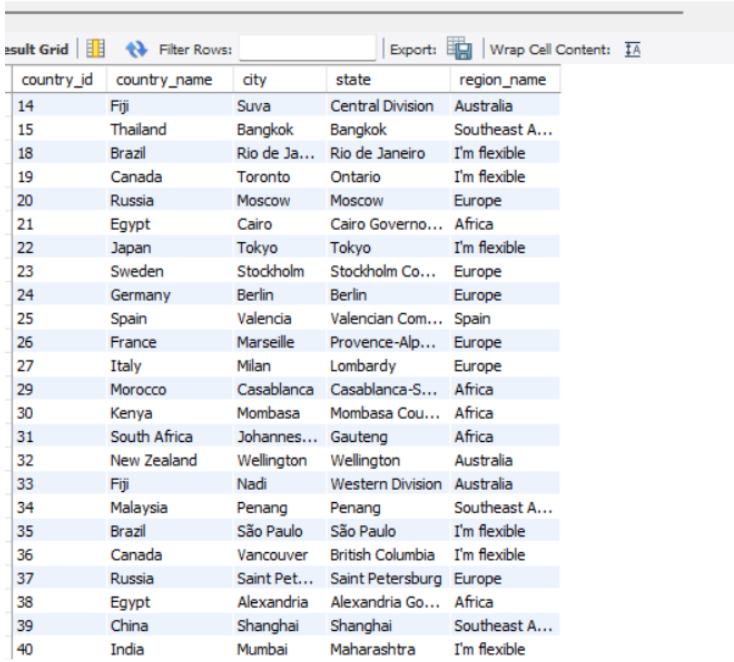


country_id	region_id	country_name	city	state
15	2	Thailand	Bangkok	Bangkok
18	1	Brazil	Rio de Ja...	Rio de Janeiro
19	1	Canada	Toronto	Ontario
20	4	Russia	Moscow	Moscow
21	6	Egypt	Cairo	Cairo Governo...
22	1	Japan	Tokyo	Tokyo
23	4	Sweden	Stockholm	Stockholm Co...
24	4	Germany	Berlin	Berlin
25	5	Spain	Valencia	Valencian Com...
26	4	France	Marseille	Provence-Alp...
27	4	Italy	Milan	Lombardy
29	6	Morocco	Casablanca	Casablanca-S...
30	6	Kenya	Mombasa	Mombasa Cou...
31	6	South Africa	Johannes...	Gauteng
32	3	New Zealand	Wellington	Wellington
33	3	Fiji	Nadi	Western Division
34	2	Malaysia	Penang	Penang
35	1	Brazil	São Paulo	São Paulo
36	1	Canada	Vancouver	British Columbia
37	4	Russia	Saint Pet...	Saint Petersburg
38	6	Egypt	Alexandria	Alexandria Go...
39	2	China	Shanghai	Shanghai
40	1	India	Mumbai	Maharashtra

There are 40 rows in total

2. Query to retrieve all countries along with their associated region names

```sql
SELECT c.country_id, c.country_name, c.city, c.state, r.region_name
FROM country c
LEFT JOIN region r ON c.region_id = r.region_id;
```



country_id	country_name	city	state	region_name
14	Fiji	Suva	Central Division	Australia
15	Thailand	Bangkok	Bangkok	Southeast A...
18	Brazil	Rio de Ja...	Rio de Janeiro	I'm flexible
19	Canada	Toronto	Ontario	I'm flexible
20	Russia	Moscow	Moscow	Europe
21	Egypt	Cairo	Cairo Governo...	Africa
22	Japan	Tokyo	Tokyo	I'm flexible
23	Sweden	Stockholm	Stockholm Co...	Europe
24	Germany	Berlin	Berlin	Europe
25	Spain	Valencia	Valencian Com...	Spain
26	France	Marseille	Provence-Alp...	Europe
27	Italy	Milan	Lombardy	Europe
29	Morocco	Casablanca	Casablanca-S...	Africa
30	Kenya	Mombasa	Mombasa Cou...	Africa
31	South Africa	Johannes...	Gauteng	Africa
32	New Zealand	Wellington	Wellington	Australia
33	Fiji	Nadi	Western Division	Australia
34	Malaysia	Penang	Penang	Southeast A...
35	Brazil	São Paulo	São Paulo	I'm flexible
36	Canada	Vancouver	British Columbia	I'm flexible
37	Russia	Saint Pet...	Saint Petersburg	Europe
38	Egypt	Alexandria	Alexandria Go...	Africa
39	China	Shanghai	Shanghai	Southeast A...
40	India	Mumbai	Maharashtra	I'm flexible

This query can be valuable for various analytical or reporting purposes.

Table: Favourites

Table Creation

```sql
CREATE TABLE `favourites` (
  `account_id` int NOT NULL,
  `property_id` int NOT NULL,
  PRIMARY KEY (`property_id`,`account_id`),
  KEY `FK_favourites_account_id_idx` (`account_id`),
  CONSTRAINT `FK_favourites_account_id` FOREIGN KEY (`account_id`)
  REFERENCES `user_account` (`account_id`),
  CONSTRAINT `FK_favourites_property_id` FOREIGN KEY (`property_id`)
  REFERENCES `property` (`property_id`)
);
```

- **Role:**
- a junction table facilitating the association between user accounts and their favorite properties.

- **Columns:**
- *account_id*: Identifies the user account associated with the favorite property.
- *property_id:* Identifies the property marked as a favorite by the user.

- **Composite Primary Key:**
- Combination of "property_id" and "account_id", ensuring uniqueness for each favorite property per user.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.favourites;
```

account_id	property_id
44	1
44	8
45	2
46	2
47	3
48	3
49	3
50	3
51	9
52	10
53	10
54	11
55	11
56	11
57	12
58	13
59	14
60	15
61	16
61	19
62	17
63	18
64	28
NULL	NULL

2. Let us use a query to count the total number of rows in the "favorites" table.

```sql
SELECT COUNT(*) AS total_rows
FROM favourites;
```

total_rows
49

3. Joined-query - selecting the account_id (‘2’) from the "favourites" table along with the property details from the "property" table.

