Requirements and Specifications for an Airbnb-like Database.

1.) **Overview**

The database is supposed to serve three primary user roles: Hosts, Guests, Administration. Host lists properties for rent, while Guest searches for and books those properties. Administrators have the rights to edit/adjust bookings and solve tickets. The system also manages payments, reviews, guidelines, and user communication.

2.) **Roles and Actions**

Host: Create/update profile, list/manage properties, accept/reject bookings, receive payments, rate guests.
Guest: Create/update profile, browse/search properties, book properties, make payments, rate hosts and properties.
Admin: Monitor/manage user accounts and profiles, resolve support tickets, enforce guidelines and policies, and maintain overall system integrity.

3.) **Required Data and Functions**

User registration/authentication, property listing/management, booking creation/modification/cancellation, payment processing, review submission/retrieval, messaging, support ticket management, integration of social networks and reviews, guidelines/cancellation policy management.

4.) **Functional Requirements**

User Management: The system provides users with the ability to sign up, create profiles, and log in. It contains the user details such as name, email, phone number, password, and roles, which can be kept within the system. Users can edit their information and notification preferences. Admins can observe user activities by logs for auditing purposes.

Property Management: Allows hosts to create, edit, and delete their property listings. Basic details include title, description of the property, location, pricing, and images. Other things a host can manage in their listings include amenities, guidelines, services and pricing options.

Booking Management: Allows guests to search through the available properties, book their stay, and manage their bookings. As for hosts, they can either approve or reject booking requests. The system keeps a record of booking history, including all the status changes over time. Guests and hosts are also allowed to track the details of their bookings.

Payment Processing: The platform provides a facility for processing the booking payments by guests through various types of payment. It also maintains transaction details like the amount that

was paid, type of payment, date, and other details in such a manner that it ensures all types of booking-related payments can be tracked, reviewed and saved.

Review System: Both the guest and host can leave reviews after a stay. Each review has ratings and comments concerning the stay or booked date and user. The system allows users to view property reviews for a decision to book.

Communication System: The platform has an inbuilt messaging system that provides direct communication between guests and hosts. Users are allowed to send and receive messages concerning bookings for the clarification of details or making inquiries.

Location Management: Arranges properties according to cities, regions, and countries, allowing the users to search and filter the properties by location. Location data helps with the categorization of properties while enhancing the search, and it is a basic and most important criterion for properties.

Support System: Users can raise support tickets for the issues they encounter with the platform. Admins can work on these tickets to resolve problems and report back to the user. Also, the status of a ticket will change while resolution is in process.

Ticket Guidelines Management: They can set rules for their property in advance, such as guidelines or booking conditions. These regulations are connected to each property and are active for a certain period of time during the reservation process, thus clarifying what each party should expect from the other. Admins should also be granted control over this.

Cancellation Policy Management: It allows the hosts to define and manage the cancellation policies for their properties. The policies determine under what circumstances a booking may be cancelled, along with any penalties or refunds applicable to the host to make sure both the host and the guest are satisfied.

Admin Functions: The admin controls the entire platform; from editing bookings, disputes, guidelines enforcement, changing the status of user profiles (such as removing them from the platform), handling support tickets, system activities - they should be able to resolve most of the issues.

Data Table

table_nm	ord	is_key	column_nm	data_typ	nullable	column_descr
address	1	PK	address_id	int(10)	NOT NULL	Unique identifier for each address.
address	2		address_line	varchar(255)	NULL	The street address.
address	3	FK	city_id	int(10)	NULL	Foreign key referencing the city table to associate the address with a city.
amenities	1	PK	amenity_id	int(10)	NOT NULL	Unique identifier for each amenity.
amenities	2		amenity_name	varchar(100)	NULL	The name of the amenity (e.g., "Game room or sauna").
amenities	3		amenity_description	text(65535)	NULL	A description of the amenity.
booking	1	PK	booking_id	int(10)	NOT NULL	Unique identifier for each booking.
booking	2	FK	user_id	int(10)	NOT NULL	Foreign key referencing the user_account table to indicate the user who made the booking.
booking	3	FK	property_id	int(10)	NOT NULL	Foreign key referencing the property table to indicate the booked property.
booking	4		booking_date	timestamp(4)	NULL	Timestamp indicating when the booking was made.
booking	5		number_of_guests	int(10)	NULL	The total number of guests included in the booking.
booking	6		total_price	decimal(10,2)	NULL	The total price of the booking, calculated based on property rates and duration.
booking	7		status_id	int(10)	NULL	Indicates the current status of the booking (e.g., "confirmed", "cancelled").
booking_guideline	1	PK	booking_guideline_id	int(10)	NOT NULL	Unique identifier for each guideline.
booking_guideline	2	FK	booking_id	int(10)	NULL	Foreign key referencing the booking table to associate the guideline with a specific booking.
booking_guideline	3	FK	property_id	int(10)	NULL	Foreign key referencing the property table to link the guideline to a property.
booking_guideline	4		guideline_description	text(65535)	NULL	Detailed description of the guideline or rule.

booking_status_history	1	PK	history_id	int(10)	NOT NULL	Unique identifier for each status change.
booking_status_history	2	FK	booking_id	int(10)	NULL	Foreign key referencing the booking table to associate the history with a booking.
booking_status_history	3		booking_status	varchar(50)	NULL	The status recorded during the change (e.g., “pending”, ”cancelled”, “confirmed”).
booking_status_history	4	FK	changed_by	int(10)	NULL	Foreign key referencing the user_account table for the user who updated the status.
booking_status_history	5	FK	property_id	int(10)	NULL	Foreign key referencing the property table to link the status change with a property.
cancellation_policies	1	PK, UK	policy_id	int(10)	NOT NULL	Unique identifier for each policy.
cancellation_policies	2	FK	property_id	int(10)	NULL	Foreign key referencing the property table to associate policies with properties.
cancellation_policies	3		cancellation_period	timestamp(4)	NULL	The cutoff time before cancellation fees apply.
cancellation_policies	4		penalty_amount	int(10)	NULL	The penalty amount charged for cancellation.
cancellation_policies	5		created_at	timestamp(4)	NULL	Timestamp indicating when the policy was created.
cancellation_policies	6		updated_at	timestamp(4)	NULL	Timestamp indicating the last update to the policy.
city	1	PK	city_id	int(10)	NOT NULL	Unique identifier for each city.
city	2		city_name	varchar(100)	NULL	Name of the city.
city	3	FK	country_id	int(10)	NULL	Foreign key referencing the country table to associate the city with a country.
country	1	PK	country_id	int(10)	NOT NULL	Unique identifier for each country.
country	2		country_name	varchar(100)	NULL	Name of the country.
country	3		country_code	varchar(10)	NULL	Abbreviated 2char code for the country (e.g., "US").
country	4	FK	region_id	int(10)	NULL	Foreign key referencing the region table to link the country to a region.
host	1	FK	user_account_id	int(10)	NULL	Foreign key linking the host to a user account.

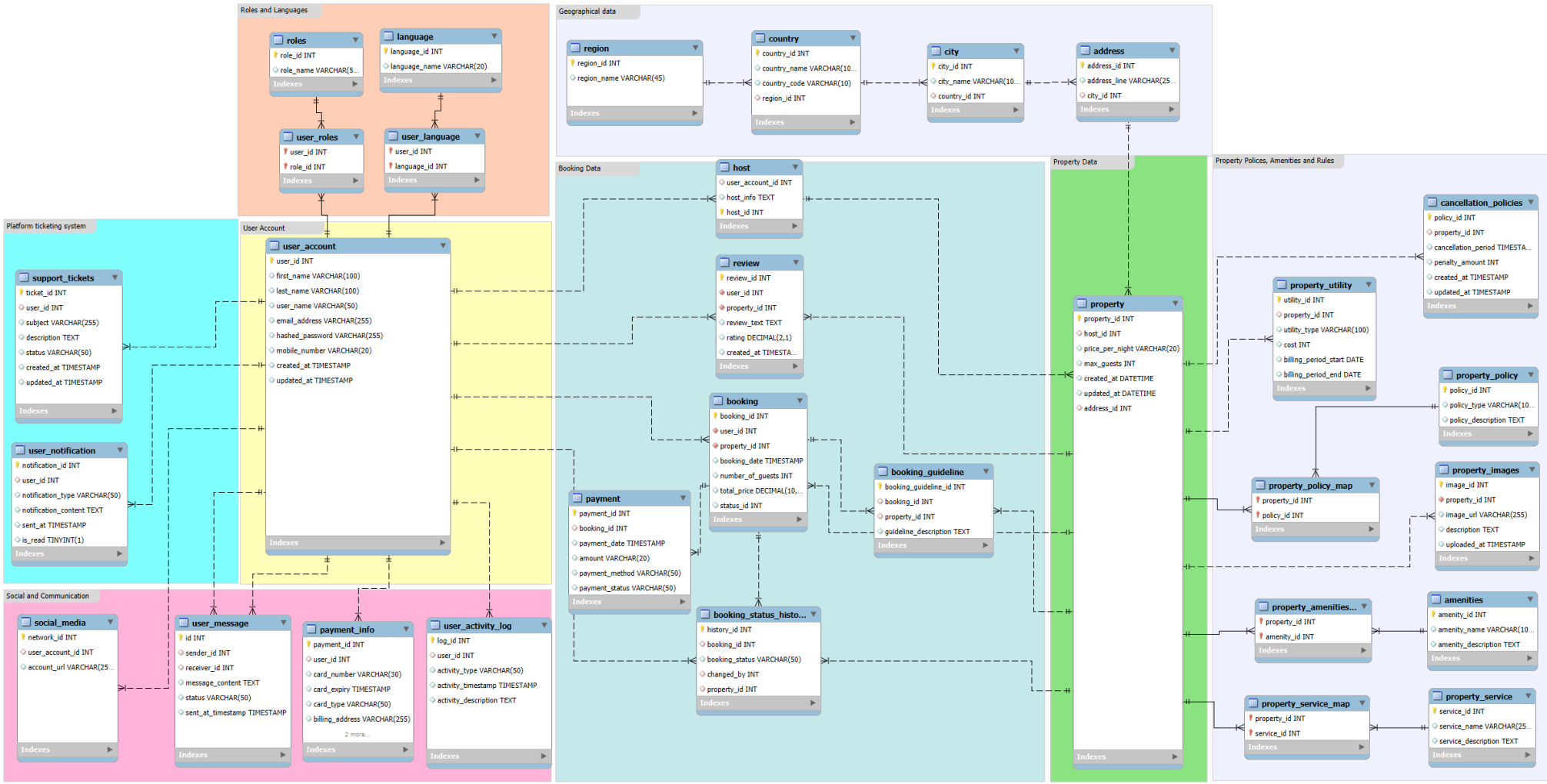
host	2		host_info	text(65535)	NULL	Descriptive information about the host.
host	3	PK	host_id	int(10)	NOT NULL	Unique identifier for each host.
language	1	PK, UK	language_id	int(10)	NOT NULL	Unique identifier for each language.
language	2		language_name	varchar(20)	NULL	Abbreviated name of the language (e.g., “Eng”)
payment	1	PK	payment_id	int(10)	NOT NULL	Unique identifier for each payment.
payment	2	FK	booking_id	int(10)	NULL	Foreign key referencing the booking table to link payment to a booking.
payment	3		payment_date	timestamp(4)	NULL	Timestamp of when the payment was made.
payment	4		amount	varchar(20)	NULL	The payment amount.
payment	5		payment_method	varchar(50)	NULL	The method used for payment (e.g., "Credit Card").
payment	6		payment_status	varchar(50)	NULL	Status of the payment (e.g., "Paid", "Pending").
payment_info	1	PK	payment_id	int(10)	NOT NULL	Foreign key linking payment details to the payment table.
payment_info	2	FK	user_id	int(10)	NULL	Foreign key linking payment information to a user.
payment_info	3		card_number	varchar(30)	NULL	The credit/debit card number.
payment_info	4		card_expiry	timestamp(4)	NULL	The expiration date of the card.
payment_info	5		card_type	varchar(50)	NULL	The type of card (e.g., "Visa", “Mastercard”).
payment_info	6		billing_address	varchar(255)	NULL	The billing address associated with the payment.
payment_info	7		created_at	timestamp(4)	NULL	Timestamp of when the payment information was added.
payment_info	8		method_type	varchar(255)	NULL	Additional method information (e.g., "PayPal").
property	1	PK	property_id	int(10)	NOT NULL	Unique identifier for the property.
property	2	FK	host_id	int(10)	NULL	Foreign key referencing the host table to associate the property with a host.
property	3		price_per_night	varchar(20)	NULL	The nightly rate for renting the property.
property	4		max_guests	int(10)	NULL	Maximum number of guests allowed.
property	5		created_at	datetime(8)	NULL	Timestamp of when the property was listed.
property	6		updated_at	datetime(8)	NULL	Timestamp of the last update to the property details.
property	9	FK	address_id	int(10)	NULL	Foreign key linking the property to an address.

property_amenities_map	1	FK, PK	property_id	int(10)	NOT NULL	Foreign key referencing the property table. Part of the composite primary key.
property_amenities_map	2	FK, PK	amenity_id	int(10)	NOT NULL	Foreign key referencing the amenities table. Part of the composite primary key.
property_images	1	PK, UK	image_id	int(10)	NOT NULL	Unique identifier for each property image.
property_images	2	FK	property_id	int(10)	NOT NULL	Foreign key referencing the property table.
property_images	3		image_url	varchar(255)	NULL	URL of the property image.
property_images	4		description	text(65535)	NULL	Description or caption for the image.
property_images	5		uploaded_at	timestamp(4)	NULL	Timestamp of when the image was uploaded.
property_policy	1	PK, UK	policy_id	int(10)	NOT NULL	Unique identifier for each policy.
property_policy	2		policy_type	varchar(100)	NULL	Type of the policy (e.g., "Cancellation").
property_policy	3		policy_description	text(65535)	NULL	Detailed description of the policy.
property_policy_map	1	FK, PK	property_id	int(10)	NOT NULL	Foreign key referencing the property table. Part of the composite primary key.
property_policy_map	2	FK, PK	policy_id	int(10)	NOT NULL	Foreign key referencing the property_policy table. Part of the composite primary key.
property_service	1	PK	service_id	int(10)	NOT NULL	Unique identifier for each property service.
property_service	2		service_name	varchar(255)	NULL	Name of the service provided.
property_service	3		service_description	text(65535)	NULL	Detailed description of the service.
property_service_map	1	FK, PK	property_id	int(10)	NOT NULL	Foreign key referencing the property table. Part of the composite primary key.
property_service_map	2	FK, PK	service_id	int(10)	NOT NULL	Foreign key referencing the property_service table. Part of the composite primary key.

property_utility	1	PK, UK	utility_id	int(10)	NOT NULL	Unique identifier for each utility.
property_utility	2	FK	property_id	int(10)	NULL	Foreign key referencing the property table.
property_utility	3		utility_type	varchar(100)	NULL	Type of utility (e.g., "Electricity").
property_utility	4		cost	int(10)	NULL	Cost associated with the utility.
property_utility	5		billing_period_start	date(3)	NULL	Start date of the billing period.
property_utility	6		billing_period_end	date(3)	NULL	End date of the billing period.
region	1	PK	region_id	int(10)	NOT NULL	Unique identifier for each region.
region	2		region_name	varchar(45)	NULL	Name of the region.
review	1	PK	review_id	int(10)	NOT NULL	Unique identifier for each review.
review	2	FK	user_id	int(10)	NOT NULL	Foreign key referencing the user_account table to indicate the reviewer.
review	3	FK	property_id	int(10)	NOT NULL	Foreign key referencing the property table for the reviewed property.
review	4		review_text	text(65535)	NULL	Detailed review content provided by the user.
review	5		rating	decimal(2,1)	NULL	Numerical rating given to the property (e.g., 4.5).
review	6		created_at	timestamp(4)	NULL	Timestamp of when the review was submitted.
roles	1	PK, UK	role_id	int(10)	NOT NULL	Unique identifier for each role.
roles	2		role_name	varchar(50)	NULL	Name of the role (e.g., "Admin").
social_media	1	PK	network_id	int(10)	NOT NULL	Unique identifier for each social media account.
social_media	2	FK	user_account_id	int(10)	NULL	Foreign key referencing the user_account table.
social_media	3		account_url	varchar(255)	NULL	URL to the social media account.
support_tickets	1	PK	ticket_id	int(10)	NOT NULL	Unique identifier for each support ticket.

support_tickets	2	FK	user_id	int(10)	NULL	Foreign key referencing the user_account table to associate the ticket with a user.
support_tickets	3		subject	varchar(255)	NULL	The subject or title of the support ticket.
support_tickets	4		description	text(65535)	NULL	Detailed description of the issue or query.
support_tickets	5		status	varchar(50)	NULL	Current status of the ticket (e.g., "Open", "Closed").
support_tickets	6		created_at	timestamp(4)	NULL	Timestamp of when the ticket was created.
support_tickets	7		updated_at	timestamp(4)	NULL	Timestamp of the last update to the ticket.
user_account	1	PK, UK	user_id	int(10)	NOT NULL	Unique identifier for each user.
user_account	2		first_name	varchar(100)	NULL	First name of the user.
user_account	3		last_name	varchar(100)	NULL	Last name of the user.
user_account	4		user_name	varchar(50)	NULL	Unique username chosen by the user.
user_account	5	UK	email_address	varchar(255)	NULL	Email address of the user. Must be unique.
user_account	6		hashed_password	varchar(255)	NULL	Securely stored hashed password for user authentication.
user_account	7		mobile_number	varchar(20)	NULL	User's contact number.
user_account	8		created_at	timestamp(4)	NULL	Timestamp of when the account was created.
user_account	9		updated_at	timestamp(4)	NULL	Timestamp of the last update to the account.
user_activity_log	1	PK	log_id	int(10)	NOT NULL	Unique identifier for each activity log.
user_activity_log	2	FK	user_id	int(10)	NULL	Foreign key referencing the user_account table for the user who performed the activity.
user_activity_log	3		activity_type	varchar(50)	NULL	Type of activity performed (e.g., "Login").
user_activity_log	4		activity_timestamp	timestamp(4)	NULL	Timestamp of when the activity occurred.
user_activity_log	5		activity_description	text(65535)	NULL	Detailed description of the activity.

user_language	1	FK, PK	user_id	int(10)	NOT NULL	Foreign key referencing the user_account table. Part of the composite primary key.
user_language	2	FK, PK	language_id	int(10)	NOT NULL	Foreign key referencing the language table. Part of the composite primary key.
user_message	1	PK	id	int(10)	NOT NULL	Unique identifier for each message.
user_message	2	FK	sender_id	int(10)	NULL	Foreign key referencing the user_account table for the sender.
user_message	3	FK	receiver_id	int(10)	NULL	Foreign key referencing the user_account table for the receiver.
user_message	4		message_content	text(65535)	NULL	Content of the message.
user_message	5		status	varchar(50)	NULL	Status of the message (e.g., "Read", "Unread").
user_message	6		sent_at_timestamp	timestamp(4)	NULL	Timestamp of when the message was sent.
user_notification	1	PK	notification_id	int(10)	NOT NULL	Unique identifier for each notification.
user_notification	2	FK	user_id	int(10)	NULL	Foreign key referencing the user_account table for the user receiving the notification.
user_notification	3		notification_type	varchar(50)	NULL	Type of notification (e.g., "Alert").
user_notification	4		notification_content	text(65535)	NULL	Content of the notification.
user_notification	5		sent_at	timestamp(4)	NULL	Timestamp of when the notification was sent.
user_notification	6		is_read	tinyint(3)	NULL	Boolean indicating whether the notification has been read.
user_roles	1	FK, PK	user_id	int(10)	NOT NULL	Foreign key referencing the user_account table. Part of the composite primary key.
user_roles	2	FK, PK	role_id	int(10)	NOT NULL	Foreign key referencing the roles table. Part of the composite primary key.


