Final project for the course of Database Systems [DT0347]
a.a. 2024/2025

Introduction

This report explains how a database was planned and built for a new Travel Recommendation and Review Platform. The main goal of this platform is to help users explore different places to visit, stay, and enjoy by reading reviews, ratings, and getting personalized suggestions. The database is designed to store and manage details like user profiles, travel locations, preferences, reviews, and custom travel plans. It also supports important features that are useful for both users and system administrators.

Requirements analysis

This section outlines the high-level functional and non-functional requirements, along with the specific use cases that guided the design and implementation of the database for the Travel Recommendation and Review Platform. Any initial ambiguities from the project brief have been addressed and clarified here to ensure a precise understanding of the system's needs.

High-Level Requirements:

R1 – User Registration and Profile Management: The system must enable users to register by providing essential personal details, including their full name and email address. Users should also have the option to specify personal preferences, such as their preferred travel destinations, activities, or budget, which will be utilized for personalized services.

R2 – Destination and Experience Catalog: The platform is required to present a comprehensive catalog of travel entities. This includes a variety of destinations, hotels, attractions, and pre-defined travel packages, allowing users to explore different travel options.

R3 – Detailed Entity Information: For each travel entity listed in the catalog, the database must store extensive details. This includes descriptive text, user-generated reviews and associated ratings (on a 1-to-5 scale), current pricing information, and real-time availability.

R4 – User-Generated Content Submission: Users who have visited a location must be able to submit their own reviews and ratings. This functionality requires capturing details such as the date of their visit, the nature of their travel experience (e.g., solo, family, romantic, business), and any specific recommendations they wish to provide.

R5 – Itinerary Creation and Sharing: The platform must support users in creating personalized travel itineraries. These itineraries can be saved for private use or be made publicly accessible, facilitating sharing among the user community.

R6 – User Activity Tracking for Recommendations: A core requirement is to continuously track user interactions and historical data, specifically their visited destinations, submitted reviews, and declared preferences. This accumulated data forms the basis for the platform's recommendation engine.

R7 – Personalized Recommendation Engine: The system needs to implement an intelligent recommendation system. This system should generate tailored suggestions for destinations or experiences by analyzing factors such as the ratings provided by similar users, shared interests, overall popularity trends, or seasonal relevance.

Use Cases:

U1 – Retrieve High-Rated Destinations: This use case allows a user to query and retrieve a list of destinations or attractions they have reviewed and rated above a specified minimum threshold (e.g., a rating of 4 out of 5 or higher).

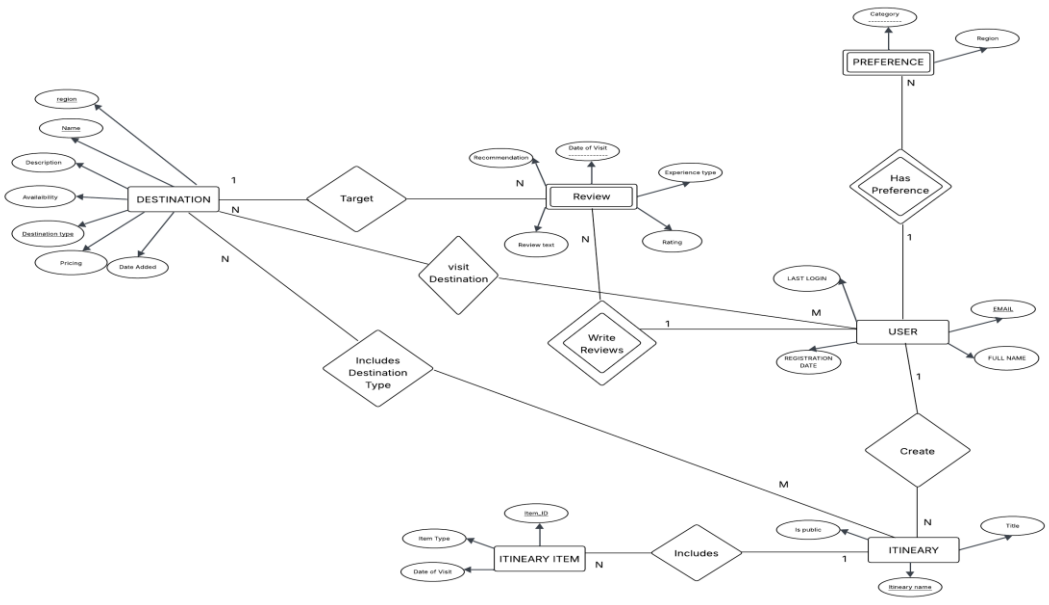
U2 – Discover Recently Added Attractions/Accommodations: Users should be able to identify and explore new accommodations or attractions that have been recently added to the platform. This discovery process can be further refined by filtering results based on their previously stated favorite travel categories or preferred geographical regions.

