Tour Recommendation System - Database Design

1. Introduction

The Tour Recommendation System database is designed to store and manage information related to tourist destinations, hotels, and users. It enables users to explore cities, places of interest, and accommodations while providing recommendations based on popularity and user preferences.

2. Database Schema Overview

The system consists of six primary models:

1. **CustomUser** - Manages user-related data.
2. **City** - Stores information about various cities.
3. **Place** - Represents tourist places in different cities.
4. **PlaceFeatureImages** - Stores additional images of tourist places.
5. **Hotel** - Stores hotel details in different cities.
6. **HotelFeatureImages** - Manages images related to hotels.

---

3. Entity Relationship Diagram (ERD)

The database follows a **relational structure** with `One-to-Many` relationships:

- **City** ↔ **Place** (One City → Many Places)
- **City** ↔ **Hotel** (One City → Many Hotels)
- **Place** ↔ **PlaceFeatureImages** (One Place → Many Images)
- **Hotel** ↔ **HotelFeatureImages** (One Hotel → Many Images)

---

4. Table Definitions

CustomUser

Field	Type	Description
`id`	AutoField (PK)	Unique user ID
`email`	EmailField	Unique user email (used for login)
`phone`	CharField(100)	User's phone number
`address`	TextField	User's address
`user_profile`	ImageField	Profile picture

- **Primary Key**: `id`

- **Unique Constraint**: `email`

City

Field	Type	Description
`id`	AutoField (PK)	Unique city ID
`city`	CharField(100)	City name
`city_image`	ImageField	Image of the city
`city_desc`	TextField	Description of the city
`city_slug`	SlugField	URL-friendly slug
`is_popular_place`	BooleanField	Flag for popular cities

- **Primary Key**: `id`
- **Unique Constraint**: `city_slug`

Place

Field	Type	Description
`id`	AutoField (PK)	Unique place ID
`city_id`	ForeignKey (City)	Associated city
`place_name`	CharField(100)	Name of the place
`place_image`	ImageField	Main image of the place
`place_desc`	TextField	Description of the place
`place_address`	CharField(100)	Address of the place
`place_price`	PositiveIntegerField	Cost for visiting the place
`is_top_tour_package`	BooleanField	Flag for top packages
`slug`	SlugField	URL-friendly identifier

- **Primary Key**: `id`
- **Foreign Key**: `city_id` → `City(id)`
- **Unique Constraint**: `slug`

PlaceFeatureImages

Field	Type	Description
`id`	AutoField (PK)	Unique image ID
`place_id`	ForeignKey (Place)	Associated place
`images`	ImageField	Additional place images

- **Primary Key**: `id`
- **Foreign Key**: `place_id` → `Place(id)`

Hotel

Field	Type	Description
`id`	AutoField (PK)	Unique hotel ID
`city_id`	ForeignKey (City)	Associated city
`hotel_name`	CharField(100)	Name of the hotel
`hotel_address`	CharField(100)	Address of the hotel
`hotel_desc`	TextField	Description of the hotel
`hotel_image`	ImageField	Main image of the hotel
`hotel_price`	PositiveIntegerField	Room price per night
`hotel_rooms`	PositiveIntegerField	Number of available rooms
`is_popular_hotel`	BooleanField	Flag for popular hotels
`slug`	SlugField	URL-friendly identifier

- **Primary Key**: `id`
- **Foreign Key**: `city_id` → `City(id)`
- **Unique Constraint**: `slug`

HotelFeatureImages

Field	Type	Description
`id`	AutoField (PK)	Unique image ID
`hotel_id`	ForeignKey (Hotel)	Associated hotel
`image`	ImageField	Additional hotel images

- **Primary Key**: `id`
- **Foreign Key**: `hotel_id` → `Hotel(id)`

---

5. Data Integrity & Constraints

- **Unique constraints**: Ensures unique slugs for URLs.
- **Foreign Key constraints**: Maintains relationships between cities, places, and hotels.
- **Default values**: Prevents null errors for fields like `is_popular_hotel` and `is_top_tour_package`.

---

6. Possible Enhancements

To improve the database design, you might consider:

1. **User Reviews & Ratings**: Add a `Review` model to allow users to rate hotels and places.
2. **Tour Packages**: A new model to group places into curated travel packages.
3. **Bookings**: A `Booking` model to handle hotel and tour reservations.

4. **User Preferences**: Track user interactions to offer better recommendations.
5. **Location-based Search**: Implement geolocation to find places near the user.

---

7. Conclusion

This database structure efficiently manages tourist data, offering an organized way to store information about cities, tourist places, hotels, and user preferences. The relationships ensure efficient data retrieval for a smooth user experience.