Roles and Languages
roles
role_id INT
role_name VARCHAR(5...
Indexes
language
language_id INT
language_name VARCHAR(20)
Indexes
user_roles
user_id INT
role_id INT
Indexes
user_language
user_id INT
language_id INT
Indexes
Geographical data
region
region_id INT
region_name VARCHAR(45)
Indexes
country
country_id INT
country_name VARCHAR(10...
country_code VARCHAR(10)
region_id INT
Indexes
city
city_id INT
city_name VARCHAR(10...
country_id INT
Indexes
address
address_id INT
address_line VARCHAR(25...
city_id INT
Indexes
Platform ticketing system
support_tickets
ticket_id INT
user_id INT
subject VARCHAR(255)
description TEXT
status VARCHAR(50)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
user_notification
notification_id INT
user_id INT
notification_type VARCHAR(50)
notification_content TEXT
sent_at TIMESTAMP
is_read TINYINT(1)
Indexes
User Account
user_account
user_id INT
first_name VARCHAR(100)
last_name VARCHAR(100)
user_name VARCHAR(50)
email_address VARCHAR(255)
hashed_password VARCHAR(255)
mobile_number VARCHAR(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
Booking Data
host
user_account_id INT
host_info TEXT
host_id INT
Indexes
review
review_id INT
user_id INT
property_id INT
review_text TEXT
rating DECIMAL(2,1)
created_at TIMESTA...
Indexes
booking
booking_id INT
user_id INT
property_id INT
booking_date TIMESTAMP
number_of_guests INT
total_price DECIMAL(10,...
status_id INT
Indexes
payment
payment_id INT
booking_id INT
payment_date TIMESTAMP
amount VARCHAR(20)
payment_method VARCHAR(50)
payment_status VARCHAR(50)
Indexes
booking_guideline
booking_guideline_id INT
booking_id INT
property_id INT
guideline_description TEXT
Indexes
booking_status_histo...
history_id INT
booking_id INT
booking_status VARCHAR(50)
changed_by INT
property_id INT
Indexes
Property Data
property
property_id INT
host_id INT
price_per_night VARCHAR(20)
max_guests INT
created_at DATETIME
updated_at DATETIME
address_id INT
Indexes
Property Polices, Amenities and Rules
cancellation_policies
policy_id INT
property_id INT
cancellation_period TIMESTA...
penalty_amount INT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
property_utility
utility_id INT
property_id INT
utility_type VARCHAR(100)
cost INT
billing_period_start DATE
billing_period_end DATE
Indexes
property_policy
policy_id INT
policy_type VARCHAR(10...
policy_description TEXT
Indexes
property_policy_map
property_id INT
policy_id INT
Indexes
property_images
image_id INT
property_id INT
image_url VARCHAR(255)
description TEXT
uploaded_at TIMESTAMP
Indexes
property_amenities...
property_id INT
amenity_id INT
Indexes
amenities
amenity_id INT
amenity_name VARCHAR(10...
amenity_description TEXT
Indexes
property_service_map
property_id INT
service_id INT
Indexes
property_service
service_id INT
service_name VARCHAR(25...
service_description TEXT
Indexes
Social and Communication
social_media
network_id INT
user_account_id INT
account_url VARCHAR(25...
Indexes
user_message
id INT
sender_id INT
receiver_id INT
message_content TEXT
status VARCHAR(50)
sent_at_timestamp TIMESTAMP
Indexes
payment_info
payment_id INT
user_id INT
card_number VARCHAR(30)
card_expiry TIMESTAMP
card_type VARCHAR(50)
billing_address VARCHAR(255)
2 more...
Indexes
user_activity_log
log_id INT
user_id INT
activity_type VARCHAR(50)
activity_timestamp TIMESTAMP
activity_description TEXT
Indexes

PROJECT: BUILD A DATAMART IN SQL

Subject code: DLBDSPBDM01

Nikolay Fedotov

Stage 2: Development Phase

Overview

1) Introduction
2) SQL create scripts and test-cases
3) Insert statements

Introduction

This project is developed in three stages and aims at creating a sample database for a hotel-operating website similar to the one implemented in Airbnb application.

This presentation covers the progress of developing the database, which is based off an Entity-Relationship diagram, that served as a blueprint for the development. Over 30 tables were created in the process, with the main tables that the application was built around being *user_account* and *property*. To make sure that the database operates as expected, several test cases were written to test the database, which has mock data inserted into it. The outcome of the tests, as well as documentation on tables that exist within the system can be found in this presentation below.

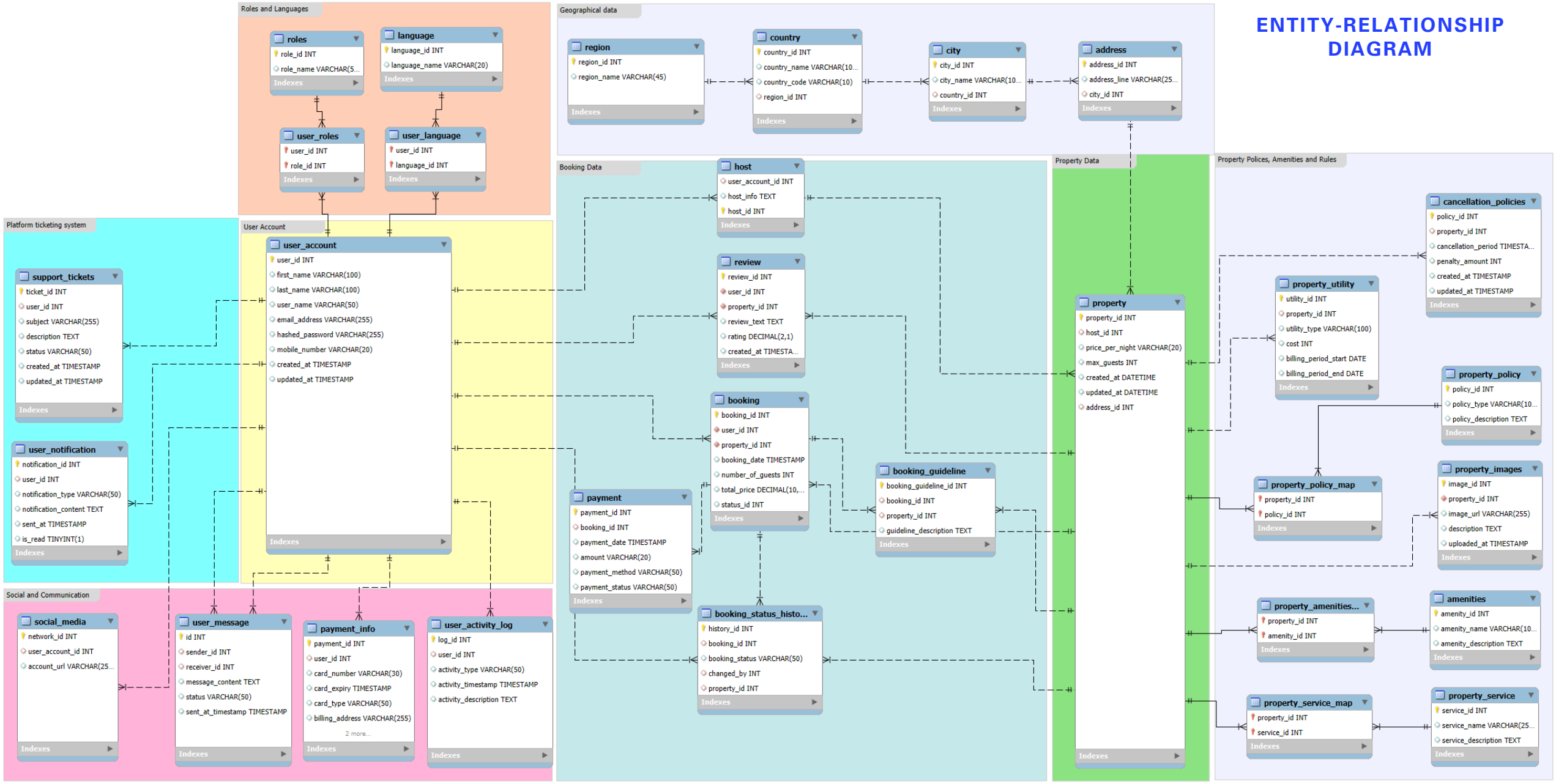
Tools used to create the project

MySQL Workbench – designing and creating the database. Mostly used for the management.

Mockaroo – tool that helped creating data for test cases and mock data.

dbForge Studio for MySQL – creating data to populate the database with.

ENTITY-RELATIONSHIP DIAGRAM



SQL scripts: Table creation and data insertion

Address Table

Create table

DDL for iu_project_header.address

```
CREATE TABLE `address` (
  `address_id` int NOT NULL,
  `address_line` varchar(45) NOT NULL,
  `city_id` int NOT NULL,
  PRIMARY KEY (`address_id`),
  KEY `city_id_idx` (`city_id`),
  CONSTRAINT `city_id` FOREIGN KEY (`city_id`) REFERENCES `city` (`city_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    a.address_id,
    a.address_line,
    c.city_name,
    co.country_name
FROM
    address a
JOIN
    city c ON a.city_id = c.city_id
JOIN
    country co ON c.country_id = co.countr
WHERE
    c.city_name = 'Toronto';
```

Result Grid | Filter Rows: | Export:

address_id	address_line	city_name	country_name
4	101 Pine St	Toronto	Canada
5	102 Oak St	Toronto	Canada

This is a query that shows all the addresses that are situated in Toronto city

Query showing data

```
SELECT * FROM address;
```

Result Grid | Filter Rows:

address_id	address_line	city_id
1	123 Main St	1
2	456 Elm St	1
3	789 Maple St	2
4	101 Pine St	3
5	102 Oak St	3
6	103 Birch St	5
7	104 Cedar St	5
8	105 Spruce St	6
9	106 Ash St	7
10	107 Willow St	7
11	108 Poplar St	9
12	109 Cherry St	9
13	110 Walnut St	11
14	111 Hickory St	11
15	112 Chestnu...	15
16	113 Fir St	15
17	114 Palm St	17
18	115 Banyan St	17
19	116 Eucalypt...	18

Amenities Table

Create table

DDL for iu_project_header.amenities

```
CREATE TABLE `amenities` (
  `amenity_id` int NOT NULL AUTO_INCREMENT,
  `amenity_name` varchar(100) DEFAULT NULL,
  `amenity_description` text,
  PRIMARY KEY (`amenity_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    a.amenity_name,
    a.amenity_description
FROM
    property_amenities_map pam
JOIN
    amenities a ON pam.amenity_id = a.amenity_id
WHERE
    pam.property_id = 6;
```

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

amenity_name	amenity_description
Dryer	In-unit clothes dryer
Dishwasher	Dishwasher in kitchen

This query shows the amenities that can be found in property_id 6

Query showing data

Limit to 1000 rows

```
SELECT * FROM amenities;
```

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

amenity_id	amenity_name	amenity_description
1	Wi-Fi	High-speed wireless internet
2	Air Conditioning	Central air conditioning
3	Heating	Central heating system
4	TV	Flat-screen TV with cable channels
5	Kitchen	Fully equipped modern kitchen
6	Pool	Outdoor swimming pool
7	Gym	In-house fitness center
8	Parking	Private parking available
9	Balcony	Spacious private balcony
10	Garden	Well-maintained garden area
11	Washer	In-unit washer and dryer
12	Dryer	In-unit clothes dryer
13	Dishwasher	Dishwasher in kitchen
14	BBQ Grill	Outdoor BBQ grill
15	Fireplace	Indoor fireplace
16	Hot Tub	Outdoor hot tub
17	Game Room	Game room with entertainment opt...
18	Playground	Outdoor playground for kids
19	Sauna	Private sauna
20	Library	Private library with books and mag...
NULL	NULL	NULL

Booking Guideline Table

Create table

DDL for iu_project_header.booking_guideline

```
CREATE TABLE `booking_guideline` (
  `booking_guideline_id` int NOT NULL,
  `booking_id` int unsigned DEFAULT NULL,
  `property_id` int DEFAULT NULL,
  `guideline_description` text,
  PRIMARY KEY (`booking_guideline_id`),
  KEY `booking_id` (`booking_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `booking_guideline_ibfk_1` FOREIGN KEY (`booking_id`) REFERENCES `booking` (`booking_id`),
  CONSTRAINT `booking_guideline_ibfk_2` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    bg.booking_guideline_id,
    bg.guideline_description,
    bg.booking_id,
    bg.property_id
FROM
    booking_guideline bg
WHERE
    bg.property_id = 4;
```

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

booking_guideline_id	guideline_description	booking_id	property_id
4	Quiet hours from 10 PM to 7 AM.	4	4
NULL	NULL	NULL	NULL

This query shows what guideline does property 4 have

Query showing data

```
SELECT * FROM booking_guideline;
```

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

booking_guideline_id	booking_id	property_id	guideline_description
1	1	1	Check-in after 3 PM.
2	2	2	No smoking in the property.
3	3	3	No pets allowed.
4	4	4	Quiet hours from 10 PM to 7 AM.
5	5	5	Pool use allowed from 9 AM to 9 PM.
6	6	6	ID required at check-in.
7	7	7	Maximum of 4 guests per booking.
8	8	8	No parties or events allowed.
9	9	9	Parking space available upon request.
10	10	10	Please dispose of garbage in designated bins.
11	11	11	Early check-out available upon request.
12	12	12	Pets allowed with a fee of $50.
13	13	13	Please report damages immediately.
14	14	14	Use of the gym is complimentary.
15	15	15	Breakfast is served from 7 AM to 10 AM.
16	16	16	Check-out before 11 AM.
17	17	17	Use of hot tub allowed until midnight.
18	18	18	Smoking allowed in designated areas only.
19	19	19	Complimentary toiletries provided.
20	20	20	Please follow local COVID-19 guidelines.
NULL	NULL	NULL	NULL

Booking Table

Create table

DDL for iu_project_header.booking

```
CREATE TABLE `booking` (
  `booking_id` int unsigned NOT NULL,
  `user_id` int unsigned NOT NULL,
  `property_id` int NOT NULL,
  `booking_date` timestamp NULL DEFAULT NULL,
  `number_of_guests` int DEFAULT NULL,
  `total_price` decimal(10,2) DEFAULT NULL,
  `status_id` int DEFAULT NULL,
  PRIMARY KEY (`booking_id`),
  KEY `user_id` (`user_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `booking_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `booking_ibfk_2` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    b.booking_id,
    b.property_id,
    b.booking_date,
    b.number_of_guests,
    b.total_price,
    b.status_id
FROM
    booking b
WHERE
    b.user_id = 5;
```

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

booking_id	property_id	booking_date	number_of_guests	total_price	status_id
5	5	2023-12-05 14:00:00	4	1000.00	1
NULL	NULL	NULL	NULL	NULL	NULL

Query showing the booking that user_id number 5 has

Query showing data

Limit to 1000 rows

```
SELECT * FROM booking;
```

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

booking_id	user_id	property_id	booking_date	number_of_guests	total_price	status_id
1	1	1	2023-12-01 10:00:00	4	800.00	1
2	2	2	2023-12-02 11:00:00	2	700.00	2
3	3	3	2023-12-03 12:00:00	3	1050.00	1
4	4	4	2023-12-04 13:00:00	5	1200.00	3
5	5	5	2023-12-05 14:00:00	4	1000.00	1
6	6	6	2023-12-06 15:00:00	6	1500.00	2
7	7	7	2023-12-07 16:00:00	3	900.00	3
8	8	8	2023-12-08 17:00:00	2	750.00	1
9	9	9	2023-12-09 18:00:00	4	1100.00	1
10	10	10	2023-12-10 19:00:00	2	800.00	3
11	11	11	2023-12-11 20:00:00	5	1300.00	2
12	12	12	2023-12-12 21:00:00	3	950.00	1
13	13	13	2023-12-13 22:00:00	2	700.00	3
14	14	14	2023-12-14 23:00:00	4	1150.00	1
15	15	15	2023-12-15 08:00:00	3	900.00	2
16	16	16	2023-12-16 09:00:00	2	650.00	3
17	17	17	2023-12-17 10:00:00	5	1250.00	1
18	18	18	2023-12-18 11:00:00	4	1000.00	3
19	19	19	2023-12-19 12:00:00	6	1400.00	2
20	20	20	2023-12-20 13:00:00	3	850.00	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Booking Status History Table

Create table

DDL for iu_project_header.booking_status_history

```
CREATE TABLE `booking_status_history` (
  `history_id` int NOT NULL,
  `booking_id` int unsigned DEFAULT NULL,
  `booking_status` varchar(50) DEFAULT NULL,
  `changed_by` int unsigned DEFAULT NULL,
  `property_id` int DEFAULT NULL,
  PRIMARY KEY (`history_id`),
  KEY `changed_by` (`changed_by`),
  KEY `booking_id` (`booking_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `booking_status_history_ibfk_1` FOREIGN KEY (`changed_by`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `booking_status_history_ibfk_2` FOREIGN KEY (`booking_id`) REFERENCES `booking` (`booking_id`),
  CONSTRAINT `booking_status_history_ibfk_3` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    bsh.history_id,
    bsh.booking_id,
    bsh.booking_status,
    bsh.changed_by,
    b.property_id
FROM
    booking_status_history bsh
JOIN
    booking b ON bsh.booking_id = b.booking_id
WHERE
    b.property_id = 11;
```

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

history_id	booking_id	booking_status	changed_by	property_id
11	11	Cancelled	11	11

Here is a query that finds the booking status history of property number 11. Therefore, we can see that the booking number 11 for property number was cancelled by user_id 11.

