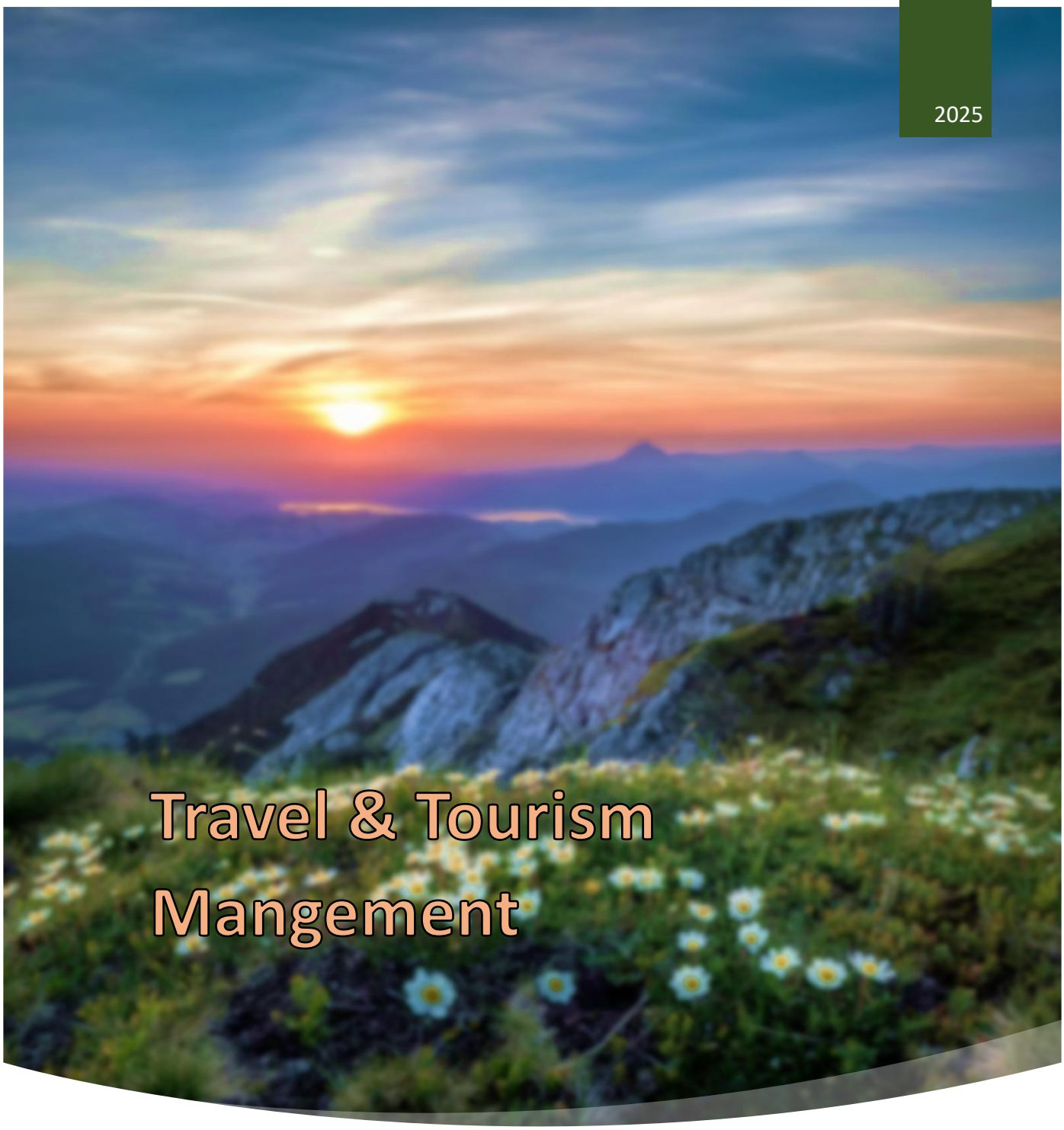
2025



Travel & Tourism Mangement

Vansh Parmar
PRN: 8023051076
Seat No.: 486141
Sub: Database Management System

THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA
सत्यं शिवं सुन्दरम्
Estd. 1949

INDEX

1. OVERVIEW OF PROJECT……2

2. SCHEMA DIAGRAM……4

3. E-R DIAGRAM……5

4. NORMALIZATION (TABLES)……7

5. PL/SQL (TRIGGER ,FUNCTION ,PROCEDURES)….11

6. Database Tables Display……15

Travel & Toursim Mangement System

Overview

- The main objective of the Travel and Tourism Management System is to manage the details of Customer, Hotel, Booking, Cancellation and Tourism places. It manages all the information about Users, Hotel, packages etc. The project is totally built at administrative and customer level application but only the administrator is guaranteed the access to the backend database. The purpose of this project is to build an application program to reduce the manual work for managing Tourists, Booking, places etc.