```sql
SELECT f.account_id, p.property_id, p.property_name, p.address_line_1, p.description
FROM property p
JOIN favourites f ON p.property_id = f.property_id
WHERE f.account_id = 2;
```

account_id	property_id	property_name	address_line_1	description
2	30	Infinity Villa	777 Oceanfront Boulevard	Luxury villa with infinity pool and ocean views
2	31	Bright Downtown Apartment	888 City Street	Bright and airy apartment in downtown area

Table: Guest

Table Creation

```
25 • CREATE TABLE `guest` (
26     `guest_id` int NOT NULL AUTO_INCREMENT,
27     `number_of_reviews` int DEFAULT NULL,
28     `years_on_airbnb` int DEFAULT NULL,
29     PRIMARY KEY (`guest_id`)
30 );
```

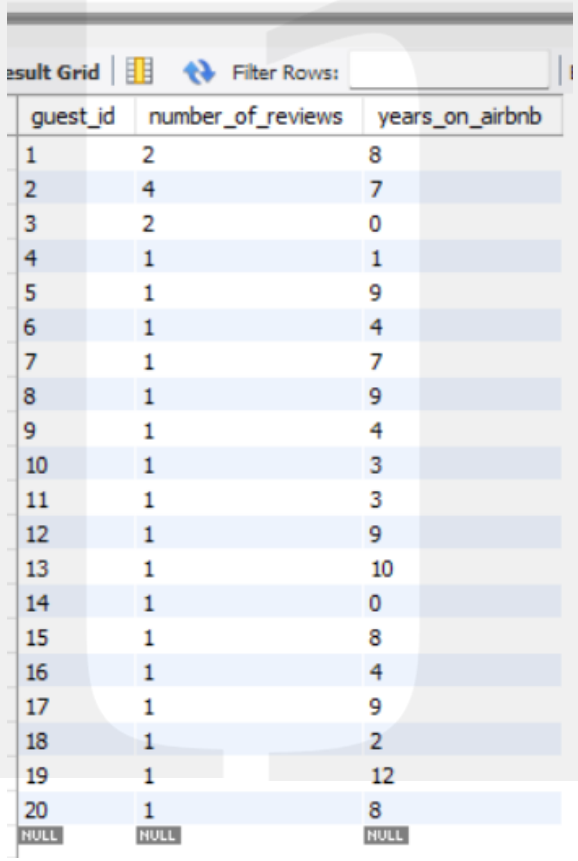
- **Role:**
- Facilitates the storage and retrieval of data related to individual guests.

- **Columns:**
- *number_of_reviews:* Number of reviews the guest has received on the platform.
- *years_on_airbnb:* Number of years the guest has been active on Airbnb or a similar platform.

Test Case

1. Selecting all data from the table.

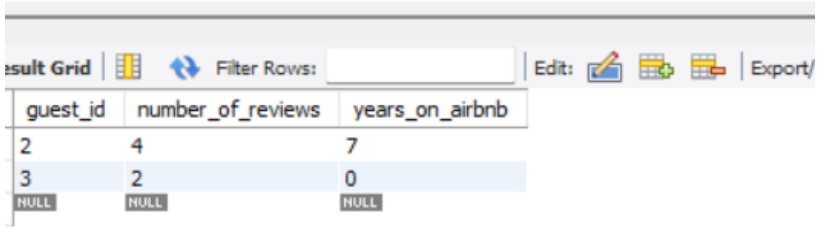
```
1 •  SELECT * FROM airbddd.guest;
```



guest_id	number_of_reviews	years_on_airbnb
1	2	8
2	4	7
3	2	0
4	1	1
5	1	9
6	1	4
7	1	7
8	1	9
9	1	4
10	1	3
11	1	3
12	1	9
13	1	10
14	1	0
15	1	8
16	1	4
17	1	9
18	1	2
19	1	12
20	1	8
NULL	NULL	NULL

2. Query to retrieve guests with the number of reviews being greater than one and the tenure being less than eight.

```
2 •  SELECT * FROM guest
3    WHERE number_of_reviews > 1 AND years_on_airbnb < 8;
4
```



guest_id	number_of_reviews	years_on_airbnb
2	4	7
3	2	0
NULL	NULL	NULL

Table: Guest Review

Table Creation

```sql
CREATE TABLE `guest_review` (
  `guest_review_id` int NOT NULL AUTO_INCREMENT,
  `guest_id` int DEFAULT NULL,
  `content` text,
  `review_date` date DEFAULT NULL,
  PRIMARY KEY (`guest_review_id`),
  KEY `FK_guest_review_guest_id` (`guest_id`),
  CONSTRAINT `FK_guest_review_guest_id` FOREIGN KEY (`guest_id`)
  REFERENCES `guest` (`guest_id`)
);
```

- **Role:**
- The table serves as a repository for hosts reviewing guests.

- **Columns:**
- *guest_review_id:* Unique identifier for each guest review, automatically incremented.
- *guest_id:* Identifies the guest who wrote the review.
- *content:* Text content of the review.
- *review_date:* Date when the review was submitted.

- **Foreign Key Constraint:**
- *FK_guest_review_guest_id:* Links the "guest_id" column to the "guest_id" column in the "guest" table, ensuring referential integrity.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.guest_review;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_review_id	guest_id	content	review_date
3	3	Friendly and easygoing guest, enjoyed hosting ...	2024-04-03
4	4	Respectful of house rules, pleasant stay.	2024-04-04
5	5	Excellent guest, left the place clean and tidy.	2024-04-05
6	6	Wonderful guest, respectful and communicative.	2024-04-06
7	7	Nice and polite, would recommend to other hosts.	2024-04-07
8	8	Pleasure to host, respectful and courteous.	2024-04-08
9	9	Great guest, very considerate and communicati...	2024-04-09
10	10	Perfect guest, would welcome back anytime.	2024-04-10
11	11	Very respectful and easy to communicate with.	2024-04-11
12	12	Fantastic guest, left everything in great condition.	2024-04-12
13	13	Excellent communication and respectful of space.	2024-04-13
14	14	Lovely guest, enjoyed hosting them.	2024-04-14
15	15	Outstanding guest, highly recommended.	2024-04-15
16	16	Very pleasant guest, respectful and kind.	2024-04-16
17	17	Courteous and considerate guest, enjoyed host...	2024-04-17
18	18	Responsible and respectful, great experience.	2024-04-18
19	19	Courteous and considerate guest, enjoyed host...	2024-04-19
20	20	Exceptional guest, left a positive impression.	2024-04-20
21	1	Courteous and considerate guest, enjoyed host...	2024-04-21
22	2	Outstanding guest, highly recommended.	2024-04-22
23	3	Very pleasant guest, respectful and kind.	2024-04-23
24	2	Excellent communication and respectful of space.	2024-04-24
25	2	Nice and polite, would recommend to other hosts.	2024-04-25
NULL	NULL	NULL	NULL

2. Query to retrieve the latest guest reviews along with guest information