Query showing data

```
SELECT * FROM booking_status_history;
```

Result Grid | Filter Rows: | Edit:

history_id	booking_id	booking_status	changed_by	property_id
1	1	Confirmed	1	1
2	2	Cancelled	2	2
3	3	Pending	3	3
4	4	Confirmed	4	4
5	5	Cancelled	5	5
6	6	Pending	6	6
7	7	Confirmed	7	7
8	8	Cancelled	8	8
9	9	Confirmed	9	9
10	10	Pending	10	10
11	11	Cancelled	11	11
12	12	Confirmed	12	12
13	13	Pending	13	13
14	14	Cancelled	14	14
15	15	Confirmed	15	15
16	16	Pending	16	16
17	17	Cancelled	17	17
18	18	Confirmed	18	18
19	19	Pending	19	19
20	20	Confirmed	20	20
NULL	NULL	NULL	NULL	NULL

Cancellation Policies Table

Create table

DDL for iu_project_header.cancellation_policies

```sql
CREATE TABLE `cancellation_policies` (
  `policy_id` int unsigned NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `cancellation_period` timestamp NULL DEFAULT NULL,
  `penalty_amount` int DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`policy_id`),
  UNIQUE KEY `policy_id` (`policy_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `cancellation_policies_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

Limit to 1000 rows

```sql
SELECT * FROM cancellation_policies;
```

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

policy_id	property_id	cancellation_period	penalty_amount	created_at	updated_at
1	1	2024-01-01 10:00:00	50	2023-12-01 12:00:00	2023-12-01 12:00:00
2	2	2024-01-02 10:00:00	75	2023-12-02 12:00:00	2023-12-02 12:00:00
3	3	2024-01-03 10:00:00	100	2023-12-03 12:00:00	2023-12-03 12:00:00
4	4	2024-01-04 10:00:00	125	2023-12-04 12:00:00	2023-12-04 12:00:00
5	5	2024-01-05 10:00:00	150	2023-12-05 12:00:00	2023-12-05 12:00:00
6	6	2024-01-06 10:00:00	175	2023-12-06 12:00:00	2023-12-06 12:00:00
7	7	2024-01-07 10:00:00	200	2023-12-07 12:00:00	2023-12-07 12:00:00
8	8	2024-01-08 10:00:00	225	2023-12-08 12:00:00	2023-12-08 12:00:00
9	9	2024-01-09 10:00:00	250	2023-12-09 12:00:00	2023-12-09 12:00:00
10	10	2024-01-10 10:00:00	275	2023-12-10 12:00:00	2023-12-10 12:00:00
11	11	2024-01-11 10:00:00	300	2023-12-11 12:00:00	2023-12-11 12:00:00
12	12	2024-01-12 10:00:00	325	2023-12-12 12:00:00	2023-12-12 12:00:00
13	13	2024-01-13 10:00:00	350	2023-12-13 12:00:00	2023-12-13 12:00:00
14	14	2024-01-14 10:00:00	375	2023-12-14 12:00:00	2023-12-14 12:00:00
15	15	2024-01-15 10:00:00	400	2023-12-15 12:00:00	2023-12-15 12:00:00
16	16	2024-01-16 10:00:00	425	2023-12-16 12:00:00	2023-12-16 12:00:00

```sql
SELECT
    cp.policy_id,
    cp.cancellation_period,
    cp.penalty_amount,
    cp.created_at,
    cp.updated_at
FROM
    cancellation_policies cp
WHERE
    cp.property_id = 7;
```

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

policy_id	cancellation_period	penalty_amount	created_at	updated_at
7	2024-01-07 10:00:00	200	2023-12-07 12:00:00	2023-12-07 12:00:00
NULL	NULL	NULL	NULL	NULL

Here we can see the cancellation policy of property 7, which was created in year 2023 for year 2024 with a possible penalty of 200

City Table

Create table

DDL for iu_project_header.city

```sql
CREATE TABLE `city` (
  `city_id` int NOT NULL AUTO_INCREMENT,
  `city_name` varchar(100) DEFAULT NULL,
  `country_id` int DEFAULT NULL,
  PRIMARY KEY (`city_id`),
  KEY `country_id_idx` (`country_id`),
  CONSTRAINT `country_id` FOREIGN KEY (`country_id`) REFERENCES `country` (`country_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data with cities, countries and regions

```sql
SELECT
    c.city_id,
    c.city_name,
    ctr.country_name,
    ctr.country_code,
    r.region_name
FROM
    city c
JOIN
    country ctr ON c.country_id = ctr.country_id
JOIN
    region r ON ctr.region_id = r.region_id;
```

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

city_id	city_name	country_name	country_code	region_name
1	New York	United States	US	North America
2	Los Angeles	United States	US	North America
3	Toronto	Canada	CA	North America
4	Vancouver	Canada	CA	North America
5	Berlin	Germany	DE	Europe
6	Munich	Germany	DE	Europe
7	Paris	France	FR	Europe
8	Lyon	France	FR	Europe
9	Beijing	China	CN	Asia
10	Shanghai	China	CN	Asia
11	Tokyo	Japan	JP	Asia
12	Osaka	Japan	JP	Asia
13	Mumbai	India	IN	Asia
14	Delhi	India	IN	Asia
15	Lagos	Nigeria	NG	Africa
16	Abuja	Nigeria	NG	Africa
17	Rio de Jan...	Brazil	BR	South America
18	São Paulo	Brazil	BR	South America
19	Sydney	Australia	AU	Australia
20	Melbourne	Australia	AU	Australia

Country Table

Create table

DDL for iu_project_header.country

```
CREATE TABLE `country` (
  `country_id` int NOT NULL AUTO_INCREMENT,
  `country_name` varchar(100) DEFAULT NULL,
  `country_code` varchar(10) DEFAULT NULL,
  `region_id` int DEFAULT NULL,
  PRIMARY KEY (`country_id`),
  KEY `region_id_idx` (`region_id`),
  CONSTRAINT `region_id` FOREIGN KEY (`region_id`) REFERENCES `region` (`region_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT c.city_name
FROM city c
JOIN country co ON c.country_id = co.country_id
JOIN region r ON co.region_id = r.region_id
WHERE r.region_name = 'Asia';
```

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

city_name
Beijing
Shanghai
Tokyo
Osaka
Mumbai
Delhi

A query that shows all cities that exist in the table that are in Asia region

Query showing data

Limit to

```
SELECT * FROM country;
```

Result Grid | Filter Rows: | Edit:

country_id	country_name	country_code	region_id
1	United States	US	1
2	Canada	CA	1
3	Germany	DE	2
4	France	FR	2
5	China	CN	3
6	Japan	JP	3
7	India	IN	3
8	Nigeria	NG	4
9	Brazil	BR	5
10	Argentina	AR	5
11	Australia	AU	6
12	Russia	RU	3
13	United Kingdom	UK	2
14	Italy	IT	2
15	Spain	ES	2
16	Mexico	MX	9

Host Table

Create table

DDL for iu_project_header.host

```
CREATE TABLE `host` (
  `user_account_id` int unsigned DEFAULT NULL,
  `host_info` text,
  `host_id` int unsigned NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`host_id`),
  KEY `user_account_id` (`user_account_id`),
  CONSTRAINT `host_ibfk_1` FOREIGN KEY (`user_account_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1001 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    h.host_id,
    h.user_account_id,
    p.property_id,
    p.location
FROM
    host h
JOIN
    property p ON h.host_id = p.host_id
WHERE
    h.host_id = 4;
```

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

host_id	user_account_id	property_id	location
4	4	4	101 Maple Ave, Toronto, ON

The command shows the user id and property id, as well as its location, for host_id

Query showing data

Limit to 1000 rows

```
SELECT * FROM host;
```

Result Grid | Filter Rows: | Edit: | Expo

user_account_id	host_info	host_id
1	Experienced host with 5 years in the hospitality ...	1
2	New host specializing in unique vacation homes	2
3	Local expert offering city tours along with acco...	3
4	Friendly host with multiple beachfront properties	4
5	Luxury host catering to high-end clients	5
6	Eco-friendly host with sustainable properties	6
7	Pet-friendly host with dedicated pet amenities	7
8	Host offering family-friendly accommodations	8
9	Experienced host managing properties in urban ...	9
10	Host specializing in long-term rental stays	10
11	Adventure host offering properties in remote ar...	11
12	Artistic host with uniquely decorated homes	12
13	Host focused on wellness retreats and spa servi...	13
14	Corporate host offering business-ready accom...	14
15	Student-focused host near educational institutions	15
16	Couples-focused host with romantic getaway o...	16
17	Tech-savvy host with smart home properties	17
18	Senior-friendly host offering accessible accomm...	18
19	Gourmet host with properties near culinary hots...	19
20	Seasoned host with expertise in cultural tourism	20
NULL	NULL	NULL

Language Table

Create table

DDL for new_schema.language

```
1  CREATE TABLE `language` (
2    `language_id` int unsigned NOT NULL AUTO_INCREMENT,
3    `language_name` varchar(20) DEFAULT NULL,
4    PRIMARY KEY (`language_id`),
5    UNIQUE KEY `language_id` (`language_id`)
6  ) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
1  SELECT
2      ua.user_id,
3      ua.first_name,
4      ua.last_name,
5      l.language_name
6  FROM
7      user_account ua
8  JOIN
9      user_language ul ON ua.user_id = ul.user_id
10 JOIN
11     language l ON ul.language_id = l.language_id
```

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

user_id	first_name	last_name	language_name
1	Carilyn	Jezard	Eng
1	Carilyn	Jezard	Spa
2	Rani	Goodhay	Fre
2	Rani	Goodhay	Ger
3	Julita	Elt	Chi
4	Meir	Davids	Jap
5	Sawyer	Cavill	Rus
6	Dan	Ledgerton	Ita
7	D'arcy	Vivien	Por
8	Lemuel	Jedraszczyk	Kor
9	Josias	Ohms	Hin
10	Viva	Houtby	Ara
11	Cathleen	Laydon	Dut
12	Sayers	Batchellor	Swe
13	Allan	Mackett	Nor
14	Naomi	Nussii	Tur
15	Vyky	Murrigans	Pol
16	Harriette	Kienle	Dan
17	Yul	Castle	Fin
18	Eleanora	Portingale	Gre

Query that shows all the languages for every user_id from user_account

Query showing data

```
1  SELECT * FROM language;
```

Result Grid | Filter Rows:

language_id	language_name
1	Eng
2	Spa
3	Fre
4	Ger
5	Chi
6	Jap
7	Rus
8	Ita
9	Por
10	Kor
11	Hin
12	Ara
13	Dut
14	Swe
15	Nor
16	Tur
17	Pol
18	Dan
19	Fin
20	Gre
NULL	NULL

Payment Table

Create table

DDL for iu_project_header.payment

```sql
CREATE TABLE `payment` (
  `payment_id` int NOT NULL,
  `booking_id` int unsigned DEFAULT NULL,
  `payment_date` timestamp NULL DEFAULT NULL,
  `amount` varchar(20) DEFAULT NULL,
  `payment_method` varchar(50) DEFAULT NULL,
  `payment_status` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`payment_id`),
  KEY `booking_id` (`booking_id`),
  CONSTRAINT `payment_ibfk_1` FOREIGN KEY (`booking_id`) REFERENCES `booking` (`booking_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```sql
SELECT
    p.amount,
    p.payment_method,
    p.payment_status,
    b.booking_id,
    b.user_id,
    b.property_id,
    b.total_price
FROM
    payment p
JOIN
    booking b ON p.booking_id = b.booking_id;
```

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

amount	payment_method	payment_status	booking_id	user_id	property_id	total_price
800	Credit Card	Completed	1	1	1	800.00
700	PayPal	Completed	2	2	2	700.00
1050	Bank Transfer	Pending	3	3	3	1050.00
1200	Credit Card	Completed	4	4	4	1200.00
1000	PayPal	Failed	5	5	5	1000.00
1500	Credit Card	Completed	6	6	6	1500.00
900	PayPal	Pending	7	7	7	900.00
750	Bank Transfer	Completed	8	8	8	750.00
1100	Credit Card	Completed	9	9	9	1100.00
800	PayPal	Completed	10	10	10	800.00
1300	Credit Card	Pending	11	11	11	1300.00
950	PayPal	Completed	12	12	12	950.00
700	Bank Transfer	Failed	13	13	13	700.00
1150	Credit Card	Completed	14	14	14	1150.00
900	PayPal	Completed	15	15	15	900.00
650	Credit Card	Completed	16	16	16	650.00
1250	Bank Transfer	Pending	17	17	17	1250.00
1000	Credit Card	Completed	18	18	18	1000.00
1400	PayPal	Pending	19	19	19	1400.00
850	Credit Card	Completed	20	20	20	850.00

Query that shows the data from payment and booking tables for each booking.

Query showing data

Limit to 1000 rows

```sql
SELECT * FROM payment;
```

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

payment_id	booking_id	payment_date	amount	payment_method	payment_status
1	1	2023-12-01 11:00:00	800	Credit Card	Completed
2	2	2023-12-02 12:00:00	700	PayPal	Completed
3	3	2023-12-03 13:00:00	1050	Bank Transfer	Pending
4	4	2023-12-04 14:00:00	1200	Credit Card	Completed
5	5	2023-12-05 15:00:00	1000	PayPal	Failed
6	6	2023-12-06 16:00:00	1500	Credit Card	Completed
7	7	2023-12-07 17:00:00	900	PayPal	Pending
8	8	2023-12-08 18:00:00	750	Bank Transfer	Completed
9	9	2023-12-09 19:00:00	1100	Credit Card	Completed
10	10	2023-12-10 20:00:00	800	PayPal	Completed
11	11	2023-12-11 21:00:00	1300	Credit Card	Pending
12	12	2023-12-12 22:00:00	950	PayPal	Completed
13	13	2023-12-13 23:00:00	700	Bank Transfer	Failed
14	14	2023-12-14 08:00:00	1150	Credit Card	Completed
15	15	2023-12-15 09:00:00	900	PayPal	Completed
16	16	2023-12-16 10:00:00	650	Credit Card	Completed
17	17	2023-12-17 11:00:00	1250	Bank Transfer	Pending
18	18	2023-12-18 12:00:00	1000	Credit Card	Completed
19	19	2023-12-19 13:00:00	1400	PayPal	Pending
20	20	2023-12-20 14:00:00	850	Credit Card	Completed
NULL	NULL	NULL	NULL	NULL	NULL

Payment Info Table

Create table

DDL for iu_project_header.payment_info

```sql
CREATE TABLE `payment_info` (
  `payment_id` int NOT NULL,
  `user_id` int unsigned DEFAULT NULL,
  `card_number` varchar(30) DEFAULT NULL,
  `card_expiry` timestamp NULL DEFAULT NULL,
  `card_type` varchar(50) DEFAULT NULL,
  `billing_address` varchar(255) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `method_type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`payment_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `payment_info_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```sql
SELECT * FROM payment_info;
```

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

payment_id	user_id	card_number	card_expiry	card_type	billing_address	created_at	method_type
1	12	5602255230400716	2027-11-17 00:00:00	bankcard	72791 Claremont Road	2024-07-21 00:00:00	apple_pay
2	3	3569815018863462	2025-01-10 00:00:00	jcb	5 Elka Circle	2023-11-30 00:00:00	venmo
3	11	3568529170833269	2026-02-14 00:00:00	jcb	3 Onsgard Alley	2023-12-15 00:00:00	debit_card
4	8	4026705996755944	2025-02-10 00:00:00	visa-electron	5920 Cascade Alley	2023-12-01 00:00:00	credit_card
5	14	490505116813608483	2028-10-17 00:00:00	switch	40 Stone Corner Way	2024-06-13 00:00:00	venmo
6	7	3532767912031866	2028-01-06 00:00:00	jcb	3 Lakewood Lane	2024-11-20 00:00:00	credit_card
7	5	3557048955841655	2024-08-28 00:00:00	jcb	8 Dapin Place	2024-05-12 00:00:00	credit_card
8	18	3549316920398619	2026-02-01 00:00:00	jcb	5243 Superior Place	2024-11-16 00:00:00	paypal
9	11	201525200798513	2028-09-16 00:00:00	diners-club-enroute	67 Summerview Way	2024-11-27 00:00:00	paypal
10	19	3560540743289671	2027-12-17 00:00:00	jcb	790 Fulton Avenue	2023-11-29 00:00:00	credit_card
11	16	3551428148820061	2027-05-15 00:00:00	jcb	5449 Division Lane	2024-05-20 00:00:00	debit_card
12	10	3564284526090263	2028-05-27 00:00:00	jcb	28873 Northridge Point	2023-12-24 00:00:00	apple_pay
13	9	3574145023038287	2026-04-08 00:00:00	jcb	5 Spaight Way	2023-12-28 00:00:00	venmo
14	14	5379544350333653	2027-03-29 00:00:00	mastercard	9604 Linden Junction	2024-01-26 00:00:00	paypal
15	15	589302717678188497	2024-08-08 00:00:00	maestro	5 Jenifer Place	2024-06-27 00:00:00	debit_card
16	6	3532576987969201	2028-08-18 00:00:00	jcb	02 Nevada Terrace	2024-02-02 00:00:00	credit_card
17	1	560222159832685464	2024-11-29 00:00:00	china-unionpay	9824 Wayridge Terrace	2024-08-07 00:00:00	apple_pay
18	4	677128397533821 66	2028-09-10 00:00:00	laser	6773 Lakewood Garde...	2024-11-23 00:00:00	venmo
19	20	3560152240238688	2026-05-23 00:00:00	jcb	41 Reinke Trail	2024-05-16 00:00:00	paypal
20	2	3548359073752894	2026-08-22 00:00:00	jcb	1 Hovde Junction	2024-10-12 00:00:00	paypal
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```sql
SELECT
pi.payment_id,
pi.user_id,
pi.card_number,
ua.user_id,
ua.first_name
FROM
payment_info pi
JOIN
user_account ua ON pi.user_id = ua.user_id;
```

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

payment_id	user_id	card_number	user_id	first_name
1	12	5602255230400716	12	Sayers
2	3	3569815018863462	3	Julita
3	11	3568529170833269	11	Cathleen
4	8	4026705996755944	8	Lemuel
5	14	490505116813608483	14	Naomi
6	7	3532767912031866	7	Darcy
7	5	3557048955841655	5	Sawyer
8	18	3549316920398619	18	Eleanora
9	11	201525200798513	11	Cathleen
10	19	3560540743289671	19	Sargent
11	16	3551428148820061	16	Harriette
12	10	3564284526090263	10	Viva
13	9	3574145023038287	9	Josias
14	14	5379544350333653	14	Naomi
15	15	589302717678188497	15	Vyky
16	6	3532576987969201	6	Dan
17	1	560222159832685464	1	Carilyn
18	4	677128397533821 66	4	Meir
19	20	3560152240238688	20	Filia
20	2	3548359073752894	2	Rani

Query that returns all information from the user_account and payment_info tables about the payments belonging to users

Property Table

Create table

DDL for iu_project_header.property

```
CREATE TABLE `property` (
  `property_id` int NOT NULL,
  `host_id` int unsigned DEFAULT NULL,
  `price_per_night` varchar(20) DEFAULT NULL,
  `max_guests` int DEFAULT NULL,
  `created_at` datetime DEFAULT NULL,
  `updated_at` datetime DEFAULT NULL,
  `address_id` int DEFAULT NULL,
  PRIMARY KEY (`property_id`),
  KEY `fk_property_address` (`address_id`),
  KEY `fk_property_host` (`host_id`),
  CONSTRAINT `fk_property_address` FOREIGN KEY (`address_id`) REFERENCES `address` (`address_id`),
  CONSTRAINT `fk_property_host` FOREIGN KEY (`host_id`) REFERENCES `host` (`host_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing all property data

```
SELECT * FROM property;
```