U3 – Identify Inactive Users and Recent Activity (Administrator): The system must provide administrators with the capability to identify users who have not logged in for a specified duration, classifying them as inactive. For these inactive accounts, the system should also display their last recorded travel-related activity, such as their most recent review or visited destination.

U4 – Generate Tailored Recommendations: This use case focuses on the system's ability to provide personalized recommendations. These suggestions are formulated by examining the user's past travel history, considering highly-rated locations by other users with similar profiles, and identifying destinations that are currently trending.

Conceptual design

The conceptual design phase involved creating an Entity-Relationship (ER) Diagram to visually represent the database structure, independent of any specific database management system. This design uses Chen's notation and models the real-world entities involved in a travel recommendation and review platform, along with the relationships between them.



The entity types and relationship types identified, along with the requirement(s) or use case(s) that led to design decisions are:

Entity types:

- User [R1, U3] Attributes: Email (Primary Key), Full Name, Registration Date, Last Login.
- Destination [R2, R3, U2] Attributes: Name (Primary Key, part of composite), Destination Type (Primary Key, part of composite), Description, Availability, Pricing, Date Added, Region.
- Preference [R1, R7, U4] Attributes: Category (Primary Key, part of composite), Region (Primary Key, part of composite).
- Review [R4, U1] Attributes: Review ID (Primary Key), Rating, Review Text, Date Of Visit, Experience Type, Recommendation.
- Itinerary [R5] Attributes: Itinerary Name (Primary Key, part of composite), Title, Is Public.
- Itinerary Item [R5] Attributes: Item ID (Primary Key), ItemType, Visit Date.

Relationship types:

- Has Preference (between User and Preference) (M:N) [R1, R7 , U4]
- Write reviews (between User and Review) (1:N) [R4]
- Target (between Destination and Review) (1:N) [R3, R4]
- Visit Destination (between User and Destination) (M:N) [R6,R7,U4]
- Create (between User and Itinerary) (1:N) [R5]
- Includes Destination Type (between Itinerary and Destination) (M:N) [R5]
- Includes (between Itinerary and Itinerary Item) (1:N) [R5]

Logical design

The logical design phase involved translating the conceptual Entity-Relationship (ER) Diagram into a detailed relational model. This model consists of a set of interconnected tables (relations), their attributes (columns), and the definition of primary and foreign keys to enforce data integrity. This process systematically converts the entities and relationships identified in your ER diagram into a structured format. The mapping procedure is as follows:

Mapping of Strong Entity Types:

Procedure: Each strong entity type from the ER Diagram is directly transformed into a distinct relation (table) in the relational model. All simple attributes of the entity become columns in this table. The chosen primary key(s) of the entity become the primary key(s) of the respective table.

Application:

User: Mapped to the User table.

Attributes: Email (VARCHAR(150), PK, NOT NULL), Full Name (VARCHAR(255), NOT NULL), Registration Date (DATE, NOT NULL), Last Login (DATETIME).

Primary Key: Email.

Destination: Mapped to the Destination table.

Attributes: Name (VARCHAR(100), PK, NOT NULL), Destination Type (VARCHAR(100), PK, NOT NULL), Description (TEXT), Availability (VARCHAR(255)), Pricing (DECIMAL(10, 2)), Date Added (DATE, NOT NULL), Region (VARCHAR(100)).

Primary Key: Composite key (Name, Destination Type).

Preference: Mapped to the Preference table.

Attributes: Category (VARCHAR(100), PK, NOT NULL), Region (VARCHAR(100), PK, NOT NULL).

Primary Key: Composite key (Category, Region).

Review: Mapped to the Review table.

Attributes: Review ID (INT, PK, AUTO_INCREMENT), Review Text (TEXT), Date Of Visit (DATE, NOT NULL), Experience Type (VARCHAR(255)), Rating (INT, NOT NULL, CHECK 1-5), Recommendation (TEXT).

Primary Key: Review ID.

Itinerary: Mapped to the Itinerary table.