```sql
SELECT gr.guest_review_id,
       gr.content,
       gr.review_date,
       g.guest_id,
       g.years_on_airbnb
FROM guest_review gr
JOIN guest g ON gr.guest_id = g.guest_id
ORDER BY gr.review_date DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

guest_review_id	content	review_date	guest_id	years_on_airbnb
25	Nice and polite, would recommend to other hosts.	2024-04-25	2	7
24	Excellent communication and respectful of space.	2024-04-24	2	7
23	Very pleasant guest, respectful and kind.	2024-04-23	3	0
22	Outstanding guest, highly recommended.	2024-04-22	2	7
21	Courteous and considerate guest, enjoyed hosting.	2024-04-21	1	8
20	Exceptional guest, left a positive impression.	2024-04-20	20	8
19	Courteous and considerate guest, enjoyed hosting.	2024-04-19	19	12
18	Responsible and respectful, great experience.	2024-04-18	18	2
17	Courteous and considerate guest, enjoyed hosting.	2024-04-17	17	9
16	Very pleasant guest, respectful and kind.	2024-04-16	16	4

Table: Host

Table Creation

```sql
CREATE TABLE `host` (
    `host_id` int NOT NULL AUTO_INCREMENT,
    `years_hosting` int NOT NULL,
    `listing_count` int DEFAULT NULL,
    `super_host` tinyint DEFAULT NULL,
    `response_rate` varchar(10) DEFAULT NULL,
    `response_time` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`host_id`)
);
```

- **Role:**
- Facilitates the storage and retrieval of host-related data.

- **Columns:**
- *years_hosting:* Number of years the host has been active.
- *listing_count:* Count of listings associated with the host.
- *super_host:* Indicator of whether the host has achieved super host status (1 for yes, 0 for no).
- *response_rate:* Percentage indicating the host's response rate to inquiries.
- *response_time:* Time taken by the host to respond to inquiries.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.host;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

host_id	years_hosting	listing_count	super_host	response_rate	response_time
13	6	3	1	100%	Responds within an hour
14	10	3	0	80%	Responds quick
15	10	4	0	84%	Responds quick
16	9	1	1	90%	Responds within an hour
17	4	1	0	89%	Responds quick
18	2	2	1	97%	Responds within an hour
19	1	2	0	88%	Responds quick
20	7	2	0	80%	Responds quick
21	4	2	0	70%	Satisfactory response
22	9	2	0	75%	Satisfactory response
23	2	2	1	100%	Responds within an hour
24	8	2	0	80%	Responds quick
25	1	2	1	98%	Responds within an hour
26	7	2	1	85%	Responds quick
27	9	2	1	92%	Responds within an hour
28	1	2	0	79%	Satisfactory response
29	5	2	0	88%	Responds quick
30	2	2	0	80%	Responds quick
31	8	2	0	85%	Responds quick
32	10	2	1	100%	Responds within an hour
NULL	NULL	NULL	NULL	NULL	NULL

2. Query to find hosts with a response rate between 70 and 90.

```sql
SELECT *
FROM host
WHERE response_rate BETWEEN '70' AND '90';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

host_id	years_hosting	listing_count	super_host	response_rate	response_time
14	10	3	0	80%	Responds quick
15	10	4	0	84%	Responds quick
17	4	1	0	89%	Responds quick
19	1	2	0	88%	Responds quick
20	7	2	0	80%	Responds quick
21	4	2	0	70%	Satisfactory response
22	9	2	0	75%	Satisfactory response
24	8	2	0	80%	Responds quick
26	7	2	1	85%	Responds quick
28	1	2	0	79%	Satisfactory response
29	5	2	0	88%	Responds quick
30	2	2	0	80%	Responds quick
31	8	2	0	85%	Responds quick
NULL	NULL	NULL	NULL	NULL	NULL

Table: Language

Table Creation

```
25 • CREATE TABLE `language` (
26     `language_id` int NOT NULL AUTO_INCREMENT,
27     `language_name` varchar(100) NOT NULL,
28     PRIMARY KEY (`language_id`)
29 );
```

- **Role:**
- Facilitates the storage and retrieval of information about different languages.

Test Case

1. Selecting all data from the table.

```
1 • SELECT * FROM airbddd.language;
```

Result Grid | Filter Rows:

language_id	language_name
5	Chinese
6	Indian
7	Japanese
8	Korean
9	Malaysian
10	Portugese
11	Spanish
12	Bulgarian
13	Lithuanian
14	Dutch
15	Danish
16	Arabic
17	Hebrew
18	Kenyan
19	Nigerian
20	Egyptian
21	Lebanese
22	Moroccan
23	Pakistan
24	Ethiopian
25	Kiwi
NULL	NULL

Table: Photos

Table Creation

```sql
CREATE TABLE `photos` (
  `photos_id` int NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `url` varchar(2048) DEFAULT NULL,
  `photo_order` int DEFAULT NULL,
  PRIMARY KEY (`photos_id`),
  KEY `FK_photos_property_id` (`property_id`),
  CONSTRAINT `FK_photos_property_id` FOREIGN KEY (`property_id`)
  REFERENCES `property` (`property_id`)
);
```

- **Role:**
- Facilitates the organization and retrieval of photos associated with specific properties.

- **Columns:**
- *property_id:* Identifies the property to which the photo belongs.
- *url:* Holds the URL link to the photo image file. Can hold up to 2048 characters.
- *photo_order:* Specifies the order of the photo in a sequence if multiple photos are associated with a property.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.photos;
```

Result Grid | Filter Rows: | Edit: | Export/Import

photos_id	property_id	url	photo_order
43	20	https://www.example.com/page43	1
44	20	https://www.example.com/page44	2
45	21	https://www.example.com/page45	1
46	22	https://www.example.com/page46	1
47	23	https://www.example.com/page47	1
48	24	https://www.example.com/page48	1
49	25	https://www.example.com/page49	1
50	26	https://www.example.com/page50	1
51	27	https://www.example.com/page51	1
52	28	https://www.example.com/page52	1
53	29	https://www.example.com/page53	1
54	30	https://www.example.com/page54	1
55	31	https://www.example.com/page55	1
56	32	https://www.example.com/page56	1
57	33	https://www.example.com/page57	1
58	34	https://www.example.com/page58	1
59	35	https://www.example.com/page59	1
60	36	https://www.example.com/page60	1
61	37	https://www.example.com/page61	1
62	38	https://www.example.com/page62	1
63	39	https://www.example.com/page63	1
64	40	https://www.example.com/page64	1
NULL	NULL	NULL	NULL