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

property_id	host_id	price_per_night	max_guests	created_at	updated_at	address_id
4	19	500	9	2023-11-19 12:08:45	2023-11-19 12:09:19	19
12	20	450	8	2023-11-20 13:10:45	2023-11-20 13:12:12	20
5	5	250	5	2023-11-05 14:04:19	2023-11-05 14:05:53	5
10	18	350	7	2023-11-18 11:06:34	2023-11-18 11:07:28	18
19	17	300	6	2023-11-17 10:02:54	2023-11-17 10:04:23	17
15	10	220	6	2023-11-10 19:07:45	2023-11-10 19:09:15	10
18	11	280	5	2023-11-11 20:05:39	2023-11-11 20:06:52	11
9	8	350	5	2023-11-08 17:09:53	2023-11-08 17:10:47	8
16	9	300	4	2023-11-09 18:03:12	2023-11-09 18:04:58	9
17	15	200	6	2023-11-15 08:13:27	2023-11-15 08:14:41	15
20	16	250	8	2023-11-16 09:08:12	2023-11-16 09:09:48	16
6	14	380	5	2023-11-14 23:07:32	2023-11-14 23:08:48	14
13	13	450	7	2023-11-13 22:09:31	2023-11-13 22:11:04	13
2	12	600	8	2023-11-12 21:05:39	2023-11-12 21:06:42	12
1	4	400	6	2023-11-06 15:13:27	2023-11-06 15:14:51	6
3	7	180	4	2023-11-07 16:11:23	2023-11-07 16:12:56	7
14	6	600	6	2023-11-06 15:04:18	2023-11-06 15:05:21	6
7	3	500	8	2023-11-03 12:02:14	2023-11-03 12:03:59	3
8	1	200	4	2023-11-01 10:15:42	2023-11-01 10:16:36	1
11	2	350	6	2023-11-02 11:14:27	2023-11-02 11:15:43	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Queries showing complex property data with joined statements are in other slides

Property Amenities Map Table

Create table

DDL for iu_project_header.property_amenities_map

```
CREATE TABLE `property_amenities_map` (
  `property_id` int NOT NULL,
  `amenity_id` int NOT NULL,
  PRIMARY KEY (`property_id`,`amenity_id`),
  KEY `amenity_id` (`amenity_id`),
  CONSTRAINT `property_amenities_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`),
  CONSTRAINT `property_amenities_ibfk_2` FOREIGN KEY (`amenity_id`) REFERENCES `amenities` (`amenity_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
SELECT
    a.amenity_name,
    a.amenity_description
FROM
    property_amenities_map pam
JOIN
    amenities a ON pam.amenity_id = a.amenity_id
WHERE
    pam.property_id = 6;
```

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

amenity_name	amenity_description
Dryer	In-unit clothes dryer
Dishwasher	Dishwasher in kitchen

This query shows the amenities that can be found in property_id 6

Query showing data

```
SELECT * FROM property_amenities_map;
```

Result Grid | Filter Rows: | Edit:

property_id	amenity_id
1	1
1	2
1	3
2	4
2	5
3	6
3	7
4	8
4	9
5	10
5	11
6	12
6	13
7	14
7	15
8	16
8	17
9	18
9	19
10	20
NULL	NULL

Property Images Table

Create table

DDL for iu_project_header.property_images

```
CREATE TABLE `property_images` (
  `image_id` int unsigned NOT NULL AUTO_INCREMENT,
  `property_id` int NOT NULL,
  `image_url` varchar(255) DEFAULT NULL,
  `description` text,
  `uploaded_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`image_id`),
  UNIQUE KEY `image_id` (`image_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `property_images_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data on properties and property_images, showing both together

```
SELECT
    p.property_id,
    p.country_code,
    pi.description,
    pi.image_url
FROM
    property_images pi
JOIN
    property p ON p.property_id = pi.property_id;
```

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

property_id	country_code	description	image_url
1	US	Front view of the property	https://example.com/images/property1.jpg
1	US	Living room	https://example.com/images/property1_interior...
2	US	Side view of the property	https://example.com/images/property2.jpg
2	US	Modern kitchen	https://example.com/images/property2_kitchen...
3	US	Backyard	https://example.com/images/property3.jpg
3	US	Master bedroom	https://example.com/images/property3_bedroo...
4	CA	Pool area	https://example.com/images/property4.jpg
4	CA	Balcony with a view	https://example.com/images/property4_balcon...
5	CA	Garden area	https://example.com/images/property5.jpg
5	CA	Spacious living room	https://example.com/images/property5_living.jpg
6	UK	Night view	https://example.com/images/property6.jpg
6	UK	Dining area	https://example.com/images/property6_dining.jpg
7	UK	Terrace	https://example.com/images/property7.jpg
7	UK	In-house gym	https://example.com/images/property7_gym.jpg
8	FR	Patio	https://example.com/images/property8.jpg
8	FR	Luxury bathroom	https://example.com/images/property8_bathro...

Property Policy Table

Create table

DDL for iu_project_header.property_policy

```
CREATE TABLE `property_policy` (
  `policy_id` int unsigned NOT NULL AUTO_INCREMENT,
  `policy_type` varchar(100) DEFAULT NULL,
  `policy_description` text,
  PRIMARY KEY (`policy_id`),
  UNIQUE KEY `policy_id` (`policy_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from property and property policy, including descriptions and property_id

```
SELECT
    pp.policy_id,
    pp.policy_description,
    p.property_id
FROM
    property_policy pp
JOIN
    property_policy_map ppm ON pp.policy_id = ppm.policy_id
JOIN
    property p ON ppm.property_id = p.property_id;
```

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

policy_id	policy_description	property_id
1	Full refund if canceled within 24 hours of booking	1
2	Damage to property incurs a $500 fine	1
3	Check-in from 3 PM to 10 PM	1
4	Check-out before 11 AM	2
5	Pets allowed with a $50 cleaning fee	2
6	No smoking allowed in the property	3
7	Maximum of 4 guests allowed	3
8	Quiet hours from 10 PM to 7 AM	3
9	Lost keys incur a $100 replacement fee	4
10	Free parking available on premises	4
11	Cleaning fee of $100 applies	5
12	50% refund if canceled 7 days before arrival	5
13	No refund for intentional damage	6
14	Check-in from 12 PM to 8 PM	6
15	Check-out before 10 AM	7
16	No pets allowed	8
17	Smoking allowed in designated areas only	8
18	Maximum of 6 guests allowed	9

Property Policy Map Table

Create table

DDL for iu_project_header.property_policy_map

```
CREATE TABLE `property_policy_map` (
  `property_id` int NOT NULL,
  `policy_id` int unsigned NOT NULL,
  PRIMARY KEY (`property_id`,`policy_id`),
  KEY `policy_id` (`policy_id`),
  CONSTRAINT `property_policy_map_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`),
  CONSTRAINT `property_policy_map_ibfk_2` FOREIGN KEY (`policy_id`) REFERENCES `property_policy` (`policy_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```
SELECT * FROM property_policy_map;
```

Result Grid | Filter Rows: | Edit:

property_id	policy_id
1	1
1	2
1	3
2	4
2	5
3	6
3	7
3	8
4	9
4	10
5	11
5	12
6	13
6	14
7	15
8	16
8	17
9	18
10	19
10	20
NULL	NULL

Specific query can be found in property policy slide

Property Service Table

Create table

DDL for iu_project_header.property_service

```
CREATE TABLE `property_service` (
  `service_id` int NOT NULL,
  `service_name` varchar(255) DEFAULT NULL,
  `service_description` text,
  PRIMARY KEY (`service_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from property_id and property service showing policies for properties

```
SELECT
    p.property_id,
    ps.service_id,
    ps.service_name,
    ps.service_description
FROM
    property p
JOIN
    property_service_map psm ON p.property_id = psm.property_id
JOIN
    property_service ps ON psm.service_id = ps.service_id;
```

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

property_id	service_id	service_name	service_description
1	1	Daily Cleaning	Daily cleaning services provided
1	2	Concierge	24/7 concierge service
1	3	Room Service	In-room dining available
2	4	Laundry	Laundry services provided
2	5	Luggage Storage	Secure luggage storage facility
3	6	Shuttle Service	Airport and local shuttle service
3	7	Spa	On-site spa services
4	8	Grocery Delivery	Grocery delivery service available
4	9	Pet Sitting	Pet sitting services available
5	10	Babysitting	Professional babysitting services
5	11	Tour Assistance	Guided tours and assistance avai...
6	12	Breakfast Included	Complimentary breakfast for gue...
6	13	Catering	Event catering services available
7	14	Valet Parking	Valet parking services
7	15	Fitness Classes	On-site fitness classes
8	16	Yoga Sessions	Private yoga sessions available
8	17	Car Rental	On-site car rental services
9	18	Bike Rental	Bicycles available for rent
9	19	Massage	In-room massage services available
10	20	Event Planning	Event planning and coordination ...

Property Service Map Table

Create table

DDL for iu_project_header.property_service_map

```
CREATE TABLE `property_service_map` (
  `property_id` int NOT NULL,
  `service_id` int NOT NULL,
  PRIMARY KEY (`property_id`,`service_id`),
  KEY `service_id` (`service_id`),
  CONSTRAINT `property_service_map_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`),
  CONSTRAINT `property_service_map_ibfk_2` FOREIGN KEY (`service_id`) REFERENCES `property_service` (`service_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```
SELECT * FROM property_service_map;
```

Result Grid | Filter Rows: | Edit:

property_id	service_id
1	1
1	2
1	3
2	4
2	5
3	6
3	7
4	8
4	9
5	10
5	11
6	12
6	13
7	14
7	15
8	16
8	17
9	18
9	19
10	20
NULL	NULL

Specific query can be found in property service slide

Property Utility Table

Create table

DDL for iu_project_header.property_utility

```
CREATE TABLE `property_utility` (
  `utility_id` int unsigned NOT NULL AUTO_INCREMENT,
  `property_id` int DEFAULT NULL,
  `utility_type` varchar(100) DEFAULT NULL,
  `cost` int DEFAULT NULL,
  `billing_period_start` date DEFAULT NULL,
  `billing_period_end` date DEFAULT NULL,
  PRIMARY KEY (`utility_id`),
  UNIQUE KEY `utility_id` (`utility_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `property_utility_ibfk_1` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data of property and property utilities together, showing the bills for utilities of properties

```
SELECT
    p.property_id,
    pu.utility_type,
    pu.cost
FROM
    property p
JOIN
    property_utility pu ON p.property_id = pu.property_id;
```

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

property_id	utility_type	cost
1	Electricity	100
2	Water	50
3	Gas	75
4	Internet	40
5	Electricity	120
6	Water	60
7	Gas	80
8	Internet	45
9	Electricity	110
10	Water	55
11	Gas	90
12	Internet	50
13	Electricity	105
14	Water	65
15	Gas	85
16	Internet	48
17	Electricity	95
18	Water	70
19	Gas	88
20	Internet	55

Region Table

Create table

DDL for iu_project_header.region

```
CREATE TABLE `region` (
  `region_id` int NOT NULL,
  `region_name` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`region_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing all properties in Asia

```
SELECT
    p.property_id,
    p.price_per_night,
    p.max_guests,
    a.address_line,
    c.city_name,
    co.country_name,
    r.region_name
FROM
    property p
JOIN
    address a ON p.address_id = a.address_id
JOIN
    city c ON a.city_id = c.city_id
JOIN
    country co ON c.country_id = co.country_id
JOIN
    region r ON co.region_id = r.region_id
WHERE
    r.region_name = 'Asia';
```

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

property_id	price_per_night	max_guests	address_line	city_name	country_name	region_name
11	280	5	108 Poplar St	Beijing	China	Asia
12	600	8	109 Cherry St	Beijing	China	Asia
13	450	7	110 Walnut St	Tokyo	Japan	Asia
14	380	5	111 Hickory St	Tokyo	Japan	Asia

Query showing data

```
SELECT * FROM region;
```

Result Grid | Filter Rows:

region_id	region_name
1	North America
2	Europe
3	Asia
4	Africa
5	South America
6	Australia
7	Antarctica
8	Middle East
9	Central America
10	Caribbean
11	Northern Europe
12	Western Europe
13	Eastern Europe
14	Southern Europe
15	Southeast Asia
16	East Asia
17	South Asia
18	Central Asia
19	North Africa
20	Sub-Saharan A...
NULL	NULL

Review Table

Create table

DDL for iu_project_header.review

```sql
CREATE TABLE `review` (
  `review_id` int NOT NULL AUTO_INCREMENT,
  `user_id` int unsigned NOT NULL,
  `property_id` int NOT NULL,
  `review_text` text,
  `rating` decimal(2,1) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`review_id`),
  KEY `user_id` (`user_id`),
  KEY `property_id` (`property_id`),
  CONSTRAINT `review_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `review_ibfk_2` FOREIGN KEY (`property_id`) REFERENCES `property` (`property_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from reviews, user accounts and properties with ratings from people that wrote the review including its context

```sql
SELECT
    r.review_id,
    ua.first_name,
    ua.last_name,
    p.property_id,
    r.review_text,
    r.rating,
    r.created_at
FROM
    review r
JOIN
    user_account ua ON r.user_id = ua.user_id
JOIN
    property p ON r.property_id = p.property_id;
```

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

review_id	first_name	last_name	property_id	review_text	rating	created_at
1	Carilyn	Jezard	1	Amazing property with stunning views!	5.0	2023-12-01 10:00:00
2	Rani	Goodhay	2	Very clean and well-maintained.	4.5	2023-12-02 11:00:00
3	Julita	Elt	3	Host was very helpful and responsive.	5.0	2023-12-03 12:00:00
4	Meir	Davids	4	Perfect location for a family vacation.	4.8	2023-12-04 13:00:00
5	Sawyer	Cavill	5	Had some issues with the Wi-Fi, but overall great.	4.0	2023-12-05 14:00:00
6	Dan	Ledgerton	6	Beautiful interiors and comfortable stay.	4.7	2023-12-06 15:00:00
7	D'arcy	Vivien	7	Would definitely recommend to friends.	4.9	2023-12-07 16:00:00
8	Lemuel	Jedraszczyk	8	The property exceeded our expectations.	5.0	2023-12-08 17:00:00
9	Josias	Ohms	9	Great amenities but a bit noisy at night.	4.3	2023-12-09 18:00:00
10	Viva	Houtby	10	Host went above and beyond to make us feel w...	5.0	2023-12-10 19:00:00
11	Cathleen	Laydon	11	Decent stay, but could use some upgrades.	3.8	2023-12-11 20:00:00
12	Sayers	Batchellor	12	Loved the pool and outdoor area.	4.6	2023-12-12 21:00:00
13	Allan	Mackett	13	Kitchen was well-equipped for cooking.	4.7	2023-12-13 22:00:00
14	Naomi	Nussii	14	Convenient location near public transport.	4.5	2023-12-14 23:00:00
15	Vyky	Murrigans	15	Bed was super comfortable!	5.0	2023-12-15 08:00:00
16	Harriette	Kienle	16	Great value for the price.	4.4	2023-12-16 09:00:00
17	Yul	Castle	17	Amazing for a weekend getaway.	4.8	2023-12-17 10:00:00
18	Eleanora	Portingale	18	Check-in process was smooth and hassle-free.	4.9	2023-12-18 11:00:00
19	Sargent	Lucken	19	The area was very quiet and relaxing.	4.7	2023-12-19 12:00:00
20	Filia	Passmore	20	Exceptional experience! Will book again.	5.0	2023-12-20 13:00:00

Roles Table

Create table

DDL for iu_project_header.roles

```
CREATE TABLE `roles` (
  `role_id` int unsigned NOT NULL AUTO_INCREMENT,
  `role_name` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`role_id`),
  UNIQUE KEY `role_id` (`role_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from the table

```
SELECT * FROM roles;
```

Result Grid | Filter Rows:

role_id	role_name
1	Admin
2	Guest
3	Host
NULL	NULL

Query showing roles of users and their names

```
SELECT
    ua.user_id,
    ua.first_name,
    ua.last_name,
    ua.email_address,
    r.role_id,
    r.role_name
FROM
    user_account ua
JOIN
    user_roles ur ON ua.user_id = ur.user_id
JOIN
    roles r ON ur.role_id = r.role_id;
```

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

user_id	first_name	last_name	email_address	role_id	role_name
1	Carilyn	Jezard	cjezard0@comcast.net	1	Admin
4	Meir	Davids	mdavids3@e-recht24.de	1	Admin
9	Josias	Ohms	johms8@symantec.com	1	Admin
10	Viva	Houtby	vhoutby9@apache.org	1	Admin
14	Naomi	Nussii	nnussiid@infoseek.co.jp	1	Admin
17	Yul	Castle	ycastleg@dedecms.com	1	Admin
18	Eleanora	Portingale	eportingaleh@simplemachines.org	1	Admin
1	Carilyn	Jezard	cjezard0@comcast.net	2	Guest
2	Rani	Goodhay	rgoodhay1@state.tx.us	2	Guest
3	Julita	Elt	jelt2@devhub.com	2	Guest
5	Sawyer	Cavill	scavill4@constantcontact.com	2	Guest
6	Dan	Ledgerton	dledgerton5@aboutads.info	2	Guest
7	D'arcy	Vivien	dvivien6@ask.com	2	Guest
11	Cathleen	Laydon	claydona@google.ru	2	Guest
13	Allan	Mackett	amackettc@msn.com	2	Guest
16	Harriette	Kienle	hkienlef@amazon.com	2	Guest
2	Rani	Goodhay	rgoodhay1@state.tx.us	3	Host
8	Lemuel	Jedraszczyk	ljedraszczyk7@themeforest.net	3	Host
12	Sayers	Batchellor	sbatchellorb@sbwire.com	3	Host
15	Vyky	Murrigans	vmurriganse@prlog.org	3	Host
19	Sargent	Lucken	sluckeni@1und1.de	3	Host

Social Media Table

Create table

DDL for iu_project_header.social_media

```sql
CREATE TABLE `social_media` (
  `network_id` int NOT NULL,
  `user_account_id` int unsigned DEFAULT NULL,
  `account_url` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`network_id`),
  KEY `user_account_id` (`user_account_id`),
  CONSTRAINT `social_media_ibfk_1` FOREIGN KEY (`user_account_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing the URLs of social networks and usernames with name

```sql
SELECT
    sm.network_id,
    sm.account_url,
    ua.user_name,
    ua.first_name
FROM
    social_media sm
JOIN
    user_account ua ON sm.user_account_id = ua.user_id;
```

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

network_id	account_url	user_name	first_name
1	https://twitter.com/user101	cjezard0	Carilyn
2	https://facebook.com/user102	rgoodhay1	Rani
3	https://instagram.com/user103	jelt2	Julita
4	https://linkedin.com/in/user104	mdavids3	Meir
5	https://twitter.com/user105	scavill4	Sawyer
6	https://facebook.com/user106	dledgerton5	Dan
7	https://instagram.com/user107	dvivien6	D'arcy
8	https://linkedin.com/in/user108	ljedraszczyk7	Lemuel
9	https://twitter.com/user109	johms8	Josias
10	https://facebook.com/user110	vhoutby9	Viva
11	https://instagram.com/user111	claydona	Cathleen
12	https://linkedin.com/in/user112	sbatchellorb	Sayers
13	https://twitter.com/user113	amackettc	Allan
14	https://facebook.com/user114	nnussiid	Naomi
15	https://instagram.com/user115	vmurriganse	Vyky
16	https://linkedin.com/in/user116	hkienlef	Harriette
17	https://twitter.com/user117	ycastleg	Yul
18	https://facebook.com/user118	eportingaleh	Eleanora
19	https://instagram.com/user119	sluckeni	Sargent
20	https://linkedin.com/in/user120	fpassmorej	Filia