- This application will help in accessing the information related to the travel to the particular destination with great ease. The users can track the information related to their tours with great ease through this application. The travel agency information can also be obtained through this application.

- Through this system, the propose system is highly automated and makes the travelling activities much easier and flexible. The user can get the very right information at the very right time. This system will include all the necessary fields which are required during online reservation time. This system will be easy to use and can be used by any person. The basic idea behind this project is to save data in a central database which can be accessed by any authorize person to get information and saves time and burden which are being faced by their customers.

- Admin and Customer both can access and modify the information stored in the database of this system, this includes adding and updating of details, and it will give accurate information and simplifies manual work

and also it minimizes the documentation related work. Provides up to date information. Finally booking confirmation notification will be send to the users.

- Tourists can register by providing personal details, make new reservation and book only one hotel and package and can make cancellation.

IDE used: MYSQL Workbench

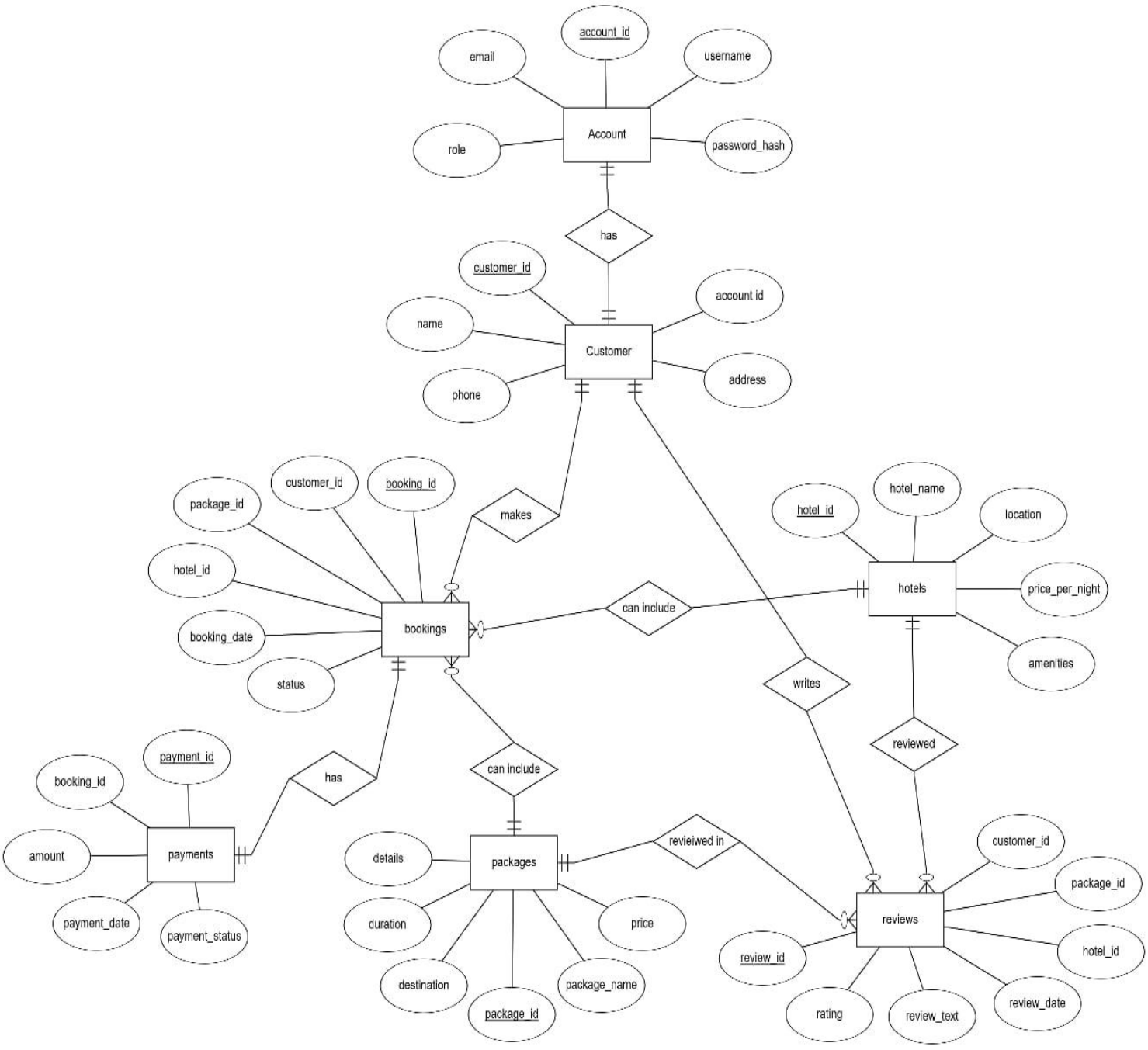
SCHEMA DIAGRAM

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data

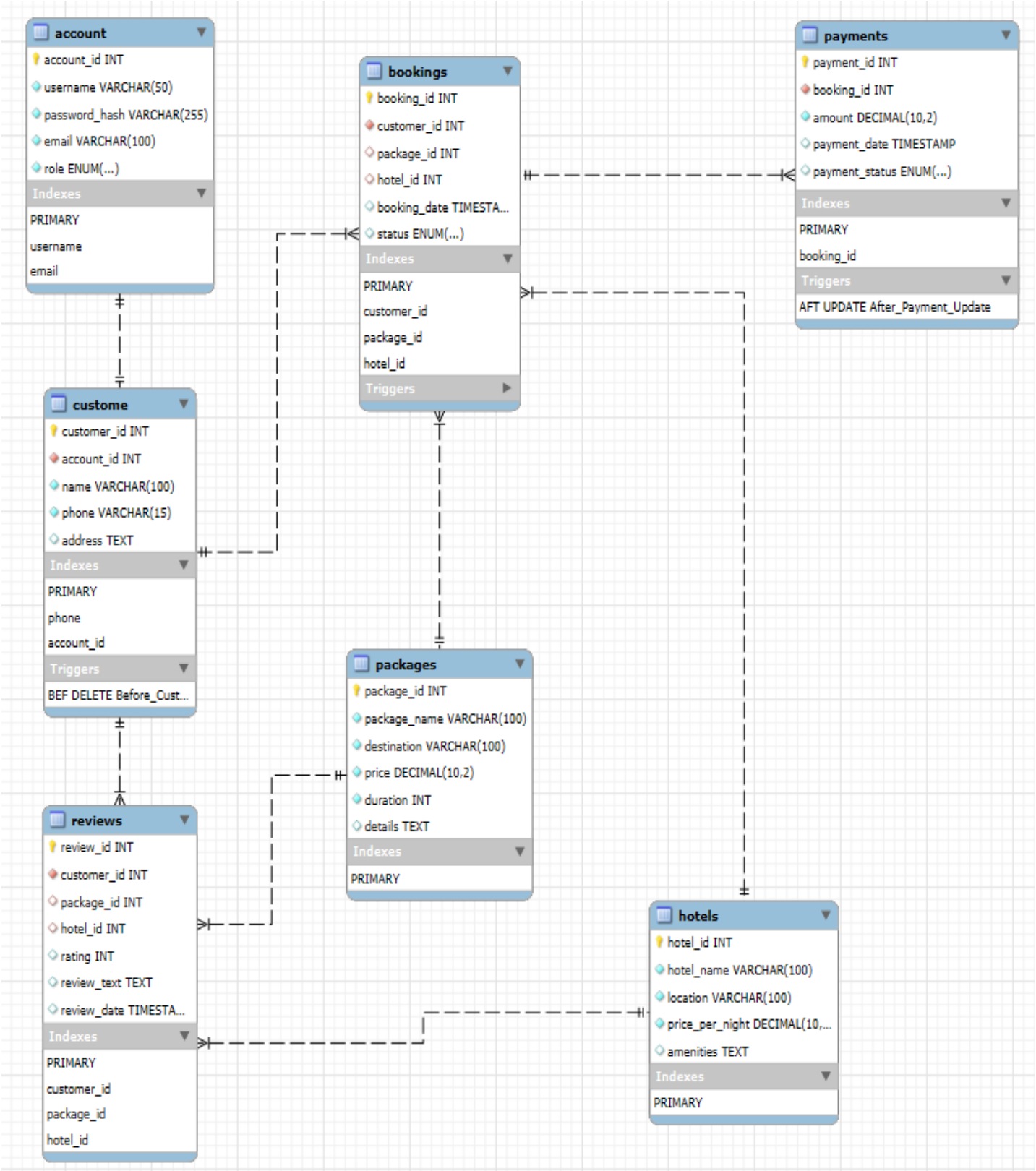
A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

E-R DIAGRAM

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.