2. Retrieving all photos associated with a specific property, ordered by their display order:.

```sql
SELECT *
FROM photos
WHERE property_id = 14
ORDER BY photo_order;
```

Result Grid | Filter Rows: | Edit: | Export

photos_id	property_id	url	photo_order
27	14	https://www.example.com/page27	1
28	14	https://www.example.com/page28	2
29	14	https://www.example.com/page29	3
NULL	NULL	NULL	NULL

Table: Property Amenities

Table Creation

```sql
CREATE TABLE `property_amenities` (
  `property_id` int NOT NULL,
  `amenities_id` int NOT NULL,
  PRIMARY KEY (`property_id`,`amenities_id`),
  KEY `FK_property_amenities_amenities_id` (`amenities_id`),
  CONSTRAINT `FK_property_amenities_amenities_id` FOREIGN KEY (`amenities_id`)
  REFERENCES `amenities` (`amenities_id`),
  CONSTRAINT `FK_property_amenities_property_id` FOREIGN KEY (`property_id`)
  REFERENCES `property` (`property_id`)
);
```

- **Role:**
- The table serves as a junction table linking properties to their associated amenities.

- **Columns:**
- *property_id:* Identifies the property to which the amenity belongs.
- *amenities_id:* Identifies the specific amenity associated with the property.

- **Composite Primary Key:**
- Combination of "property_id" and "amenities_id", ensuring uniqueness of property-amenity pairs.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.property_amenities;
```

Result Grid | Filter Rows: | Edit:

property_id	amenities_id
35	25
18	26
1	27
18	27
38	27
5	28
18	28
40	28
19	29
39	29
5	30
19	30
20	31
20	32
3	33
20	33
4	34
20	34
1	35
20	35
40	35
21	36
NULL	NULL

2. Retrieving all amenities associated with a particular property.

```sql
SELECT pa.property_id, a.amenities_id, a.amenity_name
FROM property_amenities pa
JOIN amenities a ON pa.amenities_id = a.amenities_id
WHERE pa.property_id = 24;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

property_id	amenities_id	amenity_name
24	4	Body soap
24	5	Hot water

Table: Property Review

Table Creation

```
25 • CREATE TABLE `property_review` (
26     `property_review_id` int NOT NULL AUTO_INCREMENT,
27     `guest_id` int DEFAULT NULL,
28     `property_id` int DEFAULT NULL,
29     `parent_review_id` int DEFAULT NULL,
30     `review_date` date DEFAULT NULL,
31     `comment` text,
32     `overall_rating_stars` int DEFAULT NULL,
33     PRIMARY KEY (`property_review_id`),
34     KEY `FK_property_review_guest_id` (`guest_id`),
35     KEY `FK_property_review_parent_review_id` (`parent_review_id`),
36     KEY `FK_property_review_property_id` (`property_id`),
37     CONSTRAINT `FK_property_review_guest_id` FOREIGN KEY (`guest_id`)
38     REFERENCES `guest` (`guest_id`),
39     CONSTRAINT `FK_property_review_parent_review_id` FOREIGN KEY (`parent_review_id`)
40     REFERENCES `property_review` (`property_review_id`),
41     CONSTRAINT `FK_property_review_property_id` FOREIGN KEY (`property_id`)
42     REFERENCES `property` (`property_id`)
43 );
```

- **Role:**
- Facilitates the storage and retrieval of review data on properties.

- **Columns:**
- *guest_id:* Identifies the guest who submitted the review.
- *property_id:* Identifies the property being reviewed.
- *parent_review_id:* Identifies any parent review associated with the current review (if applicable).
- *review_date:* Date when the review was submitted.
- *comment:* Textual comment provided by the guest.
- *overall_rating_stars:* Numeric rating indicating the overall satisfaction level of the guest.

Test Case

1. Selecting all data from the table.

```
5 •  SELECT * FROM airbddd.property_review;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

property_review_id	guest_id	property_id	parent_review_id	review_date	comment	overall_rating_stars
5	5	2	4	2024-06-02	I didnt. Too much people and very little privacy.	NULL
6	4	2	4	2024-06-02	I honestly think that people is part of what mak...	NULL
7	5	2	NULL	2024-06-01	Too many people. Very little privacy.	3
8	6	3	NULL	2024-04-10	I really like the view!	5
9	7	3	NULL	2024-02-10	Property is well designed and hygenic. The view...	5
10	8	3	NULL	2024-10-18	I really enjoyed the stay.	5
11	10	4	NULL	2024-02-23	Cool.	4
12	11	4	NULL	2024-03-27	Hated the stay. |Most of the amenities listed w...	2
13	10	4	12	2024-03-28	That's a really low rating lol.	NULL
14	11	4	12	2024-04-03	Lmao I know right?! The service was so bad.	NULL
15	12	5	NULL	2024-03-09	Enjoyed the stay.	4
16	13	6	NULL	2024-01-29	Experience was average at best.	3
17	14	6	NULL	2024-02-05	Could have done way better. You didn't fully ex...	2
18	15	6	NULL	2024-03-30	I feel like there's more to be desired.	2
19	16	7	NULL	2024-10-08	NULL	4
20	17	8	NULL	2024-11-28	NULL	5
21	18	9	NULL	2024-11-01	NULL	4
22	19	10	NULL	2024-08-05	NULL	5
23	20	11	NULL	2024-08-28	NULL	3
24	14	11	NULL	2024-07-16	NULL	3
25	2	12	NULL	2024-12-13	NULL	5
26	11	13	NULL	2024-04-18	NULL	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Retrieving the average overall rating for a specific property.

```
6 •  SELECT AVG(overall_rating_stars) AS average_rating
7    FROM property_review
8    WHERE property_id = 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

average_rating
4.0000

Table: Property Tag

Table Creation

```sql
CREATE TABLE `property_tag` (
    `tag_id` int NOT NULL AUTO_INCREMENT,
    `tag_name` varchar(50) NOT NULL,
    PRIMARY KEY (`tag_id`)
);
```

- **Role:**
- Enables the effective categorization and classification of properties based on certain attributes, features, or characteristics..