Attributes: Itinerary Name (VARCHAR(100), PK, NOT NULL), Title (VARCHAR(255), NOT NULL), Is Public (BOOLEAN, NOT NULL). (Note: User Email will become part of its composite primary key due to the 'Create' relationship mapping).

Primary Key: Composite key (Itinerary Name, User Email).

Itinerary Item: Mapped to the Itinerary Item table.

Attributes: Item ID (INT, PK, AUTO_INCREMENT), ItemType (VARCHAR(255), NOT NULL), Visit Date (DATE, NOT NULL).

Primary Key: Item ID.

Mapping of 1:N Relationship Types:

Procedure: For a 1-to-Many (1:N) relationship between two entities (e.g., Entity A on the '1' side and Entity B on the 'N' side), the primary key of the entity on the '1' side (Entity A) is added as a foreign key to the table representing the entity on the 'N' side (Entity B).

Application:

Write Reviews (User 1:N Review): The primary key of User (Email) is added as a foreign key (User Email VARCHAR(150), FK) to the Review table.

Target (Destination 1:N Review): The composite primary key of Destination (Name, Destination Type) is added as a composite foreign key (Destination Name VARCHAR(100), FK; Destination Type VARCHAR(100), FK) to the Review table.

Create (User 1:N Itinerary): The primary key of User (Email) is added as a foreign key (User Email VARCHAR(150), FK) to the Itinerary table. As indicated by the ERD, this User Email also becomes a component of the Itinerary table's composite primary key.

Includes (Itinerary 1:N Itinerary Item): The composite primary key of Itinerary (Itinerary Name, User Email) is added as a composite foreign key (Itinerary Name VARCHAR(100), FK; User Email VARCHAR(150), FK) to the Itinerary Item table.

Mapping of M:N Relationship Types:

Procedure: A Many-to-Many (M:N) relationship type is mapped to a new, separate relation (often called a junction table or associative entity). This new relation's primary key is typically formed by the combination of the primary keys of the two participating entity types, and these combined keys also serve as foreign keys referencing their original tables. Any attributes explicitly associated with the M:N relationship itself also become columns in this new table.

Application:

Has Preference (User M:N Preference): Mapped to a new table named Has Preference.

Attributes: User Email (VARCHAR(150), PK, FK), Preference Category (VARCHAR(100), PK, FK), Preference Region (VARCHAR(100), PK, FK).

Primary Key: Composite key (User Email, Preference Category, Preference Region).

Foreign Keys: User Email references User(Email). (Preference Category, Preference Region) references Preference(Category, Region).

Visit Destination (User M:N Destination): Mapped to a new table named Visited Destination. This relationship also has an attribute (Date of Visit).

Attributes: User Email (VARCHAR(150), PK, FK), Destination Name (VARCHAR(100), PK, FK), Destination Type (VARCHAR(100), PK, FK), Date Of Visit (DATE, NOT NULL).

Primary Key: Composite key (User Email, Destination Name, Destination Type).

Foreign Keys: User Email references User(Email). (Destination Name, Destination Type) references Destination(Name, Destination Type).

Includes Destination Type (Itinerary M:N Destination): Mapped to a new table named Includes Destination Type.

Attributes: Itinerary Name (VARCHAR(100), PK, FK), User Email (VARCHAR(150), PK, FK), Destination Name (VARCHAR(100), PK, FK), Destination Type (VARCHAR(100), PK, FK).

Primary Key: Composite key (Itinerary Name, User Email, Destination Name, Destination Type).

Foreign Keys: (Itinerary Name, User Email) references Itinerary(Itinerary Name, User Email). (Destination Name, Destination Type) references Destination(Name, Destination Type).

Handling of Attributes:

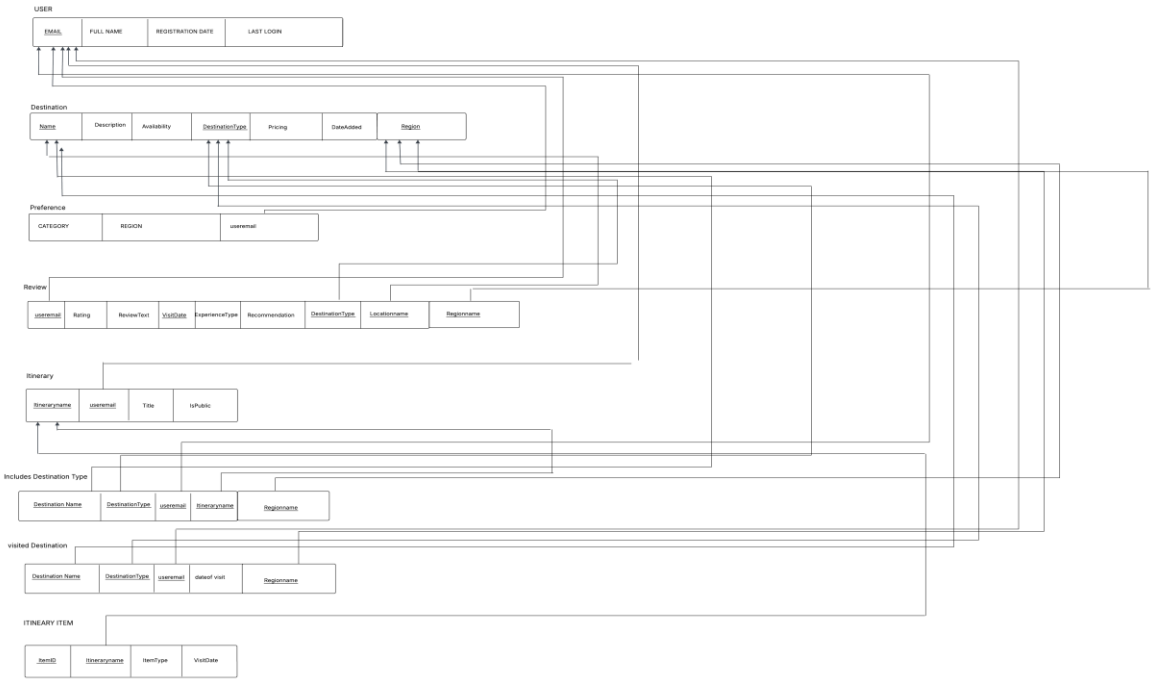
Simple Attributes: All simple attributes (e.g., Full Name, Description, Title) become direct columns in their respective tables.

Composite Attributes: In your ERD, composite keys like (Name, Destination Type) for Destination and (Category, Region) for Preference are naturally mapped as individual columns that together form the primary key of their table. The ERD does not show other composite attributes that would require decomposition into multiple columns.

Multi-valued Attributes: No multi-valued attributes were explicitly identified in your ER Diagram that would necessitate the creation of separate tables.

Final Relational Model

The detailed representation of the final relational model, including tables, primary keys, foreign keys, and attribute data domains, is provided visually in the image:



SQL schema implementation

This phase involves meticulously specifying column names, appropriate data types, defining primary keys to ensure unique identification of records, establishing foreign keys to maintain referential integrity between tables, and implementing any other necessary constraints such as NOT NULL to ensure data presence or CHECK constraints for data validity.

This part is about turning design ideas into a real, working database structure. This is achieved by writing CREATE TABLE commands for each component of the design.

In these commands, all the necessary information for each table is listed, such as the names of the columns (e.g., "Email," "Description," "Rating"). MySQL is also instructed on what kind of information each column should hold, like VARCHAR for text, DATE for dates, or INT for numbers.

A very important part is setting up primary keys. These function like unique ID numbers for each entry in a table, ensuring every piece of information can be found and is distinct. Tables are also linked together using foreign keys. These links ensure that data stays connected and correct across different tables. For example, a review must always be linked to a real user and a real destination. The ON DELETE CASCADE rule is also included; this specifies that if something important (like a user) is deleted, then any related information (like their reviews) will automatically be deleted too. This keeps the database tidy and prevents old, useless information from sticking around.

This SQL code serves as the blueprint that builds the entire travel platform database, ensuring everything is organized, connected correctly, and ready for all your travel recommendations and reviews.

Files of .SQL has been attached in the folder.

Use cases implementation

Implement the use cases in SQL (MySQL compliant). For each use case, write the SQL statement(s) that implements it. Also provide the SQL statements in a separate text file during your submission.

These use cases have the additional purpose of validating your design. If, at this point, you understand that any of these use cases cannot be realized given your current database, go back to the earlier design phases to change it accordingly.

The "Use Cases Implementation" phase is crucial for demonstrating the functionality and validating the design of the database. In this stage, each defined use case is translated into one or more SQL statements. These statements serve as a direct application of the database schema, proving its capability to retrieve, manipulate, and organize data according to the platform's operational requirements. All SQL statements provided are compliant with MySQL syntax and are designed to be directly executable.