E-R Diagram generated by MYSQL Workbench



NORMALIZATION (TABLES)

Normalization is a process in database design that organizes data into smaller, related tables to reduce redundancy and improve data integrity. By ensuring that each table contains data about a single subject and linking them with keys, normalization minimizes data repetition and avoids inconsistencies. This makes the database more efficient, easier to maintain, and less prone to errors.

Normalization for Tables in the Travel & Tourism Management System

1. Account Table

- **Purpose:** Stores user login information such as username, password, email, and user role.
- **1NF:**
    - Has a unique key (account_id) for each user.
    - Every column (username, email, etc.) holds a single value.
- **2NF:**
    - All the details (username, password_hash, email, role) fully depend on account_id.
- **3NF:**
    - There are no extra attributes that depend on other non-key columns.
    - Every detail is directly related only to the user's unique ID.

---

2. Customer Table

- **Purpose:** Holds customer-specific details such as name, phone, and address, and connects to the Account table.
- **1NF:**
    - Contains a unique customer_id for each customer.

    - Each column holds one simple value (e.g., one phone number per customer).
- **2NF:**
    - Every piece of data in this table (name, phone, address) depends on customer_id.
- **3NF:**
    - There are no transitive dependencies; customer details depend directly on customer_id only.

---

3. Packages Table

- **Purpose:** Contains travel package details like package name, destination, price, duration, and additional details.
- **1NF:**
    - Has a unique package_id for every package.
    - Each column stores atomic values.
- **2NF:**
    - All attributes such as package name, destination, price, etc., depend fully on package_id.
- **3NF:**
    - There are no non-key attributes depending on other non-key attributes.
    - Each package detail is directly related to its own unique ID.

---

4. Hotels Table

- **Purpose:** Stores hotel information such as hotel name, location, price per night, and amenities.
- **1NF:**
    - Each hotel has a unique hotel_id.

    - All data is stored in individual columns with a single value.
- **2NF:**
    - All hotel information is completely dependent on hotel_id.
- **3NF:**
    - There are no transitive dependencies.
    - Every attribute (location, price_per_night, amenities) is directly tied to hotel_id.

---

5. Bookings Table

- **Purpose:** Records booking details including which customer booked a package or hotel, and the booking status.
- **1NF:**
    - Uses a unique booking_id for each booking.
    - Contains atomic values in every column.
- **2NF:**
    - All booking details depend fully on booking_id.
    - Foreign keys like customer_id, package_id, and hotel_id ensure each booking is linked to the correct record in other tables.
- **3NF:**
    - No non-key attribute depends on another non-key attribute.
    - Booking status and booking date are directly dependent on booking_id.

---

6. Payments Table

- **Purpose:** Manages payment information for each booking, including the amount, date, and payment status.
- **1NF:**
    - Each payment record has a unique payment_id.

- Every field (amount, payment_date, etc.) holds a single value.

- **2NF:**
  - All payment details are entirely dependent on payment_id.
- **3NF:**
  - There are no additional dependencies between non-key attributes.
  - Each payment detail is directly related to its unique ID.

---

7. Reviews Table

- **Purpose:** Contains customer reviews for packages and hotels, including ratings and review text.
- **1NF:**
  - Has a unique review_id for each review.
  - All columns hold one piece of information (e.g., one rating value, one review text).
- **2NF:**
  - Each review detail (rating, review_text) depends fully on review_id.
  - The foreign keys (customer_id, package_id, hotel_id) ensure correct linking.
- **3NF:**
  - There are no transitive dependencies.
  - Each review attribute is directly associated with the unique review ID.