- **Columns:**
- *tag_name:* Name or label of the property tag, describing its purpose or category.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.property_tag;
```

tag_id	tag_name
7	Islands
8	Arctic
9	Amazing pools
10	OMG!
11	Treehouses
12	Amazing views
13	Design
14	Tropical
15	Trending
16	Farms
17	Countryside
18	Caves
19	National parks
20	Bed & breakfasts
21	New
22	Camping
23	Mansions
24	Vineyards
25	Rooms
26	Earth homes
27	Golfing
28	Skiing
NULL	NULL

2. Retrieving all property tags sorted alphabetically.

```sql
SELECT * FROM property_tag
ORDER BY tag_name;
```

tag_id	tag_name
9	Amazing pools
12	Amazing views
8	Arctic
2	Beach
20	Bed & breakfasts
4	Cabins
6	Campers
22	Camping
18	Caves
17	Countryside
13	Design
26	Earth homes
16	Farms
27	Golfing
1	Historical homes
7	Islands
23	Mansions
19	National parks
21	New
10	OMG!
25	Rooms
28	Skiing
5	Surfing
3	Tiny homes

Table: Property Type

Table Creation

```
25 • CREATE TABLE `property_type` (
26     `type_id` int NOT NULL AUTO_INCREMENT,
27     `type_name` varchar(50) NOT NULL,
28     PRIMARY KEY (`type_id`)
29 );
```

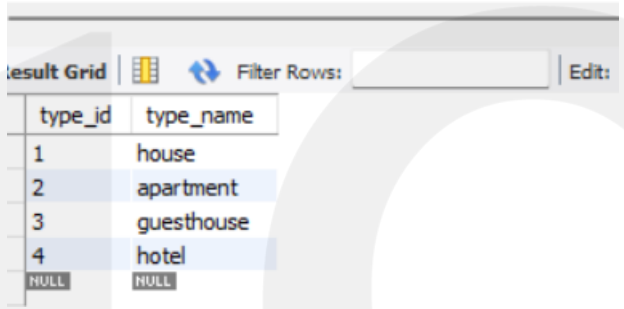
- **Role:**
- classify properties based on their structural characteristics, layout, or usage.

- **Columns:**
- *type_name:* Name or label of the property type, providing a descriptive category for properties.

Test Case

1. Selecting all data from the table.

```
5 • SELECT * FROM airbddd.property_type;
```



type_id	type_name
1	house
2	apartment
3	guesthouse
4	hotel
NULL	NULL

There are only four rows available, as that's the entirety of the listings currently listed on the Airbnb website.

2. Counting the number of properties associated with each property type.

```
2 • SELECT
3       pt.type_id,
4       pt.type_name,
5       COUNT(p.property_id) AS property_count
6   FROM
7       property_type pt
8   LEFT JOIN
9       property p ON pt.type_id = p.type_id
10  GROUP BY
11      pt.type_id, pt.type_name
12  ORDER BY
13      property_count DESC;
14
```

type_id	type_name	property_count
1	house	10
2	apartment	10
3	guesthouse	10
4	hotel	10

Table: Region

Table Creation

```
25 • CREATE TABLE `region` (
26     `region_id` int NOT NULL AUTO_INCREMENT,
27     `region_name` varchar(50) NOT NULL,
28     PRIMARY KEY (`region_id`)
29 );
```

- **Role:**
- Stores the information for the types of regions.

Test Case

1. Selecting all data from the table.

```
5 •  SELECT * FROM airbddd.region;
```

esult Grid | Filter Rows:

region_id	region_name
1	I'm flexible
2	Southeast Asia
3	Australia
4	Europe
5	Spain
6	Africa
NULL	NULL

Only 4 regions as that is the max listed in the official Airbnb website.

2. Retrieving all properties belonging to a specific region.

```
8 •  SELECT p.* FROM property p
9    JOIN country c ON p.country_id = c.country_id
10   JOIN region r ON c.region_id = r.region_id
11   WHERE r.region_name = 'Europe';
12
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

property_id	type_id	tag_id	host_id	country_id	description
6	2	6	18	6	Charming townhouse with garden view
7	3	7	19	7	Rustic cabin nestled in the mountains
8	4	8	20	8	Spacious penthouse with city skyline views
20	4	20	14	20	Riverside cabin with fishing access
23	3	23	17	23	Cozy countryside cottage with fireplace
24	4	24	18	24	Spacious loft with panoramic city views
26	2	26	20	26	Sunny apartment close to downtown attractions
27	3	27	21	27	Modern condo with pool and fitness center
37	1	9	31	37	Modern condo with scenic river views

Table: Social Profile

Table Creation

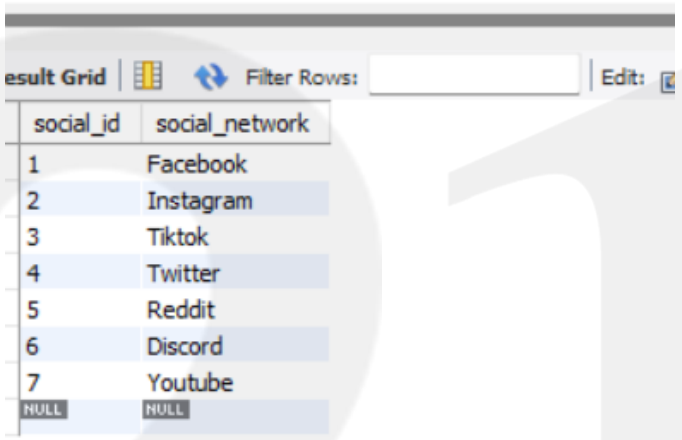
```sql
CREATE TABLE `social_profile` (
    `social_id` int NOT NULL AUTO_INCREMENT,
    `social_network` varchar(50) NOT NULL,
    PRIMARY KEY (`social_id`)
);
```

- **Role:**
- Stores information about social media profiles.