Support Tickets Table

Create table

DDL for iu_project_header.support_tickets

```
CREATE TABLE `support_tickets` (
  `ticket_id` int NOT NULL,
  `user_id` int unsigned DEFAULT NULL,
  `subject` varchar(255) DEFAULT NULL,
  `description` text,
  `status` varchar(50) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `assigned_admin_id` int unsigned DEFAULT NULL,
  PRIMARY KEY (`ticket_id`),
  KEY `user_id` (`user_id`),
  KEY `assigned_admin_id` (`assigned_admin_id`),
  CONSTRAINT `support_tickets_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `support_tickets_ibfk_2` FOREIGN KEY (`assigned_admin_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from support tickets showing the username, names and descriptions with status and subject

```
SELECT
    st.ticket_id,
    st.subject,
    st.description,
    st.status,
    ua.first_name,
    ua.user_name
FROM
    support_tickets st
JOIN
    user_account ua ON st.user_id = ua.user_id;
```

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

ticket_id	subject	description	status	first_name	user_name
1	Login Issue	Unable to log into my account	Open	Carilyn	cjezard0
2	Payment Problem	Payment failed while booking a property	Resolved	Rani	rgoodhay1
3	Account Suspension	My account was suspended without reason	Pending	Julita	jelt2
4	Property Listing	Need help listing my property	In Progress	Meir	mdavids3
5	Refund Request	Requesting refund for a cancelled booking	Open	Sawyer	scavill4
6	Technical Issue	Website not loading on my device	Resolved	Dan	dledgerton5
7	Feature Request	Add a filter for pet-friendly properties	Closed	D'arcy	dvivien6
8	Cancellation Issue	Unable to cancel my booking	Open	Lemuel	ljedraszczyk7
9	Mobile App Bug	App crashes when viewing properties	Pending	Josias	johms8
10	Security Concern	Suspicious activity on my account	Resolved	Viva	vhoutby9
11	Booking Problem	Could not complete booking process	Open	Cathleen	claydona
12	Slow Performance	Website loading very slowly	In Progress	Sayers	sbatchellorb
13	Account Deletion	Want to delete my account permanently	Closed	Allan	amackettc
14	Double Charge	Charged twice for the same booking	Resolved	Naomi	nnussiid
15	Unresponsive Host	Host is not responding to my messages	Open	Vyky	vmurriganse
16	Booking Confirmation	Did not receive confirmation email	Resolved	Harriette	hkienlef
17	Payment Refund	Refund not received for a cancelled booking	Pending	Yul	ycastleg
18	App Update Issue	Unable to update the app on my phone	In Progress	Eleanora	eportingaleh
19	Broken Link	A link on the FAQ page is broken	Open	Sargent	sluckeni
20	Discount Issue	Promo code is not being applied	Resolved	Filia	fpassmorej

User Account Table

Create table

DDL for iu_project_header.user_account

```sql
CREATE TABLE `user_account` (
  `user_id` int unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(100) DEFAULT NULL,
  `last_name` varchar(100) DEFAULT NULL,
  `user_name` varchar(50) DEFAULT NULL,
  `email_address` varchar(255) DEFAULT NULL,
  `hashed_password` varchar(255) DEFAULT NULL,
  `mobile_number` varchar(20) DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `user_id` (`user_id`),
  UNIQUE KEY `email_address` (`email_address`)
) ENGINE=InnoDB AUTO_INCREMENT=1056 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```sql
SELECT * FROM user_account;
```

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

user_id	first_name	last_name	user_name	email_address	hashed_password	mobile_number	created_at	updated_at
1	Carilyn	Jezard	cjezard0	cjezard0@comcast.net	$2a$04$8/FmFT16UCtIFrL91ofqh.SxzzT1apdllL...	267-393-5416	1994-09-10 03:43:02	2018-11-19 17:29:43
2	Rani	Goodhay	rgoodhay1	rgoodhay1@state.tx.us	$2a$04$tQp7k.kotISGsN59.Sg4U.OowpFviAe9...	211-860-3729	1986-08-23 09:30:34	2022-05-16 21:39:20
3	Julita	Elt	jelt2	jelt2@devhub.com	$2a$04$WhnEA6g4qLbMl4Vo0UrEL.enNBbwa//...	207-383-5664	2013-02-24 15:54:05	2015-10-21 22:27:33
4	Meir	Davids	mdavids3	mdavids3@e-recht24.de	$2a$04$2sYV6AhnkauYN/nlLZFdW.K77tXqJKDv...	813-256-3109	2017-02-27 10:23:51	2015-09-24 09:41:13
5	Sawyer	Cavill	scavill4	scavill4@constantcontact.com	$2a$04$gLjet8SAb0bjrKneWQt2aO9iFZSl5.ZLSs...	992-296-0550	1978-09-14 06:05:14	2015-04-17 02:00:54
6	Dan	Ledgerton	dledgerton5	dledgerton5@aboutads.info	$2a$04$XUq1Lxrvk6LV/80iYtC4uO2K4/s.BcljQ0...	844-170-5843	2015-05-14 14:35:40	2017-08-23 01:54:09
7	D'arcy	Vivien	dvivien6	dvivien6@ask.com	$2a$04$NQh6suWXtaJDfF7HjLCE9u3i8R49fl5cE...	159-336-9091	1999-11-01 06:48:26	2016-05-02 12:16:19
8	Lemuel	Jedraszczyk	ljedraszczyk7	ljedraszczyk7@themeforest.net	$2a$04$h7X33S44CULqJwKNAEOe2u3VD5177k...	648-151-9429	2023-04-10 05:52:06	2021-04-07 20:49:13
9	Josias	Ohms	johms8	johms8@symantec.com	$2a$04$gkL8srek565qRyheimuBTuJAyTPFKFpM...	746-824-5262	1988-10-24 09:41:52	2013-09-21 18:57:11
10	Viva	Houtby	vhoutby9	vhoutby9@apache.org	$2a$04$VeZtA3zERGDf..BzIzv2Z.FNYnjTzSQrD...	554-837-9848	1999-11-14 15:43:34	2017-12-02 14:46:46
11	Cathleen	Laydon	claydona	claydona@google.ru	$2a$04$KPOARjBOH5D5j/QmHLGLEelVteYD0mn...	525-625-3256	2006-11-08 10:28:37	2009-06-10 11:40:56
12	Sayers	Batchellor	sbatchellorb	sbatchellorb@sbwire.com	$2a$04$yeKNdJa3fjHKNmYRs26ndOBw3ob7/KF...	129-452-5043	1998-09-24 09:50:14	2019-02-16 07:20:02
13	Allan	Mackett	amackettc	amackettc@msn.com	$2a$04$IXdEy91aiynjkhv2lFbJaeL4isrURbhdWe...	922-481-7859	1971-01-28 11:12:04	2023-09-24 22:53:36
14	Naomi	Nussii	nnussiid	nnussiid@infoseek.co.jp	$2a$04$g91dG9fL3z3E/AOZ6y8B7.rRx7gdBHqp...	112-829-4282	1989-07-06 04:43:20	2017-03-25 22:09:57
15	Vyky	Murrigans	vmurriganse	vmurriganse@prlog.org	$2a$04$aSVfOYhIvaDWptkaBpbJ1eP8R97K1LG...	455-197-5281	2010-05-07 05:01:43	2014-08-14 08:11:17
16	Harriette	Kienle	hkienlef	hkienlef@amazon.com	$2a$04$6YkJTHouPo5mEYx8nokKBustuh4bZCR...	568-429-2865	2002-09-06 22:22:17	2015-03-03 16:54:23
17	Yul	Castle	ycastleg	ycastleg@dedecms.com	$2a$04$.0A9dccPhCfHAyWoGZI3NO.q0PXbY3Z...	776-153-2326	2013-08-04 17:36:49	2018-03-05 11:13:50
18	Eleanora	Portingale	eportingaleh	eportingaleh@simplemachines....	$2a$04$3lvtwmP6ET.F5K5aDZJjm.t1hUKigZhgt...	227-275-1312	2021-12-13 06:59:02	2022-07-14 21:25:48
19	Sargent	Lucken	sluckeni	sluckeni@1und1.de	$2a$04$PLyJkyfLtjLBzU7WcKGXCuOcYCxBSnxu...	604-537-3589	2017-03-17 00:09:40	2016-03-20 21:03:36
20	Filia	Passmore	fpassmorej	fpassmorej@dmoz.org	$2a$04$5yV.nFTtutRZaiU1Y.VVLOZioPXRrOWIz...	100-328-0168	2012-04-22 00:36:34	2014-12-27 06:36:40
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Other presentation slides have queries combining data from user account table

User Activity Log Table

Create table

DDL for iu_project_header.user_activity_log

```
CREATE TABLE `user_activity_log` (
  `log_id` int NOT NULL,
  `user_id` int unsigned DEFAULT NULL,
  `activity_type` varchar(50) DEFAULT NULL,
  `activity_timestamp` timestamp NULL DEFAULT NULL,
  `activity_description` text,
  PRIMARY KEY (`log_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `user_activity_log_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from activity log and user account, showing names, usernames and activity types with timestamps

```
SELECT
    ua.first_name,
    ua.user_name,
    ual.activity_type,
    ual.activity_description,
    ual.activity_timestamp
FROM
    user_account ua
JOIN
    user activity log ual ON ua.user id = ual.user id;
```

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

first_name	user_name	activity_type	activity_description	activity_timestamp
Carilyn	cjezard0	Login	User logged into the system	2023-12-01 08:00:00
Rani	rgoodhay1	Profile Update	User updated profile information	2023-12-01 09:00:00
Julita	jelt2	Password Change	User changed account password	2023-12-02 10:00:00
Meir	mdavids3	Payment	User made a payment	2023-12-02 11:00:00
Sawyer	scavill4	Logout	User logged out of the system	2023-12-03 12:00:00
Dan	dledgerton5	Message Sent	User sent a message	2023-12-03 13:00:00
D'arcy	dvivien6	Login	User logged into the system	2023-12-04 14:00:00
Lemuel	ljedraszczyk7	Booking	User made a booking	2023-12-04 15:00:00
Josias	johms8	Profile Update	User updated profile information	2023-12-05 16:00:00
Viva	vhoutby9	Payment	User made a payment	2023-12-05 17:00:00
Cathleen	claydona	Logout	User logged out of the system	2023-12-06 18:00:00
Sayers	sbatchellorb	Message Received	User received a message	2023-12-06 19:00:00
Allan	amackettc	Login	User logged into the system	2023-12-07 20:00:00
Naomi	nnussiid	Profile Update	User updated profile information	2023-12-07 21:00:00
Vyky	vmurriganse	Password Change	User changed account password	2023-12-08 22:00:00
Harriette	hkienlef	Payment	User made a payment	2023-12-08 23:00:00
Yul	ycastleg	Logout	User logged out of the system	2023-12-09 08:00:00
Eleanora	eportingaleh	Message Sent	User sent a message	2023-12-09 09:00:00
Sargent	sluckeni	Login	User logged into the system	2023-12-10 10:00:00
Filia	fpassmorej	Booking	User made a booking	2023-12-10 11:00:00

User Language Table

Create table

DDL for iu_project_header.user_language

```
CREATE TABLE `user_language` (
  `user_id` int unsigned NOT NULL,
  `language_id` int unsigned NOT NULL,
  PRIMARY KEY (`user_id`,`language_id`),
  KEY `language_id` (`language_id`),
  CONSTRAINT `user_language_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `user_language_ibfk_2` FOREIGN KEY (`language_id`) REFERENCES `language` (`language_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```
SELECT * FROM user_language;
```

Result Grid Filter Rows:

user_id	language_id
1	1
1	2
2	3
2	4
3	5
4	6
5	7
6	8
7	9
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17
16	18
17	19
18	20
NULL	NULL

Specific query example is in language slide

User Message Table

Create table

DDL for iu_project_header.user_message

```sql
CREATE TABLE `user_message` (
  `id` int NOT NULL,
  `sender_id` int unsigned DEFAULT NULL,
  `receiver_id` int unsigned DEFAULT NULL,
  `message_content` text,
  `status` varchar(50) DEFAULT NULL,
  `sent_at_timestamp` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `sender_id` (`sender_id`),
  KEY `receiver_id` (`receiver_id`),
  CONSTRAINT `user_message_ibfk_1` FOREIGN KEY (`sender_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `user_message_ibfk_2` FOREIGN KEY (`receiver_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data with the messages between the users and the content of the messages, as well as their status.

```sql
SELECT
    um.id AS message_id,
    sender.first_name AS sender_first_name,
    receiver.first_name AS receiver_first_name,
    um.message_content,
    um.status,
    um.sent_at_timestamp
FROM
    user_message um
JOIN
    user_account sender ON um.sender_id = sender.user_id
JOIN
    user_account receiver ON um.receiver_id = receiver.user_id;
```

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

message_id	sender_first_name	receiver_first_name	message_content	status	sent_at_timestamp
1	Carilyn	Rani	Hello, how are you?	Sent	2023-12-01 10:00:00
2	Rani	Carilyn	I am good, thanks!	Read	2023-12-01 10:05:00
3	Julita	Meir	Are you available for a call?	Sent	2023-12-02 11:00:00
4	Meir	Julita	Yes, let me know when.	Read	2023-12-02 11:10:00
5	Sawyer	Dan	Can we reschedule our meeting?	Sent	2023-12-03 12:00:00
6	Dan	Sawyer	Sure, what time works for you?	Read	2023-12-03 12:15:00
7	D'arcy	Lemuel	I loved your recent post!	Sent	2023-12-04 13:00:00
8	Lemuel	D'arcy	Thank you so much!	Read	2023-12-04 13:20:00
9	Josias	Viva	Please check the document I sent.	Sent	2023-12-05 14:00:00
10	Viva	Josias	Got it, will review and revert.	Read	2023-12-05 14:30:00
11	Cathleen	Sayers	Let's catch up soon.	Sent	2023-12-06 15:00:00
12	Sayers	Cathleen	Sure, let me know when.	Read	2023-12-06 15:20:00
13	Allan	Naomi	I have sent you the proposal.	Sent	2023-12-07 16:00:00
14	Naomi	Allan	Thanks, I will go through it.	Read	2023-12-07 16:30:00
15	Vyky	Harriette	Are you free this weekend?	Sent	2023-12-08 17:00:00
16	Harriette	Vyky	Yes, let's plan something.	Read	2023-12-08 17:15:00
17	Yul	Eleanora	Please share the presentation.	Sent	2023-12-09 18:00:00
18	Eleanora	Yul	I will send it by EOD.	Read	2023-12-09 18:30:00
19	Sargent	Filia	Can you assist with the report?	Sent	2023-12-10 19:00:00
20	Filia	Sargent	Sure, send me the details.	Read	2023-12-10 19:20:00

User Notification Table

Create table

DDL for iu_project_header.user_notification

```
CREATE TABLE `user_notification` (
  `notification_id` int NOT NULL,
  `user_id` int unsigned DEFAULT NULL,
  `notification_type` varchar(50) DEFAULT NULL,
  `notification_content` text,
  `sent_at` timestamp NULL DEFAULT NULL,
  `is_read` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`notification_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `user_notification_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data from user notification and user_data, containing information from both

```
SELECT
    ua.user_id,
    ua.first_name,
    ua.last_name,
    ua.email_address,
    r.role_id,
    r.role_name
FROM
    user_account ua
JOIN
    user_roles ur ON ua.user_id = ur.user_id
JOIN
    roles r ON ur.role_id = r.role_id;
```

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

user_id	first_name	last_name	email_address	role_id	role_name
1	Carilyn	Jezard	cjezard0@comcast.net	1	Admin
4	Meir	Davids	mdavids3@e-recht24.de	1	Admin
9	Josias	Ohms	johms8@symantec.com	1	Admin
10	Viva	Houtby	vhoutby9@apache.org	1	Admin
14	Naomi	Nussii	nnussiid@infoseek.co.jp	1	Admin
17	Yul	Castle	ycastleg@dedecms.com	1	Admin
18	Eleanora	Portingale	eportingaleh@simplemachines.org	1	Admin
1	Carilyn	Jezard	cjezard0@comcast.net	2	Guest
2	Rani	Goodhay	rgoodhay1@state.tx.us	2	Guest
3	Julita	Elt	jelt2@devhub.com	2	Guest
5	Sawyer	Cavill	scavill4@constantcontact.com	2	Guest
6	Dan	Ledgerton	dledgerton5@aboutads.info	2	Guest
7	D'arcy	Vivien	dvivien6@ask.com	2	Guest
11	Cathleen	Laydon	claydona@google.ru	2	Guest
13	Allan	Mackett	amackettc@msn.com	2	Guest
16	Harriette	Kienle	hkienlef@amazon.com	2	Guest
2	Rani	Goodhay	rgoodhay1@state.tx.us	3	Host
8	Lemuel	Jedraszczyk	ljedraszczyk7@themeforest.net	3	Host
12	Sayers	Batchellor	sbatchellorb@sbwire.com	3	Host
15	Vyky	Murrigans	vmurriganse@prlog.org	3	Host
19	Sargent	Lucken	sluckeni@1und1.de	3	Host

User Roles Table

Create table

DDL for iu_project_header.user_roles

```
CREATE TABLE `user_roles` (
  `user_id` int unsigned NOT NULL,
  `role_id` int unsigned NOT NULL,
  PRIMARY KEY (`user_id`,`role_id`),
  KEY `role_id` (`role_id`),
  CONSTRAINT `user_roles_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user_account` (`user_id`),
  CONSTRAINT `user_roles_ibfk_2` FOREIGN KEY (`role_id`) REFERENCES `roles` (`role_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Query showing data

```
SELECT * FROM user_roles;
```

Result Grid Filter Rows:

user_id	role_id
1	1
4	1
9	1
10	1
14	1
17	1
18	1
1	2
2	2
3	2
5	2
6	2
7	2
11	2
13	2
16	2
2	3
8	3
12	3
15	3
19	3
NULL	NULL

Example of a query can be found in roles table

SQL scripts: Insert statements
(in the order needed to be inserted)

Dummy data for the region table

```sql
-- GEOGRAPHICAL DATA
-- Insert dummy data into the region table
INSERT IGNORE INTO region (region_id, region_name) VALUES
(1, 'North America'),
(2, 'Europe'),
(3, 'Asia'),
(4, 'Africa'),
(5, 'South America'),
(6, 'Australia'),
(7, 'Antarctica'),
(8, 'Middle East'),
(9, 'Central America'),
(10, 'Caribbean'),
(11, 'Northern Europe'),
(12, 'Western Europe'),
(13, 'Eastern Europe'),
(14, 'Southern Europe'),
(15, 'Southeast Asia'),
(16, 'East Asia'),
(17, 'South Asia'),
(18, 'Central Asia'),
(19, 'North Africa'),
(20, 'Sub-Saharan Africa');
```

Dummy data for the country table