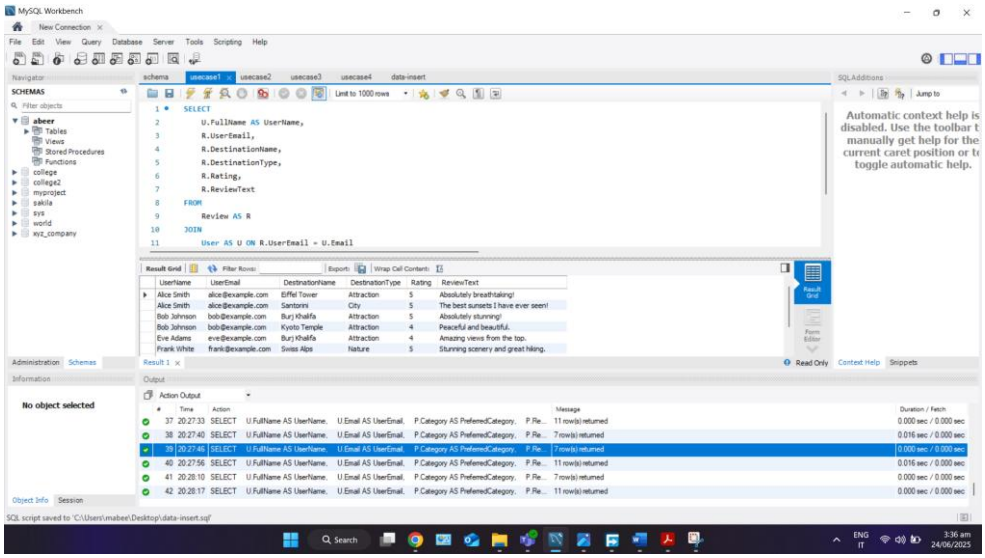
This section presents the implementation for each use case, along with a brief explanation of how each query addresses its specific requirement.

Here's the content for the "Use Cases Implementation" section of your report. You can copy and paste this directly into your Word file.

SQL Statements for Use Cases:

U1 – List High-Rated Destinations:

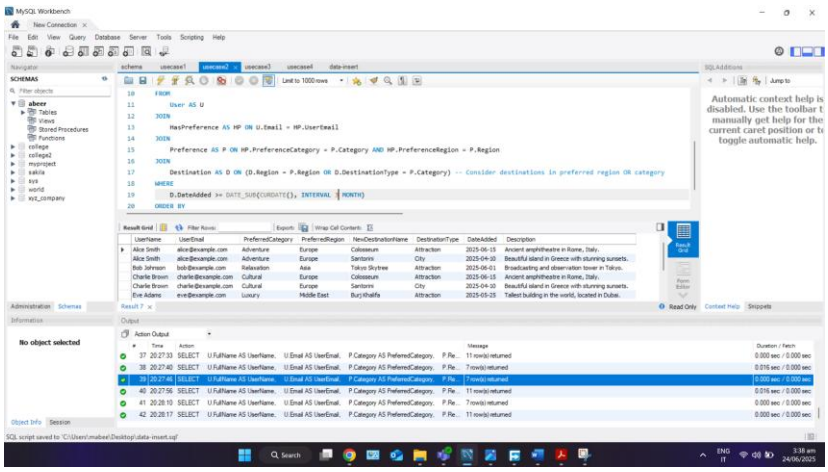
Users can retrieve a list of destinations or attractions that they have personally rated above a specified threshold (e.g., places rated at least 4 out of 5 stars). Screen shot attached:



Explanation: This query joins the Review table with the User table to display the full name of the user who submitted the review. It filters for reviews where the Rating is 4 or higher and orders the results by the user's name and then by rating in descending order.

U2 – Discover New Attractions/Accommodations:

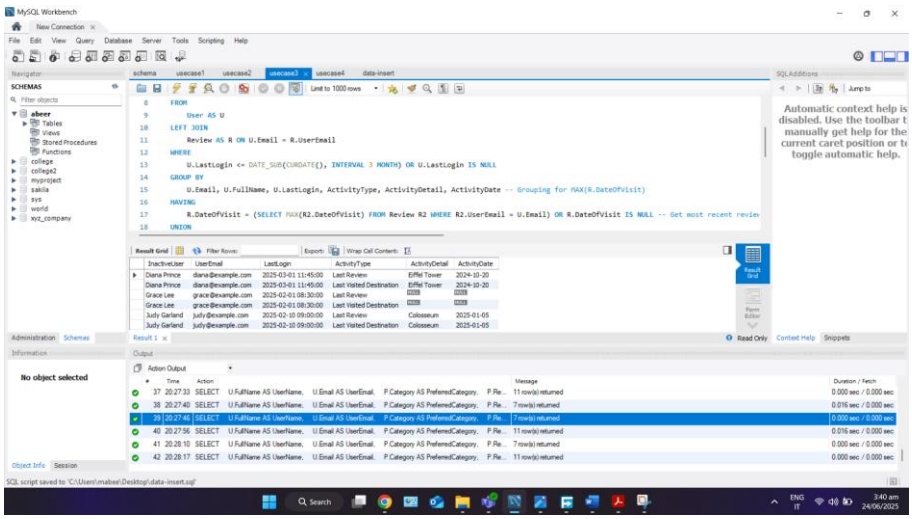
Users can find newly added accommodation or attractions (e.g., those added within the last few months). This discovery process can be refined by filtering based on their favorite travel categories or preferred regions.



Explanation: This query identifies new destinations relevant to a user's preferences. It links Users with their Preferences through Has Preference, and then finds Destinations that match either the preferred Region or Category (Destination Type). The WHERE clause filters for destinations added within the last six months from the current date, ensuring "new" discoveries.

U3 – Identify Inactive Users & Their Activity (Administrator Use Case):

Administrators need the ability to identify users who have been inactive (i.e., have not logged in for a defined period). For these inactive users, the system should also be able to retrieve details of their most recent travel-related activity recorded on the platform (e.g., their last submitted review or last recorded visited destination).

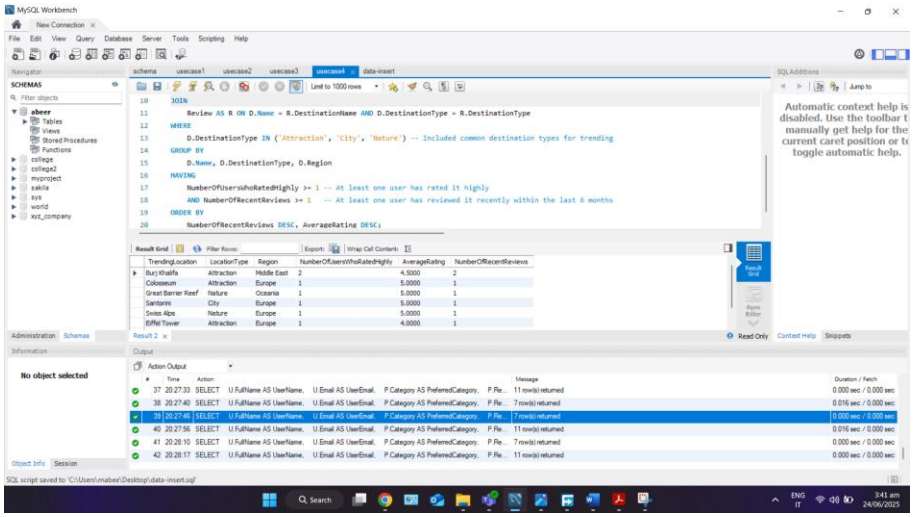


Explanation: This complex query identifies users whose LastLogin date is older than three months or is null, categorizing them as inactive. It then uses UNION to combine results: one part finds their most recent Review activity, and the other finds their most recent VisitedDestination activity. The

GROUP BY and HAVING clauses ensure that only the single most recent activity for each inactive user is returned across both activity types, providing administrators with a quick overview of their last interaction.

U4 – Generate Personalized Recommendations:

The system must be able to provide tailored recommendations for destinations or experiences to individual users. These recommendations should leverage insights from similarities in user travel histories, highly rated locations by users with comparable interests, and identification of currently trending destinations.



Explanation: This query focuses on identifying trending destinations. It joins Destination with Review to analyze rating and visit patterns. It counts the number of unique users who have rated a destination highly (4 or 5 stars) and the number of unique users who have reviewed it recently (within the last six months). Destinations are grouped by their name, type, and region. The HAVING clause ensures only destinations with at least one high rating and at least one recent review are considered "trending," ordered by recent reviews and then average rating.

Summary:

The successful execution of these use case queries confirms the validity and robustness of the database design, demonstrating its capability to meet the specified functional requirements of the Travel Recommendation and Review Platform. As specified in the project guidelines, these SQL statements are also provided as a separate text file (.sql) for submission.