Test Case

1. Selecting all data from the table.

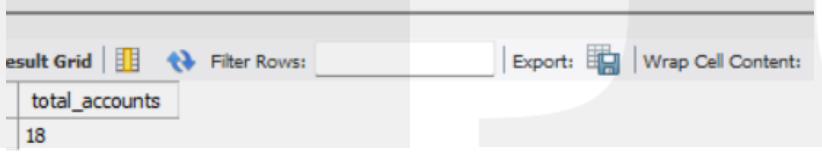
```sql
SELECT * FROM airbddd.social_profile;
```



social_id	social_network
1	Facebook
2	Instagram
3	Tiktok
4	Twitter
5	Reddit
6	Discord
7	Youtube
NULL	NULL

2. Counting the total accounts associated with a specified social network.

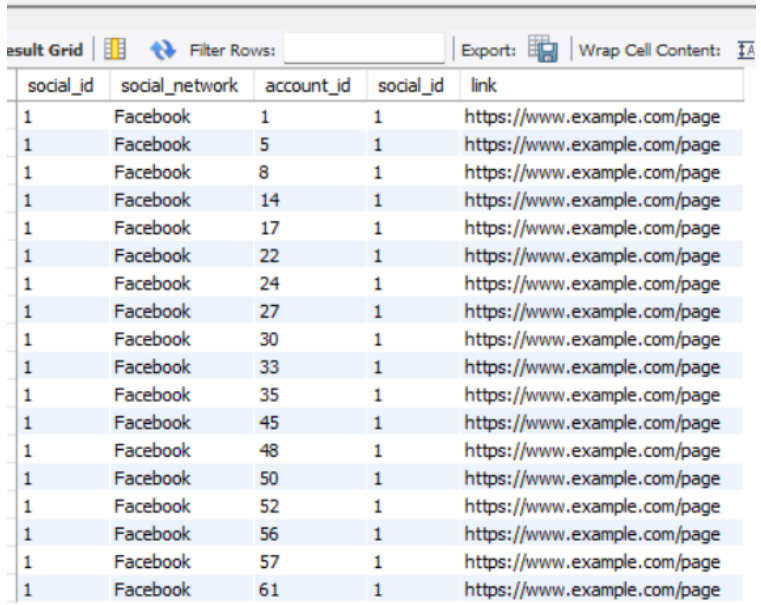
```sql
SELECT COUNT(DISTINCT acs.account_id) AS total_accounts
FROM social_profile sp
JOIN account_social acs ON sp.social_id = acs.social_id
WHERE sp.social_network = 'Facebook';
```



total_accounts
18

3. Retrieving all users associated with a specific social network.

```sql
SELECT * FROM social_profile sp
JOIN account_social acs ON sp.social_id = acs.social_id
WHERE sp.social_network = 'Facebook';
```



social_id	social_network	account_id	social_id	link
1	Facebook	1	1	https://www.example.com/page
1	Facebook	5	1	https://www.example.com/page
1	Facebook	8	1	https://www.example.com/page
1	Facebook	14	1	https://www.example.com/page
1	Facebook	17	1	https://www.example.com/page
1	Facebook	22	1	https://www.example.com/page
1	Facebook	24	1	https://www.example.com/page
1	Facebook	27	1	https://www.example.com/page
1	Facebook	30	1	https://www.example.com/page
1	Facebook	33	1	https://www.example.com/page
1	Facebook	35	1	https://www.example.com/page
1	Facebook	45	1	https://www.example.com/page
1	Facebook	48	1	https://www.example.com/page
1	Facebook	50	1	https://www.example.com/page
1	Facebook	52	1	https://www.example.com/page
1	Facebook	56	1	https://www.example.com/page
1	Facebook	57	1	https://www.example.com/page
1	Facebook	61	1	https://www.example.com/page

Table: User Account

Table Creation

```
25 • CREATE TABLE `user_account` (
26     `account_id` int NOT NULL AUTO_INCREMENT,
27     `country_id` int DEFAULT NULL,
28     `first_name` varchar(255) NOT NULL,
29     `last_name` varchar(255) NOT NULL,
30     `email_address` varchar(255) NOT NULL,
31     `password` varchar(255) DEFAULT NULL,
32     `profile_picture` tinyint DEFAULT NULL,
33     `joined_date` date NOT NULL,
34     `user_role` varchar(255) DEFAULT NULL,
35     `admin_id` int DEFAULT NULL,
36     `host_id` int DEFAULT NULL,
37     `guest_id` int DEFAULT NULL,
38     PRIMARY KEY (`account_id`),
39     KEY `FK_user_account_country_id` (`country_id`),
40     KEY `FK_user_account_admin_idx` (`admin_id`),
41     KEY `FK_user_account_host_idx` (`host_id`),
42     KEY `FK_user_account_guest_idx` (`guest_id`),
43     CONSTRAINT `FK_user_account_admin` FOREIGN KEY (`admin_id`) REFERENCES `admin` (`admin_id`),
44     CONSTRAINT `FK_user_account_country_id` FOREIGN KEY (`country_id`) REFERENCES `country` (`country_id`),
45     CONSTRAINT `FK_user_account_guest` FOREIGN KEY (`guest_id`) REFERENCES `guest` (`guest_id`),
46     CONSTRAINT `FK_user_account_host` FOREIGN KEY (`host_id`) REFERENCES `host` (`host_id`)
47 );
```

- **Role:**
- Table stores user account information.

Test Case

1. Selecting all data from the table.

```
5 •  SELECT * FROM airbddd.user_account;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

account_id	country_id	first_name	last_name	email_address	password	profile_picture	joined_date	user_role	admin_id	host_id	guest_id
46	38	Miles	Five	guest5@example.com	guestpassword5	1	2020-05-24	guest	NULL	NULL	6
47	38	Morales	Six	guest6@example.com	guestpassword6	0	2017-11-07	guest	NULL	NULL	7
48	37	Parker	Seven	guest7@example.com	guestpassword7	1	2015-01-25	guest	NULL	NULL	8
49	36	Peter	Eight	guest8@example.com	guestpassword8	0	2020-10-12	guest	NULL	NULL	9
50	35	Steven	Nine	guest9@example.com	guestpassword9	0	2021-09-15	guest	NULL	NULL	10
51	34	Jake	Ten	guest10@example.com	guestpassword10	0	2021-03-09	guest	NULL	NULL	11
52	33	Amber	Eleven	guest11@example.com	guestpassword11	1	2015-10-26	guest	NULL	NULL	12
53	32	Timber	Twelve	guest12@example.com	guestpassword12	0	2014-07-12	guest	NULL	NULL	13
54	31	Zack	Thirteen	guest13@example.com	guestpassword13	1	2024-03-06	guest	NULL	NULL	14
55	30	Zachary	Fourteen	guest14@example.com	guestpassword14	1	2016-04-27	guest	NULL	NULL	15
56	29	Jack	Fifteen	guest15@example.com	guestpassword15	1	2020-01-24	guest	NULL	NULL	16
57	29	Tim	Sixteen	guest16@example.com	guestpassword16	1	2014-05-09	guest	NULL	NULL	17
58	27	Jake	Seventeen	guest17@example.com	guestpassword17	1	2022-07-27	guest	NULL	NULL	18
59	26	Alice	Eighteen	guest18@example.com	guestpassword18	0	2020-10-15	guest	NULL	NULL	19
60	25	Amber	Nineteen	guest19@example.com	guestpassword19	1	2023-03-28	guest	NULL	NULL	20
61	24	Jake	Twenty	guest20@example.com	guestpassword20	1	2016-10-26	guest	NULL	NULL	20
62	23	Tron	Twentyone	guest21@example.com	guestpassword21	0	2012-03-01	guest	NULL	NULL	19
63	22	Fi	Twentytwo	guest22@example.com	guestpassword22	1	2021-04-06	guest	NULL	NULL	20
64	21	Fo	Twentyone	host21@example.com	hostpassword21	1	2015-08-29	host	NULL	32	NULL
65	20	Fum	Twentytwo	host22@example.com	hostpassword22	1	2013-08-29	host	NULL	31	NULL
66	19	Sif	Twentyone	admin21@example.com	adminpassword21	0	2016-07-19	admin	20	NULL	NULL
67	19	Elijah	twentytwo	admin22@example.com	adminpassword22	0	2014-10-11	admin	19	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: User Language

Table Creation