```sql
-- Insert dummy data into the country table
INSERT IGNORE INTO country (country_id, country_name, country_code, region_id) VALUES
(1, 'United States', 'US', 1),
(2, 'Canada', 'CA', 1),
(3, 'Germany', 'DE', 2),
(4, 'France', 'FR', 12),
(5, 'China', 'CN', 16),
(6, 'Japan', 'JP', 16),
(7, 'India', 'IN', 17),
(8, 'Nigeria', 'NG', 20),
(9, 'Brazil', 'BR', 5),
(10, 'Argentina', 'AR', 5),
(11, 'Australia', 'AU', 6),
(12, 'Russia', 'RU', 13),
(13, 'United Kingdom', 'UK', 11),
(14, 'Italy', 'IT', 14),
(15, 'Spain', 'ES', 14),
(16, 'Mexico', 'MX', 9),
(17, 'Saudi Arabia', 'SA', 8),
(18, 'South Africa', 'ZA', 20),
(19, 'Egypt', 'EG', 19),
(20, 'Thailand', 'TH', 15);
```

Dummy data for the city table

```sql
-- Insert dummy data into the cities table
INSERT IGNORE INTO city (city_id, city_name, country_id) VALUES
(1, 'Tokyo', 6),
(2, 'Beijing', 5),
(3, 'Munich', 3),
(4, 'Delhi', 7),
(5, 'São Paulo', 9),
(6, 'Sydney', 11),
(7, 'New York', 1),
(8, 'Abuja', 8),
(9, 'Paris', 4),
(10, 'Berlin', 3),
(11, 'Vancouver', 2),
(12, 'Mumbai', 7),
(13, 'Osaka', 6),
(14, 'Shanghai', 5),
(15, 'Melbourne', 11),
(16, 'Los Angeles', 1),
(17, 'Rio de Janeiro', 9),
(18, 'Toronto', 2),
(19, 'Lagos', 8),
(20, 'Lyon', 4);
```

Dummy data for the address table

```sql
-- Insert dummy data into the addresses table
INSERT IGNORE INTO address (address_id, address_line, city_id) VALUES
(1, '63 Fulton Way', 7),
(2, '067 School Way', 15),
(3, '7702 Dawn Center', 8),
(4, '410 Amoth Alley', 10),
(5, '7384 Talisman Parkway', 14),
(6, '23048 Union Parkway', 14),
(7, '8 Loftsgordon Circle', 1),
(8, '63800 Prairieview Alley', 11),
(9, '364 Atwood Hill', 1),
(10, '17 Longview Pass', 4),
(11, '01022 Dayton Park', 7),
(12, '62477 Spohn Park', 2),
(13, '86 Steensland Junction', 5),
(14, '4016 Heffernan Pass', 7),
(15, '0 Hallows Avenue', 1),
(16, '386 Lighthouse Bay Plaza', 11),
(17, '50201 6th Hill', 9),
(18, '64703 Washington Lane', 14),
(19, '6 Warrior Trail', 14),
(20, '70755 Monica Avenue', 18);
```

Dummy data for the roles table

```sql
-- ROLES AND LANGUAGES
-- Insert dummy data into the roles table
INSERT IGNORE INTO roles (role_id, role_name) VALUES
(1, 'Admin'),
(2, 'Guest'),
(3, 'Host');
```

Dummy data for the language table

```sql
-- Insert dummy data into the language table
INSERT IGNORE INTO language (language_id, language_name) VALUES
(1, 'English'),
(2, 'Spanish'),
(3, 'French'),
(4, 'German'),
(5, 'Chinese'),
(6, 'Japanese'),
(7, 'Russian'),
(8, 'Italian'),
(9, 'Portuguese'),
(10, 'Korean'),
(11, 'Hindi'),
(12, 'Arabic'),
(13, 'Dutch'),
(14, 'Swedish'),
(15, 'Norwegian'),
(16, 'Turkish'),
(17, 'Polish'),
(18, 'Danish'),
(19, 'Finnish'),
(20, 'Greek');
```

Dummy data for the user_account table

```sql
-- USER ACCOUNT
-- Insert dummy data into the user_account table
INSERT IGNORE INTO user_account (user_id, first_name, last_name, user_name, email_address, hashed_password, mobile_number, created_at, updated_at) VALUES
(1, 'Carilyn', 'Jezard', 'cjezard0', 'cjezard0@comcast.net', '$2a$04$8/FmFT16UCtIFrL91ofqh.SxzzT1apdllLvepkHtzmvOmSOkYCeNW', '267-393-5416', '1994-09-10 03:43:02', '2018-11-19 17:29:43'),
(2, 'Rani', 'Goodhay', 'rgoodhay1', 'rgoodhay1@state.tx.us', '$2a$04$tQp7k.kotISGsN59.Sg4U.OowpFviAe9w1qNBwp.4KfzQQuGMuvJi', '211-860-3729', '1986-08-23 09:30:34', '2022-05-16 21:39:20'),
(3,'Julita', 'Elt', 'jelt2', 'jelt2@devhub.com', '$2a$04$WhnEA6g4qLbMl4Vo0UrEL.enNBbwa//yMIHqPBTp.DYEe0yRUM2AW', '207-383-5664', '2013-02-24 15:54:05', '2015-10-21 22:27:33'),
(4,'Meir', 'Davids', 'mdavids3', 'mdavids3@e-recht24.de', '$2a$04$2sYV6AhnkauYN/nlLZFdW.K77tXqJKDvexwn289f7NPa2pU1tLtM6', '813-256-3109', '2017-02-27 10:23:51', '2015-09-24 09:41:13'),
(5,'Sawyer', 'Cavill', 'scavill4', 'scavill4@constantcontact.com', '$2a$04$gLjet8SAb0bjrKneWQt2aO9iFZSl5.ZLSsF/ZFPzQWLLTtBquKjMK', '992-296-0550', '1978-09-14 06:05:14', '2015-04-17 02:00:54'),
(6,'Dan', 'Ledgerton', 'dledgerton5', 'dledgerton5@aboutads.info', '$2a$04$XUq1Lxrvk6LV/80iYtC4uO2K4/s.BcljQ0yBiXxoei.MQNuprGuBy', '844-170-5843', '2015-05-14 14:35:40', '2017-08-23 01:54:09'),
(7,'Darcy', 'Vivien', 'dvivien6', 'dvivien6@ask.com', '$2a$04$NQh6suWXtaJDfF7HjLCE9u3i8R49fl5cEnAqyhT/t7VpAZZeazQLi', '159-336-9091', '1999-11-01 06:48:26', '2016-05-02 12:16:19'),
(8,'Lemuel', 'Jedraszczyk', 'ljedraszczyk7', 'ljedraszczyk7@themeforest.net', '$2a$04$h7X33S44CULqJwKNAEOe2u3VD5177kjplvfToKWGD503WKMoHiJL.', '648-151-9429', '2023-04-10 05:52:06', '2021-04-07 20:49:13'),
(9,'Josias', 'Ohms', 'johms8', 'johms8@symantec.com', '$2a$04$gkL8srek565qRyheimuBTuJAyTPFKFpMDOVCpFab5opvJCQRLHOZa', '746-824-5262', '1988-10-24 09:41:52', '2013-09-21 18:57:11'),
(10,'Viva', 'Houtby', 'vhoutby9', 'vhoutby9@apache.org', '$2a$04$VeZtA3zERGDf..BzIzv2Z.FNYnjTzSQrDNaWMcNy4ijfoKZ7FV.2K', '554-837-9848', '1999-11-14 15:43:34', '2017-12-02 14:46:46'),
(11,'Cathleen', 'Laydon', 'claydona', 'claydona@google.ru', '$2a$04$KPOARjBOH5D5j/QmHLGLEelVteYD0mnVrgwqbxUCk2UDdP0pR8DI6', '525-625-3256', '2006-11-08 10:28:37', '2009-06-10 11:40:56'),
(12,'Sayers', 'Batchellor', 'sbatchellorb', 'sbatchellorb@sbwire.com', '$2a$04$yeKNdJa3fjHKNmYRs26ndOBw3ob7/KFohq5BiRcA/FJ0ycbzAu1Oe', '129-452-5043', '1998-09-24 09:50:14', '2019-02-16 07:20:02'),
(13,'Allan', 'Mackett', 'amackettc', 'amackettc@msn.com', '$2a$04$IXdEy91aiynjkhv2lFbJaeL4isrURbhdWeHDt8RcK8YjBiuufjD7K', '922-481-7859', '1971-01-28 11:12:04', '2023-09-24 22:53:36'),
(14,'Naomi', 'Nussii', 'nnussiid', 'nnussiid@infoseek.co.jp', '$2a$04$g91dG9fL3z3E/AOZ6y8B7.rRx7gdBHqppAiggu0Ra26DWyY1UFtEe', '112-829-4282', '1989-07-06 04:43:20', '2017-03-25 22:09:57'),
(15,'Vyky', 'Murrigans', 'vmurriganse', 'vmurriganse@prlog.org', '$2a$04$aSVfOYhIvaDWptkaBpbJ1eP8R97K1LGbMGeon5pFM86ExtKz5MG6i', '455-197-5281', '2010-05-07 05:01:43', '2014-08-14 08:11:17'),
(16,'Harriette', 'Kienle', 'hkienlef', 'hkienlef@amazon.com', '$2a$04$6YkUTHouPo5mEYx8nokKBustuh4bZCRRYGD7TKl.X3OI6yNchlCzy', '568-429-2865', '2002-09-06 22:22:17', '2015-03-03 16:54:23'),
(17,'Yul', 'Castle', 'ycastleg', 'ycastleg@dedecms.com', '$2a$04$.0A9dccPhCfHAyWoGZI3NO.q0PXbY3ZUttJhAlUpFqhd2yeUP3Sl2', '776-153-2326', '2013-08-04 17:36:49', '2018-03-05 11:13:50'),
(18,'Eleanora', 'Portingale', 'eportingaleh', 'eportingaleh@simplemachines.org', '$2a$04$3lvtwmP6ET.F5K5aDZJjm.t1hUKigZhgtc0kUL/8tiKKR5GBuIlEW', '227-275-1312', '2021-12-13 06:59:02', '2022-07-14 21:25:48'),
(19,'Sargent', 'Lucken', 'sluckeni', 'sluckeni@1und1.de', '$2a$04$PLyJkyfLtjLBzU7WcKGXCuOcYCxBSnxu6o.hE3yyOLqARtyVYpaYu', '604-537-3589', '2017-03-17 00:09:40', '2016-03-20 21:03:36'),
(20,'Filia', 'Passmore', 'fpassmorej', 'fpassmorej@dmoz.org', '$2a$04$5yV.nFTtutRZaiU1Y.VVLOZioPXRrOWIzDe8VIOsQZpF4OYZ0mk7y', '100-328-0168', '2012-04-22 00:36:34', '2014-12-27 06:36:40');
```

Dummy data for the user_roles and user_language tables

```sql
-- Insert dummy data into the user_roles table
INSERT IGNORE INTO user_roles (user_id, role_id) VALUES
(1, 1), (1, 2), (2, 2), (2, 3), (3, 2),
(4, 1), (5, 2), (6, 2), (7, 2), (8, 3),
(9, 1), (10, 1), (11, 2), (12, 3), (13, 2),
(14, 1), (15, 3), (16, 2), (17, 1), (18, 1),
(19, 3);

-- Insert dummy data into the user_language table
INSERT IGNORE INTO user_language (user_id, language_id) VALUES
(1, 1), (1, 2), (2, 3), (2, 4), (3, 5),
(4, 6), (5, 7), (6, 8), (7, 9), (8, 10),
(9, 11), (10, 12), (11, 13), (12, 14), (13, 15),
(14, 16), (15, 17), (16, 18), (17, 19), (18, 20);
```

Dummy data for the host table

```sql
-- Insert dummy data into the host table
INSERT IGNORE INTO host (host_id, user_account_id, host_info) VALUES
(1, 1, 'Experienced host with 5 years in the hospitality industry'),
(2, 2, 'New host specializing in unique vacation homes'),
(3, 3, 'Local expert offering city tours along with accommodations'),
(4, 4, 'Friendly host with multiple beachfront properties'),
(5, 5, 'Luxury host catering to high-end clients'),
(6, 6, 'Eco-friendly host with sustainable properties'),
(7, 7, 'Pet-friendly host with dedicated pet amenities'),
(8, 8, 'Host offering family-friendly accommodations'),
(9, 9, 'Experienced host managing properties in urban areas'),
(10, 10, 'Host specializing in long-term rental stays'),
(11, 11, 'Adventure host offering properties in remote areas'),
(12, 12, 'Artistic host with uniquely decorated homes'),
(13, 13, 'Host focused on wellness retreats and spa services'),
(14, 14, 'Corporate host offering business-ready accommodations'),
(15, 15, 'Student-focused host near educational institutions'),
(16, 16, 'Couples-focused host with romantic getaway options'),
(17, 17, 'Tech-savvy host with smart home properties'),
(18, 18, 'Senior-friendly host offering accessible accommodations'),
(19, 19, 'Gourmet host with properties near culinary hotspots'),
(20, 20, 'Seasoned host with expertise in cultural tourism');
```

Dummy data for the property table

```sql
-- Insert dummy data into the property table
INSERT IGNORE INTO property (property_id, host_id, price_per_night, max_guests, created_at, updated_at, address_id) VALUES
(1, 4, '400', 6, '2023-11-06 15:13:27', '2023-11-06 15:14:51', 6),
(2, 12, '600', 8, '2023-11-12 21:05:39', '2023-11-12 21:06:42', 12),
(3, 7, '180', 4, '2023-11-07 16:11:23', '2023-11-07 16:12:56', 7),
(4, 19, '500', 9, '2023-11-19 12:08:45', '2023-11-19 12:09:19', 19),
(5, 5, '250', 5, '2023-11-05 14:04:19', '2023-11-05 14:05:53', 5),
(6, 14, '380', 5, '2023-11-14 23:07:32', '2023-11-14 23:08:48', 14),
(7, 3, '500', 8, '2023-11-03 12:02:14', '2023-11-03 12:03:59', 3),
(8, 1, '200', 4, '2023-11-01 10:15:42', '2023-11-01 10:16:36', 1),
(9, 8, '350', 5, '2023-11-08 17:09:53', '2023-11-08 17:10:47', 8),
(10, 18,  '350', 7, '2023-11-18 11:06:34', '2023-11-18 11:07:28', 18),
(11, 2, '350', 6, '2023-11-02 11:14:27', '2023-11-02 11:15:43', 2),
(12, 20,  '450', 8, '2023-11-20 13:10:45', '2023-11-20 13:12:12', 20),
(13, 13,  '450', 7, '2023-11-13 22:09:31', '2023-11-13 22:11:04', 13),
(14, 6,  '600', 6, '2023-11-06 15:04:18', '2023-11-06 15:05:21', 6),
(15, 10,  '220', 6, '2023-11-10 19:07:45', '2023-11-10 19:09:15', 10),
(16, 9, '300', 4, '2023-11-09 18:03:12', '2023-11-09 18:04:58', 9),
(17, 15,  '200', 6, '2023-11-15 08:13:27', '2023-11-15 08:14:41', 15),
(18, 11,  '280', 5, '2023-11-11 20:05:39', '2023-11-11 20:06:52', 11),
(19, 17,  '300', 6, '2023-11-17 10:02:54', '2023-11-17 10:04:23', 17),
(20, 16, '250', 8, '2023-11-16 09:08:12', '2023-11-16 09:09:48', 16);
```

Dummy data for the property_policy and property_policy_map table

```sql
-- Insert dummy data into the property_policy table
INSERT IGNORE INTO property_policy (policy_id, policy_type, policy_description) VALUES
(1, 'Cancellation Policy', 'Full refund if canceled within 24 hours of booking'),
(2, 'Damage Policy', 'Damage to property incurs a $500 fine'),
(3, 'Check-in Policy', 'Check-in from 3 PM to 10 PM'),
(4, 'Check-out Policy', 'Check-out before 11 AM'),
(5, 'Pet Policy', 'Pets allowed with a $50 cleaning fee'),
(6, 'Smoking Policy', 'No smoking allowed in the property'),
(7, 'Guest Policy', 'Maximum of 4 guests allowed'),
(8, 'Noise Policy', 'Quiet hours from 10 PM to 7 AM'),
(9, 'Key Policy', 'Lost keys incur a $100 replacement fee'),
(10, 'Parking Policy', 'Free parking available on premises'),
(11, 'Cleaning Policy', 'Cleaning fee of $100 applies'),
(12, 'Cancellation Policy', '50% refund if canceled 7 days before arrival'),
(13, 'Damage Policy', 'No refund for intentional damage'),
(14, 'Check-in Policy', 'Check-in from 12 PM to 8 PM'),
(15, 'Check-out Policy', 'Check-out before 10 AM'),
(16, 'Pet Policy', 'No pets allowed'),
(17, 'Smoking Policy', 'Smoking allowed in designated areas only'),
(18, 'Guest Policy', 'Maximum of 6 guests allowed'),
(19, 'Noise Policy', 'No loud music after 9 PM'),
(20, 'Parking Policy', 'No parking available on premises');

-- Insert dummy data into the property_policy_map table
INSERT IGNORE INTO property_policy_map (property_id, policy_id) VALUES
(1, 1), (1, 2), (1, 3), (2, 4), (2, 5),
(3, 6), (3, 7), (3, 8), (4, 9), (4, 10),
(5, 11), (5, 12), (6, 13), (6, 14), (7, 15),
(8, 16), (8, 17), (9, 18), (10, 19), (10, 20);
```

Dummy data for the property_service and property_service_map tables

```sql
-- Insert dummy data into the property_service table
INSERT IGNORE INTO property_service (service_id, service_name, service_description) VALUES
(1, 'Daily Cleaning', 'Daily cleaning services provided'),
(2, 'Concierge', '24/7 concierge service'),
(3, 'Room Service', 'In-room dining available'),
(4, 'Laundry', 'Laundry services provided'),
(5, 'Luggage Storage', 'Secure luggage storage facility'),
(6, 'Shuttle Service', 'Airport and local shuttle service'),
(7, 'Spa', 'On-site spa services'),
(8, 'Grocery Delivery', 'Grocery delivery service available'),
(9, 'Pet Sitting', 'Pet sitting services available'),
(10, 'Babysitting', 'Professional babysitting services'),
(11, 'Tour Assistance', 'Guided tours and assistance available'),
(12, 'Breakfast Included', 'Complimentary breakfast for guests'),
(13, 'Catering', 'Event catering services available'),
(14, 'Valet Parking', 'Valet parking services'),
(15, 'Fitness Classes', 'On-site fitness classes'),
(16, 'Yoga Sessions', 'Private yoga sessions available'),
(17, 'Car Rental', 'On-site car rental services'),
(18, 'Bike Rental', 'Bicycles available for rent'),
(19, 'Massage', 'In-room massage services available'),
(20, 'Event Planning', 'Event planning and coordination services');

-- Insert dummy data into the property_service_map table
INSERT IGNORE INTO property_service_map (property_id, service_id) VALUES
(1, 1), (1, 2), (1, 3), (2, 4), (2, 5),
(3, 6), (3, 7), (4, 8), (4, 9), (5, 10),
(5, 11), (6, 12), (6, 13), (7, 14), (7, 15),
(8, 16), (8, 17), (9, 18), (9, 19), (10, 20);
```

Dummy data for the amenity and property_amenities_map table