PL/SQL (TRIGGER ,FUNCTION ,PROCEDURES)

- **Stored Procedures**

  - **AddCustomer**: This procedure creates a new customer account by inserting a record into the Account table with a default role of 'Customer' and then retrieves the generated account ID using the LAST_INSERT_ID() function. It uses this ID to insert corresponding details into the Customer table (name, phone, address). This ensures that each customer has both authentication and personal information properly linked.

  - **BookPackage**: This procedure manages new bookings by inserting a record into the Bookings table. It verifies that at least one identifier (package or hotel) is provided; if neither is given, it signals an error. Bookings are initially recorded with a 'Pending' status, ensuring that incomplete or invalid bookings are not processed further.

  - **CancelBooking**: Before canceling a booking, this procedure checks if the booking status is still 'Pending'. If the booking is not pending, it signals an error. Otherwise, it updates the booking's status to 'Cancelled' and simultaneously updates any associated payment record (if still pending) to 'Failed', ensuring that the booking cancellation is fully reflected in both tables.

  - **MakePayment**: This procedure handles payment processing by updating the payment record with the provided amount and changing its status to 'Completed'. This action, in turn, triggers a confirmation of the booking through an associated trigger, ensuring that the payment update and booking confirmation are synchronized.

- ## Functions

  - **GetCustomerBookingCount:** This function returns the total number of active (non-cancelled) bookings for a specific customer. It uses a COUNT query on the Bookings table, filtering out any bookings that have been cancelled. This provides a quick summary of a customer's current engagements.

  - **GetTotalPayment**: This function calculates the total amount paid for a specific booking. It sums up the amounts from the Payments table and uses COALESCE to return 0 if no payment records exist. This ensures that the function always returns a valid numerical value.

  - **GetAvgHotelRating**: This function computes the average rating for a hotel based on customer reviews. It averages the rating values from the Reviews table, again using COALESCE to return 0 when there are no ratings available. This provides a clear metric of a hotel's performance from the customer's perspective.

- ## Triggers

  - **After_Booking_Insert**: Triggered immediately after a new booking is inserted, this trigger automatically creates an associated payment record. It sets the payment amount to 0 and status to 'Pending', ensuring that every booking has a corresponding payment entry without requiring manual intervention.

  - **After_Payment_Update**: This trigger monitors changes in the Payments table. When it detects that a payment's status has been updated to 'Completed', it automatically updates the corresponding booking's status to 'Confirmed'. This automation streamlines the payment-to-booking confirmation process.

- **Before_Customer_Delete:** To maintain data integrity, this trigger prevents the deletion of a customer record if the customer has any active, confirmed bookings. It checks the number of such bookings and signals an error if one or more exist, safeguarding critical customer data and the relationships within the database.

📸 Output Screenshots: Database Tables Display

This section shows the output of SQL queries run on the database, including data from various tables such as Customer, Bookings, Payments, Packages, Reviews, etc., after performing insertion, booking, payment, and review operations.

message
Customer added successfully

message
Booking placed successfully

Total_Bookings
1

message
Payment successful, booking confirmed

message
Booking cancelled successfully

Total_Payment
45000.00

Avg_Hotel_Rating
4.50

customer_id	account_id	name	phone	address
1	1	Raj Kumar	9876543210	Delhi, India
2	2	Anita Singh	9123456780	Mumbai, India
NULL	NULL	NULL	NULL	NULL

booking_id	customer_id	package_id	hotel_id	booking_date	status
1	1	1	1	2025-04-04 17:00:31	Cancelled
2	1	2	2	2025-04-04 17:00:31	Confirmed
NULL	NULL	NULL	NULL	NULL	NULL

payment_id	booking_id	amount	payment_date	payment_status
1	1	0.00	2025-04-04 17:00:31	Failed
2	2	45000.00	2025-04-04 17:00:31	Completed
NULL	NULL	NULL	NULL	NULL

package_id	package_name	destination	price	duration	details
1	Beach Paradise	Goa	50000.00	5	Includes beach resort stay and water activities
2	Mountain Retreat	Manali	45000.00	4	Includes trekking and adventure activities
NULL	NULL	NULL	NULL	NULL	NULL

review_id	customer_id	package_id	hotel_id	rating	review_text	review_date
1	1	NULL	2	5	Amazing experience!	2025-04-04 17:00:31
2	2	NULL	2	4	Beautiful location and great service.	2025-04-04 17:00:31
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **Contributers :-**

❖ Mayank Patel
❖ Vansh Parmar
❖ Afzal Surti
❖ Purvesh Rohit