```sql
CREATE TABLE `user_language` (
  `account_id` int NOT NULL,
  `language_id` int NOT NULL,
  PRIMARY KEY (`account_id`,`language_id`),
  KEY `FK_user_language_language_id` (`language_id`),
  CONSTRAINT `FK_user_language_account_id` FOREIGN KEY (`account_id`)
  REFERENCES `user_account` (`account_id`),
  CONSTRAINT `FK_user_language_language_id` FOREIGN KEY (`language_id`)
  REFERENCES `language` (`language_id`)
);
```

- **Role:**
- Junction table - associates user accounts with their preferred languages.

- **Columns:**
- *account_id:* Identifies the user account.
- *language_id:* Identifies the language spoken by the user.

Composite Primary Key:
- Combination of "account_id" and "language_id" ensures uniqueness and serves as the primary means of identification.

Test Case

1. Selecting all data from the table.

```sql
SELECT * FROM airbddd.user_language;
```

Result Grid | Filter Rows: | Edit:

account_id	language_id
24	16
9	17
38	17
51	17
10	18
25	18
10	19
25	19
39	19
10	20
11	21
27	21
40	21
12	22
27	22
57	22
13	23
52	23
13	24
28	24
42	24
14	25
NULL	NULL

2. Retrieving the languages spoken by a specific user account.

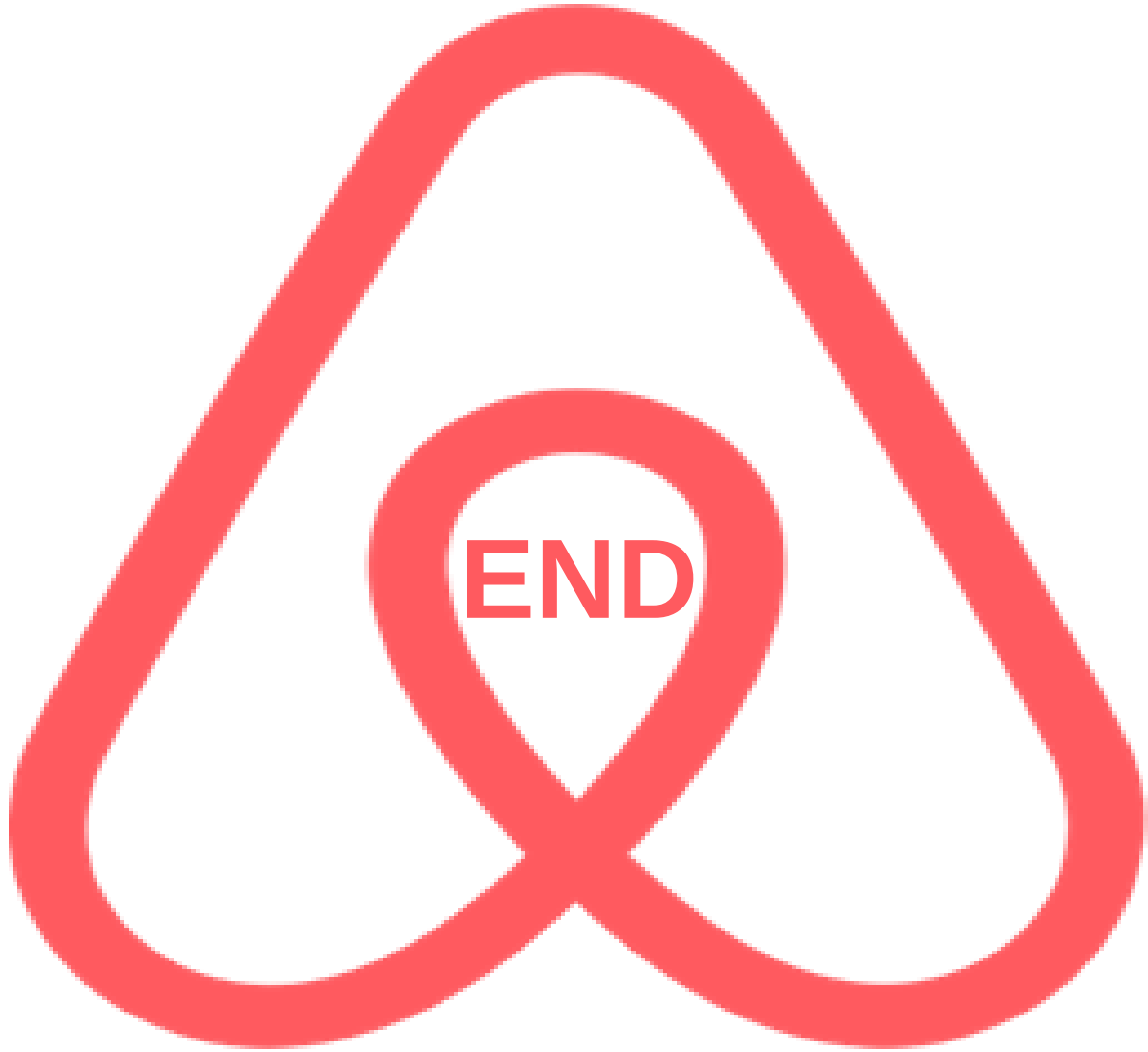
```sql
SELECT language.language_name
FROM user_language
JOIN language ON user_language.language_id = language.language_id
WHERE user_language.account_id = 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

language_name
Kenyan
Nigerian
Egyptian

7. Conclusion

- In summary, the objective of establishing an efficient database for Airbnb has been successfully accomplished.
- The newly updated Entity-Relationship Model (ERM) is juxtaposed with the previous structure to illustrate enhancements in normalization.
- Comprehensive instructions for importing the database are provided in detail.
- Each table of the database is elucidated in terms of its creation process and accompanied by test queries.