```sql
-- Insert dummy data into the amenities table
INSERT IGNORE INTO amenities (amenity_id, amenity_name, amenity_description) VALUES
(1, 'Wi-Fi', 'High-speed wireless internet'),
(2, 'Air Conditioning', 'Central air conditioning'),
(3, 'Heating', 'Central heating system'),
(4, 'TV', 'Flat-screen TV with cable channels'),
(5, 'Kitchen', 'Fully equipped modern kitchen'),
(6, 'Pool', 'Outdoor swimming pool'),
(7, 'Gym', 'In-house fitness center'),
(8, 'Parking', 'Private parking available'),
(9, 'Balcony', 'Spacious private balcony'),
(10, 'Garden', 'Well-maintained garden area'),
(11, 'Washer', 'In-unit washer and dryer'),
(12, 'Dryer', 'In-unit clothes dryer'),
(13, 'Dishwasher', 'Dishwasher in kitchen'),
(14, 'BBQ Grill', 'Outdoor BBQ grill'),
(15, 'Fireplace', 'Indoor fireplace'),
(16, 'Hot Tub', 'Outdoor hot tub'),
(17, 'Game Room', 'Game room with entertainment options'),
(18, 'Playground', 'Outdoor playground for kids'),
(19, 'Sauna', 'Private sauna'),
(20, 'Library', 'Private library with books and magazines');

-- Insert dummy data into the property_amenities_map table
INSERT IGNORE INTO property_amenities_map (property_id, amenity_id) VALUES
(1, 1), (1, 2), (1, 3), (2, 4), (2, 5),
(3, 6), (3, 7), (4, 8), (4, 9), (5, 10),
(5, 11), (6, 12), (6, 13), (7, 14), (7, 15),
(8, 16), (8, 17), (9, 18), (9, 19), (10, 20);
```

Dummy data for the property_utility table

```sql
-- Insert dummy data into the property_utility table
INSERT IGNORE INTO property_utility (utility_id, property_id, utility_type, cost, billing_period_start, billing_period_end) VALUES
(1, 3, 'Water', 55, '2023-11-02', '2023-11-29'),
(2, 5, 'Gas', 90, '2023-11-01', '2023-11-30'),
(3, 1, 'Electricity', 110, '2023-11-03', '2023-11-28'),
(4, 10, 'Internet', 48, '2023-11-01', '2023-11-30'),
(5, 8, 'Water', 70, '2023-11-04', '2023-11-29'),
(6, 4, 'Gas', 85, '2023-11-01', '2023-11-30'),
(7, 16, 'Electricity', 105, '2023-11-02', '2023-11-30'),
(8, 9, 'Internet', 50, '2023-11-01', '2023-11-30'),
(9, 11, 'Water', 65, '2023-11-01', '2023-11-29'),
(10, 6, 'Gas', 88, '2023-11-01', '2023-11-30'),
(11, 20, 'Electricity', 95, '2023-11-01', '2023-11-30'),
(12, 13, 'Internet', 55, '2023-11-02', '2023-11-30'),
(13, 12, 'Water', 60, '2023-11-03', '2023-11-30'),
(14, 14, 'Gas', 80, '2023-11-01', '2023-11-29'),
(15, 7, 'Electricity', 100, '2023-11-01', '2023-11-30'),
(16, 18, 'Internet', 45, '2023-11-02', '2023-11-30'),
(17, 2, 'Water', 50, '2023-11-01', '2023-11-28'),
(18, 15, 'Gas', 75, '2023-11-03', '2023-11-30'),
(19, 17, 'Electricity', 120, '2023-11-01', '2023-11-30'),
(20, 19, 'Internet', 40, '2023-11-04', '2023-11-29');
```

Dummy data for the property_images table

```sql
-- Insert dummy data into the property_images table
INSERT IGNORE INTO property_images (image_id, property_id, image_url, description, uploaded_at) VALUES
(1, 3, 'https://example.com/images/property3_bedroom.jpg', 'Master bedroom', '2023-12-02 11:18:43'),
(2, 1, 'https://example.com/images/property1_interior.jpg', 'Living room', '2023-12-01 10:12:25'),
(3, 7, 'https://example.com/images/property7.jpg', 'Terrace', '2023-12-07 16:09:36'),
(4, 10, 'https://example.com/images/property10.jpg', 'Entrance', '2023-12-10 19:05:57'),
(5, 5, 'https://example.com/images/property5_living.jpg', 'Spacious living room', '2023-12-05 14:11:34'),
(6, 2, 'https://example.com/images/property2.jpg', 'Side view of the property', '2023-12-02 11:07:12'),
(7, 4, 'https://example.com/images/property4_balcony.jpg', 'Balcony with a view', '2023-12-04 13:16:45'),
(8, 9, 'https://example.com/images/property9_office.jpg', 'Home office', '2023-12-09 18:09:27'),
(9, 6, 'https://example.com/images/property6_dining.jpg', 'Dining area', '2023-12-06 15:21:49'),
(10, 8, 'https://example.com/images/property8.jpg', 'Patio', '2023-12-08 17:12:38'),
(11, 1, 'https://example.com/images/property1.jpg', 'Front view of the property', '2023-12-01 10:03:17'),
(12, 5, 'https://example.com/images/property5.jpg', 'Garden area', '2023-12-05 14:02:58'),
(13, 8, 'https://example.com/images/property8_bathroom.jpg', 'Luxury bathroom', '2023-12-08 17:17:45'),
(14, 2, 'https://example.com/images/property2_kitchen.jpg', 'Modern kitchen', '2023-12-02 11:29:16'),
(15, 10, 'https://example.com/images/property10_pool.jpg', 'Private pool', '2023-12-10 19:11:02'),
(16, 7, 'https://example.com/images/property7_gym.jpg', 'In-house gym', '2023-12-07 16:24:10'),
(17, 4, 'https://example.com/images/property4.jpg', 'Pool area', '2023-12-04 13:13:41'),
(18, 6, 'https://example.com/images/property6.jpg', 'Night view', '2023-12-06 15:07:52'),
(19, 3, 'https://example.com/images/property3.jpg', 'Backyard', '2023-12-03 12:08:14'),
(20, 9, 'https://example.com/images/property9.jpg', 'Driveway', '2023-12-09 18:03:59');
```

Dummy data for the booking table

```sql
-- Insert dummy data into the booking table
INSERT IGNORE INTO booking (booking_id, user_id, property_id, booking_date, number_of_guests, total_price, status_id) VALUES
(1, 3, 2, '2023-10-12 10:22:15', 3, 1050.00, 1),
(2, 15, 8, '2023-09-21 11:18:27', 3, 900.00, 2),
(3, 7, 5, '2023-08-15 12:14:36', 2, 750.00, 3),
(4, 10, 7, '2023-12-05 13:07:12', 3, 900.00, 1),
(5, 18, 12, '2023-07-18 14:08:19', 4, 1000.00, 1),
(6, 1, 4, '2023-11-03 15:11:54', 5, 1200.00, 2),
(7, 6, 1, '2023-06-24 16:15:41', 4, 800.00, 1),
(8, 13, 13, '2023-11-10 17:12:48', 2, 700.00, 3),
(9, 19, 14, '2023-09-06 18:09:37', 6, 1400.00, 2),
(10, 2, 3, '2023-08-29 19:13:27', 2, 700.00, 3),
(11, 5, 6, '2023-10-07 20:08:15', 6, 1500.00, 2),
(12, 9, 9, '2023-12-22 21:17:04', 4, 1100.00, 1),
(13, 20, 15, '2023-07-27 22:19:47', 3, 850.00, 1),
(14, 8, 16, '2023-06-11 23:09:51', 2, 650.00, 3),
(15, 11, 10, '2023-08-20 08:14:29', 4, 1150.00, 1),
(16, 12, 11, '2023-11-01 09:18:32', 5, 1300.00, 2),
(17, 14, 17, '2023-10-18 10:06:44', 5, 1250.00, 1),
(18, 4, 19, '2023-09-09 11:03:55', 6, 1400.00, 2),
(19, 17, 18, '2023-07-14 12:15:49', 4, 1000.00, 3),
(20, 16, 20, '2023-10-25 13:08:22', 3, 850.00, 1);
```

Dummy data for the booking_guideline table

```sql
-- Insert dummy data into the booking_guidelines table
INSERT IGNORE INTO booking_guideline (booking_guideline_id, booking_id, property_id, guideline_description) VALUES
(15, 7, 3, 'Check-in after 3 PM.'),
(12, 4, 8, 'No smoking in the property.'),
(9, 6, 5, 'No pets allowed.'),
(3, 2, 10, 'Quiet hours from 10 PM to 7 AM.'),
(17, 5, 6, 'Pool use allowed from 9 AM to 9 PM.'),
(8, 1, 7, 'ID required at check-in.'),
(14, 3, 9, 'Maximum of 4 guests per booking.'),
(1, 8, 1, 'No parties or events allowed.'),
(16, 10, 2, 'Parking space available upon request.'),
(6, 9, 4, 'Please dispose of garbage in designated bins.'),
(20, 12, 11, 'Early check-out available upon request.'),
(10, 13, 15, 'Pets allowed with a fee of $50.'),
(4, 11, 13, 'Please report damages immediately.'),
(11, 14, 16, 'Use of the gym is complimentary.'),
(19, 20, 20, 'Breakfast is served from 7 AM to 10 AM.'),
(2, 18, 18, 'Check-out before 11 AM.'),
(18, 19, 19, 'Use of hot tub allowed until midnight.'),
(13, 15, 14, 'Smoking allowed in designated areas only.'),
(7, 17, 17, 'Complimentary toiletries provided.'),
(5, 16, 12, 'Please follow local COVID-19 guidelines.');
```

Dummy data for the cancellation_policies table

```sql
-- Insert dummy data into the cancellation_policies table
INSERT IGNORE INTO cancellation_policies (policy_id, property_id, cancellation_period, penalty_amount, created_at, updated_at) VALUES
(1, 14, '2024-01-07 09:43:29', 75, '2023-12-03 11:15:32', '2023-12-03 11:17:48'),
(2, 3, '2024-01-15 08:22:19', 125, '2023-12-05 10:08:14', '2023-12-05 10:12:45'),
(3, 10, '2024-01-12 10:39:54', 200, '2023-12-10 12:22:17', '2023-12-10 12:27:03'),
(4, 5, '2024-01-01 14:03:12', 50, '2023-12-01 09:15:24', '2023-12-01 09:18:37'),
(5, 18, '2024-01-18 11:23:45', 300, '2023-12-16 08:11:49', '2023-12-16 08:13:58'),
(6, 8, '2024-01-05 16:17:22', 175, '2023-12-04 15:03:41', '2023-12-04 15:06:29'),
(7, 20, '2024-01-09 10:07:56', 275, '2023-12-11 12:09:14', '2023-12-11 12:13:22'),
(8, 7, '2024-01-04 08:19:33', 150, '2023-12-06 07:17:52', '2023-12-06 07:19:42'),
(9, 1, '2024-01-11 13:45:12', 425, '2023-12-09 14:12:31', '2023-12-09 14:14:17'),
(10, 13, '2024-01-20 10:18:47', 500, '2023-12-19 09:07:53', '2023-12-19 09:11:39'),
(11, 2, '2024-01-02 12:30:16', 100, '2023-12-02 10:15:06', '2023-12-02 10:18:45'),
(12, 9, '2024-01-03 10:17:31', 125, '2023-12-07 11:04:29', '2023-12-07 11:07:53'),
(13, 19, '2024-01-16 09:11:28', 375, '2023-12-13 10:22:37', '2023-12-13 10:24:12'),
(14, 12, '2024-01-14 07:29:58', 275, '2023-12-12 08:06:21', '2023-12-12 08:11:04'),
(15, 11, '2024-01-06 08:54:21', 225, '2023-12-08 09:17:11', '2023-12-08 09:20:18'),
(16, 15, '2024-01-13 10:15:12', 325, '2023-12-14 10:04:36', '2023-12-14 10:08:14'),
(17, 4, '2024-01-08 11:32:45', 450, '2023-12-15 12:19:31', '2023-12-15 12:22:47'),
(18, 6, '2024-01-10 15:41:56', 200, '2023-12-17 13:14:22', '2023-12-17 13:16:58'),
(19, 16, '2024-01-17 08:43:29', 375, '2023-12-18 07:06:19', '2023-12-18 07:09:47'),
(20, 17, '2024-01-19 10:12:14', 450, '2023-12-20 11:05:14', '2023-12-20 11:08:31');
```

Dummy data for the booking_status_history table

```sql
-- Insert dummy data into the booking_status_history table
INSERT IGNORE INTO booking_status_history (history_id, booking_id, booking_status, changed_by, property_id) VALUES
(10, 5, 'Pending', 12, 7),
(3, 9, 'Cancelled', 3, 8),
(8, 15, 'Confirmed', 6, 11),
(19, 1, 'Pending', 19, 14),
(1, 14, 'Cancelled', 1, 5),
(17, 18, 'Confirmed', 17, 2),
(11, 6, 'Pending', 11, 10),
(5, 10, 'Cancelled', 5, 15),
(12, 20, 'Confirmed', 8, 4),
(7, 3, 'Pending', 9, 12),
(20, 13, 'Cancelled', 20, 18),
(9, 4, 'Confirmed', 14, 9),
(4, 7, 'Pending', 4, 20),
(2, 2, 'Cancelled', 2, 6),
(15, 12, 'Confirmed', 18, 16),
(18, 16, 'Pending', 10, 3),
(6, 11, 'Cancelled', 16, 1),
(14, 17, 'Confirmed', 13, 19),
(16, 8, 'Pending', 15, 17),
(13, 19, 'Cancelled', 7, 13);
```

Dummy data for the review table

```sql
-- Insert dummy data into the review table
INSERT IGNORE INTO review (review_id, user_id, property_id, review_text, rating, created_at) VALUES
(1, 5, 1, 'Had some issues with the Wi-Fi, but overall great.', 4.0, '2023-12-01 10:12:34'),
(2, 18, 2, 'Check-in process was smooth and hassle-free.', 4.9, '2023-12-02 11:18:47'),
(3, 11, 3, 'Decent stay, but could use some upgrades.', 3.8, '2023-12-03 12:09:15'),
(4, 3, 4, 'Host was very helpful and responsive.', 5.0, '2023-12-04 13:07:22'),
(5, 10, 5, 'Host went above and beyond to make us feel welcome.', 5.0, '2023-12-05 14:11:56'),
(6, 9, 6, 'Great amenities but a bit noisy at night.', 4.3, '2023-12-06 15:06:37'),
(7, 7, 7, 'Would definitely recommend to friends.', 4.9, '2023-12-07 16:03:14'),
(8, 8, 8, 'The property exceeded our expectations.', 5.0, '2023-12-08 17:02:48'),
(9, 19, 9, 'The area was very quiet and relaxing.', 4.7, '2023-12-09 18:16:41'),
(10, 13, 10, 'Kitchen was well-equipped for cooking.', 4.7, '2023-12-10 19:13:25'),
(11, 1, 11, 'Amazing property with stunning views!', 5.0, '2023-12-11 20:14:59'),
(12, 15, 12, 'Bed was super comfortable!', 5.0, '2023-12-12 21:09:36'),
(13, 6, 13, 'Beautiful interiors and comfortable stay.', 4.7, '2023-12-13 22:05:47'),
(14, 4, 14, 'Perfect location for a family vacation.', 4.8, '2023-12-14 23:08:16'),
(15, 2, 15, 'Very clean and well-maintained.', 4.5, '2023-12-15 08:11:23'),
(16, 20, 16, 'Exceptional experience! Will book again.', 5.0, '2023-12-16 09:17:41'),
(17, 14, 17, 'Convenient location near public transport.', 4.5, '2023-12-17 10:13:29'),
(18, 17, 18, 'Amazing for a weekend getaway.', 4.8, '2023-12-18 11:07:52'),
(19, 12, 19, 'Loved the pool and outdoor area.', 4.6, '2023-12-19 12:09:18'),
(20, 16, 20, 'Great value for the price.', 4.4, '2023-12-20 13:08:33');
```

Dummy data for the payment table

```sql
-- Insert dummy data into the payment table
INSERT IGNORE INTO payment (payment_id, booking_id, payment_date, amount, payment_method, payment_status) VALUES
(1, 6, '2023-11-04 11:13:21', '1200', 'Credit Card', 'Completed'),
(2, 3, '2023-08-16 12:18:49', '1050', 'Bank Transfer', 'Pending'),
(3, 9, '2023-09-07 13:05:15', '1400', 'PayPal', 'Pending'),
(4, 5, '2023-07-19 14:14:37', '1000', 'Credit Card', 'Completed'),
(5, 20, '2023-10-26 15:11:58', '850', 'Credit Card', 'Completed'),
(6, 14, '2023-06-12 16:07:32', '650', 'PayPal', 'Completed'),
(7, 18, '2023-09-10 17:19:44', '1400', 'Credit Card', 'Completed'),
(8, 13, '2023-07-28 18:15:23', '850', 'PayPal', 'Completed'),
(9, 19, '2023-07-15 19:17:41', '1000', 'Bank Transfer', 'Failed'),
(10, 15, '2023-08-21 20:06:52', '1150', 'Credit Card', 'Completed'),
(11, 7, '2023-06-25 21:10:11', '800', 'Credit Card', 'Completed'),
(12, 8, '2023-11-11 22:08:25', '750', 'PayPal', 'Completed'),
(13, 1, '2023-10-13 23:14:29', '800', 'Bank Transfer', 'Failed'),
(14, 16, '2023-11-02 08:09:56', '1300', 'Credit Card', 'Pending'),
(15, 2, '2023-09-22 09:03:41', '700', 'PayPal', 'Completed'),
(16, 11, '2023-08-30 10:18:37', '1500', 'Credit Card', 'Completed'),
(17, 12, '2023-12-23 11:05:23', '1100', 'PayPal', 'Pending'),
(18, 17, '2023-10-19 12:19:45', '1250', 'Bank Transfer', 'Pending'),
(19, 4, '2023-06-26 13:07:38', '900', 'PayPal', 'Pending'),
(20, 10, '2023-09-11 14:11:15', '800', 'Credit Card', 'Completed');
```

Dummy data for the payment_info table

```sql
-- Insert dummy data into the payment_info table
INSERT IGNORE INTO payment_info
(payment_id, user_id, card_number, card_expiry, card_type, billing_address, created_at, method_type) VALUES
(1, 12, '5602255230400716', '2027-11-17', 'bankcard', '72791 Claremont Road', '2024-07-21', 'apple_pay'),
(2, 3, '3569815018863462', '2025-01-10', 'jcb', '5 Elka Circle', '2023-11-30', 'venmo'),
(3, 11, '3568529170833269', '2026-02-14', 'jcb', '3 Onsgard Alley', '2023-12-15', 'debit_card'),
(4, 8, '4026705996755944', '2025-02-10', 'visa-electron', '5920 Cascade Alley', '2023-12-01', 'credit_card'),
(5, 14, '490505116813608483', '2028-10-17', 'switch', '40 Stone Corner Way', '2024-06-13', 'venmo'),
(6, 7, '3532767912031866', '2028-01-06', 'jcb', '3 Lakewood Lane', '2024-11-20', 'credit_card'),
(7, 5, '3557048955841655', '2024-08-28', 'jcb', '8 Dapin Place', '2024-05-12', 'credit_card'),
(8, 18, '3549316920398619', '2026-02-01', 'jcb', '5243 Superior Place', '2024-11-16', 'paypal'),
(9, 11, '201525200798513', '2028-09-16', 'diners-club-enroute', '67 Summerview Way', '2024-11-27', 'paypal'),
(10, 19, '3560540743289671', '2027-12-17', 'jcb', '790 Fulton Avenue', '2023-11-29', 'credit_card'),
(11, 16, '3551428148820061', '2027-05-15', 'jcb', '5449 Division Lane', '2024-05-20', 'debit_card'),
(12, 10, '3564284526090263', '2028-05-27', 'jcb', '28873 Northridge Point', '2023-12-24', 'apple_pay'),
(13, 9, '3574145023038287', '2026-04-08', 'jcb', '5 Spaight Way', '2023-12-28', 'venmo'),
(14, 14, '5379544350333653', '2027-03-29', 'mastercard', '9604 Linden Junction', '2024-01-26', 'paypal'),
(15, 15, '589302717678188497', '2024-08-08', 'maestro', '5 Jenifer Place', '2024-06-27', 'debit_card'),
(16, 6, '3532576987969201', '2028-08-18', 'jcb', '02 Nevada Terrace', '2024-02-02', 'credit_card'),
(17, 1, '560222159832685464', '2024-11-29', 'china-unionpay', '9824 Wayridge Terrace', '2024-08-07', 'apple_pay'),
(18, 4, '67712839753382166', '2028-09-10', 'laser', '6773 Lakewood Gardens Terrace', '2024-11-23', 'venmo'),
(19, 20, '3560152240238688', '2026-05-23', 'jcb', '41 Reinke Trail', '2024-05-16', 'paypal'),
(20, 2, '3548359073752894', '2026-08-22', 'jcb', '1 Hovde Junction', '2024-10-12', 'paypal');
```

Dummy data for the social_media table

```sql
-- Insert dummy data into the social_media table
INSERT IGNORE INTO social_media (network_id, user_account_id, account_url) VALUES
(8, 15, 'https://linkedin.com/in/user115'),
(3, 6, 'https://instagram.com/user106'),
(14, 1, 'https://facebook.com/user101'),
(7, 18, 'https://instagram.com/user118'),
(1, 10, 'https://twitter.com/user110'),
(19, 3, 'https://instagram.com/user103'),
(5, 8, 'https://twitter.com/user108'),
(13, 4, 'https://twitter.com/user104'),
(17, 7, 'https://twitter.com/user107'),
(11, 13, 'https://instagram.com/user113'),
(2, 2, 'https://facebook.com/user102'),
(10, 14, 'https://facebook.com/user114'),
(9, 17, 'https://twitter.com/user117'),
(15, 5, 'https://instagram.com/user105'),
(6, 20, 'https://facebook.com/user120'),
(4, 12, 'https://linkedin.com/in/user112'),
(20, 9, 'https://linkedin.com/in/user109'),
(18, 16, 'https://facebook.com/user116'),
(16, 11, 'https://linkedin.com/in/user111'),
(12, 19, 'https://linkedin.com/in/user119');
```

Dummy data for the user_message table

```sql
-- Insert dummy data into the user_message table
INSERT IGNORE INTO user_message (id, sender_id, receiver_id, message_content, status, sent_at_timestamp) VALUES
(1, 5, 6, 'Can we reschedule our meeting?', 'Sent', '2023-09-03 12:14:23'),
(2, 6, 5, 'Sure, what time works for you?', 'Read', '2023-09-03 12:19:41'),
(3, 7, 8, 'I loved your recent post!', 'Sent', '2023-11-22 13:05:37'),
(4, 8, 7, 'Thank you so much!', 'Read', '2023-11-22 13:15:29'),
(5, 17, 18, 'Please share the presentation.', 'Sent', '2023-08-09 18:07:13'),
(6, 18, 17, 'I will send it by EOD.', 'Read', '2023-08-09 18:29:48'),
(7, 1, 2, 'Hello, how are you?', 'Sent', '2023-07-01 10:12:47'),
(8, 2, 1, 'I am good, thanks!', 'Read', '2023-07-01 10:19:33'),
(9, 15, 16, 'Are you free this weekend?', 'Sent', '2023-10-08 17:09:25'),
(10, 16, 15, 'Yes, let's plan something.', 'Read', '2023-10-08 17:23:54'),
(11, 13, 14, 'I have sent you the proposal.', 'Sent', '2023-12-07 16:11:32'),
(12, 14, 13, 'Thanks, I will go through it.', 'Read', '2023-12-07 16:28:49'),
(13, 19, 20, 'Can you assist with the report?', 'Sent', '2023-09-10 19:06:15'),
(14, 20, 19, 'Sure, send me the details.', 'Read', '2023-09-10 19:22:14'),
(15, 11, 12, 'Let's catch up soon.', 'Sent', '2023-06-06 15:08:53'),
(16, 12, 11, 'Sure, let me know when.', 'Read', '2023-06-06 15:25:41'),
(17, 9, 10, 'Please check the document I sent.', 'Sent', '2023-08-15 14:07:37'),
(18, 10, 9, 'Got it, will review and revert.', 'Read', '2023-08-15 14:29:52'),
(19, 3, 4, 'Are you available for a call?', 'Sent', '2023-11-02 11:13:27'),
(20, 4, 3, 'Yes, let me know when.', 'Read', '2023-11-02 11:28:15');
```

Dummy data for the user_notification table

```sql
-- Insert dummy data into the user_notification table
INSERT IGNORE INTO user_notification (notification_id, user_id, notification_type, notification_content, sent_at, is_read) VALUES
(1, 5, 'Refund Processed', 'Your refund has been processed.', '2023-10-14 14:28:32', 1),
(2, 10, 'Promo Alert', 'A new promo code is available.', '2023-11-03 19:33:15', 0),
(3, 17, 'Support Ticket', 'Your support ticket has been updated.', '2023-12-01 10:47:28', 0),
(4, 8, 'Cancellation Confirmed', 'Your booking cancellation is confirmed.', '2023-08-25 17:52:14', 1),
(5, 15, 'Feedback Request', 'Please rate your recent booking.', '2023-09-18 08:49:37', 0),
(6, 4, 'Reminder', 'Your booking starts tomorrow.', '2023-10-22 13:19:43', 1),
(7, 6, 'Security Alert', 'Unusual activity detected on your account.', '2023-11-12 15:42:57', 0),
(8, 14, 'Booking Update', 'Your booking status has changed.', '2023-12-15 23:12:36', 1),
(9, 3, 'Ticket Update', 'Your support ticket has been resolved.', '2023-10-05 12:43:11', 0),
(10, 19, 'Discount Offer', 'Special discounts for holiday bookings.', '2023-09-29 12:16:22', 1),
(11, 1, 'Booking Update', 'Your booking has been confirmed.', '2023-08-11 10:13:49', 1),
(12, 18, 'App Update', 'A new version of the app is available.', '2023-09-20 11:29:07', 1),
(13, 11, 'Payment Failure', 'Your payment could not be processed.', '2023-07-07 20:49:25', 1),
(14, 9, 'Booking Reminder', 'Your booking starts in 2 days.', '2023-10-03 18:17:39', 1),
(15, 13, 'Maintenance Notice', 'Our site will be down for maintenance.', '2023-09-24 22:27:14', 0),
(16, 7, 'New Feature', 'Check out our new search filters.', '2023-10-18 16:25:08', 1),
(17, 16, 'Password Change', 'Your password was changed successfully.', '2023-11-09 09:14:29', 1),
(18, 12, 'Account Update', 'Your account details have been updated.', '2023-12-02 21:18:43', 1),
(19, 2, 'Payment Received', 'We have received your payment.', '2023-09-17 11:34:12', 1),
(20, 20, 'Booking Reminder', 'Your booking starts in 3 days.', '2023-12-20 13:37:55', 1);
```

Dummy data for the support_tickets table

```sql
-- Insert dummy data into the support_tickets table
INSERT IGNORE INTO support_tickets (ticket_id, user_id, subject, description, status, created_at, updated_at) VALUES
(1, 5, 'Refund Request', 'Requesting refund for a cancelled booking', 'Open', '2023-10-12 14:12:15', '2023-10-12 14:33:48'),
(2, 12, 'Slow Performance', 'Website loading very slowly', 'In Progress', '2023-09-22 21:18:27', '2023-09-22 21:39:54'),
(3, 18, 'App Update Issue', 'Unable to update the app on my phone', 'In Progress', '2023-07-05 11:14:36', '2023-07-05 11:31:42'),
(4, 3, 'Account Suspension', 'My account was suspended without reason', 'Pending', '2023-12-01 12:07:12', '2023-12-01 12:21:47'),
(5, 7, 'Feature Request', 'Add a filter for pet-friendly properties', 'Closed', '2023-08-19 16:15:49', '2023-08-19 16:24:15'),
(6, 10, 'Security Concern', 'Suspicious activity on my account', 'Resolved', '2023-11-03 19:09:22', '2023-11-03 19:39:12'),
(7, 16, 'Booking Confirmation', 'Did not receive confirmation email', 'Resolved', '2023-06-30 09:07:12', '2023-06-30 09:47:41'),
(8, 19, 'Broken Link', 'A link on the FAQ page is broken', 'Open', '2023-09-10 12:13:56', '2023-09-10 12:32:18'),
(9, 1, 'Login Issue', 'Unable to log into my account', 'Open', '2023-07-15 10:09:25', '2023-07-15 10:12:43'),
(10, 14, 'Double Charge', 'Charged twice for the same booking', 'Resolved', '2023-12-13 23:05:37', '2023-12-13 23:39:12'),
(11, 2, 'Payment Problem', 'Payment failed while booking a property', 'Resolved', '2023-08-21 11:18:21', '2023-08-21 12:12:37'),
(12, 4, 'Property Listing', 'Need help listing my property', 'In Progress', '2023-11-15 13:14:32', '2023-11-15 13:49:04'),
(13, 8, 'Cancellation Issue', 'Unable to cancel my booking', 'Open', '2023-09-25 17:17:48', '2023-09-25 17:53:11'),
(14, 11, 'Booking Problem', 'Could not complete booking process', 'Open', '2023-10-17 20:06:44', '2023-10-17 20:23:35'),
(15, 6, 'Technical Issue', 'Website not loading on my device', 'Resolved', '2023-06-29 15:13:55', '2023-06-29 15:38:41'),
(16, 13, 'Account Deletion', 'Want to delete my account permanently', 'Closed', '2023-08-30 22:09:41', '2023-08-30 22:36:25'),
(17, 17, 'Payment Refund', 'Refund not received for a cancelled booking', 'Pending', '2023-10-05 10:18:14', '2023-10-05 10:41:22'),
(18, 20, 'Discount Issue', 'Promo code is not being applied', 'Resolved', '2023-11-09 13:15:49', '2023-11-09 13:45:37'),
(19, 15, 'Unresponsive Host', 'Host is not responding to my messages', 'Open', '2023-12-02 08:17:42', '2023-12-02 08:39:27'),
(20, 9, 'Mobile App Bug', 'App crashes when viewing properties', 'Pending', '2023-07-10 18:13:14', '2023-07-10 18:31:49');
```

Dummy data for the user_activity_log table

```sql
-- Insert dummy data into the user_activity_log table
INSERT IGNORE INTO user_activity_log (log_id, user_id, activity_type, activity_timestamp, activity_description) VALUES
(1, 5, 'Logout', '2023-09-03 12:15:42', 'User logged out of the system'),
(2, 12, 'Message Received', '2023-10-06 19:12:23', 'User received a message'),
(3, 19, 'Login', '2023-08-10 10:05:34', 'User logged into the system'),
(4, 2, 'Profile Update', '2023-09-12 09:18:56', 'User updated profile information'),
(5, 8, 'Booking', '2023-07-04 15:25:19', 'User made a booking'),
(6, 13, 'Login', '2023-12-02 20:14:07', 'User logged into the system'),
(7, 3, 'Password Change', '2023-11-02 10:07:29', 'User changed account password'),
(8, 6, 'Message Sent', '2023-09-01 13:03:41', 'User sent a message'),
(9, 17, 'Logout', '2023-08-09 08:11:16', 'User logged out of the system'),
(10, 7, 'Login', '2023-11-04 14:22:51', 'User logged into the system'),
(11, 20, 'Booking', '2023-10-10 11:27:45', 'User made a booking'),
(12, 9, 'Profile Update', '2023-06-05 16:09:34', 'User updated profile information'),
(13, 18, 'Message Sent', '2023-07-09 09:14:48', 'User sent a message'),
(14, 1, 'Login', '2023-08-01 08:23:37', 'User logged into the system'),
(15, 10, 'Payment', '2023-11-05 17:18:52', 'User made a payment'),
(16, 15, 'Password Change', '2023-07-08 22:10:42', 'User changed account password'),
(17, 16, 'Payment', '2023-10-08 23:11:25', 'User made a payment'),
(18, 14, 'Profile Update', '2023-09-07 21:15:58', 'User updated profile information'),
(19, 4, 'Payment', '2023-12-03 11:03:17', 'User made a payment'),
(20, 11, 'Logout', '2023-06-06 18:21:45', 'User logged out of the system');
```

Database Management Functionality

Database Management

The database behaves similarly to an Airbnb-like platform, simulating behaviour with relatively similar content, managing three key of users, who either host or rent a place, a booking and social system that allows to book a property, and finally the properties themself that can be chosen and booked, as well as that contain all the main information about themself. The main functionalities are:

1. **User Management**
    - Contains user data that is used by the platform to manage users.
    - Users have roles (Guest, Host, or Admin), and the system ensures role-based access and operations.
2. **Property Management**
    - Hosts can list, update, or remove properties.
    - Properties include details such as location, pricing, amenities, images, services and policies.
3. **Booking Management**
    - Guests can book available properties.
    - Booking details include date, number of guests, and status, as well as payment information and booking history.
4. **Payment Management**
    - Payments are tracked for each booking.
    - Supports payment details like method, amount, and status.
5. **Review System**
    - Guests can leave reviews and ratings.
6. **Support and Notifications**
    - Users can create support tickets as well as be tracked using notifications and activity monitoring.
    - Notification management keeps users updated on important events.
7. **Messaging System**
    - Facilitates direct communication between guests and hosts.
8. **Location Management**
    - Properties are linked hierarchically to addresses, cities, countries, and regions.
    - Supports location-based filtering and search functionality.

Metadata

The database contains 31 tables. The three main tables that connect to most of the other tables are **user_account**, **booking** and **property**. The decision to organize the tables depending on the topics that they are dealing made was made: User Management, Property Management, Booking Management, Support, and Social Interaction.

- **user_account**: Holds user details with expected entries for hosts and guests.

- **property**: Includes property-related entries.

- **booking**: Contains reservation records including user and property references.

The dummy data that was inserted is 20 records per table, with the total number of records being 30*20 + 3 (roles) = 603 records. The final size of the structure and data dump is 61.5KB, which can be found in the Phase 3 folder under the name “Structure and Data Dump”.

The main areas that the table can be split into are

User account and management: *user_account, roles, user_roles, user_language, language.*

Support and Notifications (ticketing): *support_tickets, user_notification*.

Social and communication: *social_media, user_message, payment_info, user_activity_log*

Booking and Payments: *booking, booking_status_history, payment, booking_guideline, host, review.*

Property Management: *property, cancellation_policies, property_utility, property_policy, property_policy_map, property_images, amenities, property_amenities_map, property_service, property_service_map.*

Location: address, city, country, region.

Making of the SQL Airbnb Datamart IU project.

DLBDSPBDM01

1) Conception

The primary idea of the first phase was to design and create an Entity Relationship Model of the project using either MySQL, Postgres or DBeaver. After carefully analyzing the three applications, I decided to try DBeaver, where the initial blueprint for the ERD was created, but in the later development a decision to eventually switch to MySQL was made. The much user-friendlier interface as well as an opportunity to edit the tables using the reverse engineer wizard in MySQL that allowed to adjust the tables both via the modelling interface as well as via the tables interface, proved to be an irreplaceable tool during the development, as I spent most of the time in this stage, planning how the final result should look like and how to connect the foreign keys. Therefore, a conclusion on making three main tables was reached: User Account, Booking and Property. The rest of the tables were derived from the and facilitated the efficient communication between them. The chosen methodology involved the following steps:

- Creation of the ERM to identify entities, attributes and relationships. Adjusting the relationship between them and defining them using the crow-foot notation.
- Normalization of the database schema following the received feedback, as well as reducing the complexity of the design.
- Implementation of the database using MySQL Reverse Engineer functionality.
- Execution of sample queries to validate the database's capabilities.

2) Development

The second phase proved to be more complicated than the first one, since I also encountered several issues that appeared due to poor planning of the first stage:

- Some of the tables used a simple INT data type, while others had INT UNSIGNED data type. This became an issue when it was necessary to link foreign keys.
- Populating the tables with data was a challenging task, because I didn't know that there is a specific sequence of doing it, so creating the file with the data to be inserted was not as straightforward as was initially anticipated. The result is a specific sequence of tables to be filled first. Each dummy data had to be first synthetized using Mockaroo website and then handled manually to make sure that the input was accepted correctly and sometimes text queries had to be appended fully manually.

- Figuring out the relationships between the users (host, guest, admin) was simple at first, but became a lot harder when the idea to assign an admin to a ticket was considered. Ultimately, this feature was scrapped, because it proved to alter the design of the user_account table and complicate the normalization.
- Sample queries were executed and documented in this phase to demonstrate the full functionality of the technical solution. Such queries that either retrieve all the available properties for a specific attribute or the whole tables are documented in the file.
- The implemented database meets the functional requirements and fulfills the roles that were defined in the conception phase.

3) Finalization

During the finalization phase, a retrospective analysis on the overall development procedure as well as what could've been done differently was conducted with the following conclusions:

- Normalization was done to a degree that was possible. Further normalization proved to be complicated and ended up messing with the overall design and simplicity of the final solution. As an example, special tables with composite primary keys were defined for property amenities, property services and property policies to solve some of the issues.
- One of the most important features that was missed during the design was definition of what should happen to the columns on deletion. Since the cascading rules were not defined with the creation of the tables, it became really complicated to experiment with the dummy data and manipulations with it. Therefore, the tables don't support deleting data and were created with the goal of storing information long term for the sake of traceability.
- Ensuring data integrity via foreign keys also proved to be a challenge, especially when older tables had an integer that was signed, and the newer tables had an unsigned integer. This was adjusted for all the tables.
- The process of designing and implementing the database improved my understanding of ER modelling and SQL. The most important step of the design process is the initial concept and rigorous normalization, which was achieved with the help and feedback of the professor. Testing filling out the data with dummy data, as well as writing queries showed possible design flaws that were adjusted for by changing the data in the tables and foreign keys.