END

Course Name: Project: Build a Data Mart in SQL

Course Code: DLBDSPBDM01

Student Name: Jedidiah Prince Amos

Matriculation No: 32009079

ABSTRACT

Airbnb Data Mart

Introduction

This project aimed to build a database for storing and processing information regarding the Airbnb use case. The first step involved developing an entity relationship model (ERM) to describe the data tables, their attributes, and the relationships between them, which served as the foundation for the database. Using a state-of-the-art database management system, the database structure was defined, populated with reasonable dummy data, and thoroughly documented to ensure appropriate usage and effective querying.

Technical Approach

For this project, I chose MySQL as the database management system. MySQL is renowned for its robustness, high performance, and versatility, making it an ideal choice for modeling the Airbnb database. Its scalability and reliability support the structured organization of data, which is essential for complex relationships within Airbnb's data, such as user profiles, property listings, and booking transactions. The extensive documentation, community support, and plethora of online resources available for MySQL facilitate smoother development and troubleshooting. Additionally, MySQL's compatibility with various platforms and seamless integration with numerous programming languages and frameworks provide the necessary flexibility for application development.

For creating the entity relationship model (ERM), I chose Lucidchart due to its user-friendly interface and comprehensive diagramming tools. Lucidchart simplifies the process of creating and modifying ER diagrams with its drag-and-drop interface. Lucidchart enables the creation of detailed and precise ER models that accurately represent complex data structures.

The Airbnb database offers various functionalities such as:

- managing user profiles
- property listings
- booking transactions
- allowing hosts to upload property details, including descriptions, photos, and pricing information
- allowing guests to for properties, view listings, and make reservations
- supporting user ratings and reviews, enabling both hosts and guests to provide feedback on their experiences

- payment processing, ensuring secure transactions between guests and hosts
- administrative functionalities for managing users, properties, and bookings, providing a comprehensive solution for the Airbnb platform.

Metadata

The two figures below showcase the number of tables and corresponding entries and the size of the database regarding its volume.

Figure 1

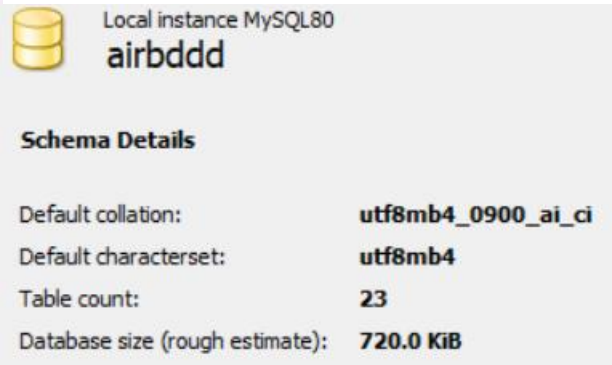


Figure 2

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free
account_social	InnoDB	10	Dynamic	111	147	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
admin	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
amenities	InnoDB	10	Dynamic	36	455	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
booking	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
booking_guests	InnoDB	10	Dynamic	28	585	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
category	InnoDB	10	Dynamic	12	1365	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
component_rating	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
country	InnoDB	10	Dynamic	37	442	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
favourites	InnoDB	10	Dynamic	49	334	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
guest	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
guest_review	InnoDB	10	Dynamic	25	655	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
host	InnoDB	10	Dynamic	20	819	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
language	InnoDB	10	Dynamic	25	655	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
photos	InnoDB	10	Dynamic	64	256	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
property	InnoDB	10	Dynamic	40	409	16.0 KiB	0.0 bytes	64.0 KiB	0.0 bytes
property_amenities	InnoDB	10	Dynamic	93	176	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes
property_review	InnoDB	10	Dynamic	26	630	16.0 KiB	0.0 bytes	48.0 KiB	0.0 bytes
property_tag	InnoDB	10	Dynamic	28	585	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
property_type	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
region	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
social_profile	InnoDB	10	Dynamic	7	2340	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes
user_account	InnoDB	10	Dynamic	67	244	16.0 KiB	0.0 bytes	64.0 KiB	0.0 bytes
user_language	InnoDB	10	Dynamic	100	163	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes

Trials & Triumphs

One of the significant challenges encountered during the development of the Airbnb database was managing the many-to-many (M) relationships. Initial attempt to handle these relationships by including foreign keys directly in the tables proved problematic because it restricted data input to single values, thereby limiting the representation of complex relationships. This issue was effectively resolved by introducing join tables, which facilitated

the accurate modeling of M relationships. These join tables allow for multiple connections between tables and ensuring the integrity and scalability of the database design.

Another challenge was selecting the appropriate data type for the 'response_rate' attribute in the HOST table. Initially, the FLOAT data type was used, but it became apparent that this was insufficient, as it did not accommodate the percentage symbol alongside the numeric value. To address this, the data type was switched to VARCHAR, which can store both the numeric value and the percentage symbol, providing a more accurate and user-friendly representation of the response rate.

On the associate table ‘favourites’, I encountered a duplicate key constraint error named “FK_property_id”. It took considerable time to diagnose, but I eventually discovered the issue stemmed from using the same constraint name on another table. I resolved this by prefixing the constraint name with the respective table name, ensuring unique and clear constraint identifiers across the database.

I also encountered an error where the referencing column ‘amenities_id’ in the foreign key constraint ‘FK_property_amenities_amenities_id’ was incompatible. This issue arose because the columns involved in the foreign key relationship did not have matching data types. I solved this by ensuring that the data types of the referencing column and the referenced column were identical, thus maintaining data type consistency across the relationship.

Conclusion

In conclusion, this project successfully developed a comprehensive database system for Airbnb, addressing the complex relationships and data requirements inherent in the platform. Utilizing MySQL for its robustness and Lucidchart for effective ER modeling, we created a scalable, reliable, and well-documented database structure. The challenges encountered, such as managing relationships and resolving datatype inconsistencies, were effectively overcome, resulting in a robust and efficient system ready for deployment. This project highlights the importance of thorough planning, consistent documentation, and problem-solving skills in database development.