CSA0593

**DATABASE MANAGEMENT SYSTEM**

**K.SWETHA**

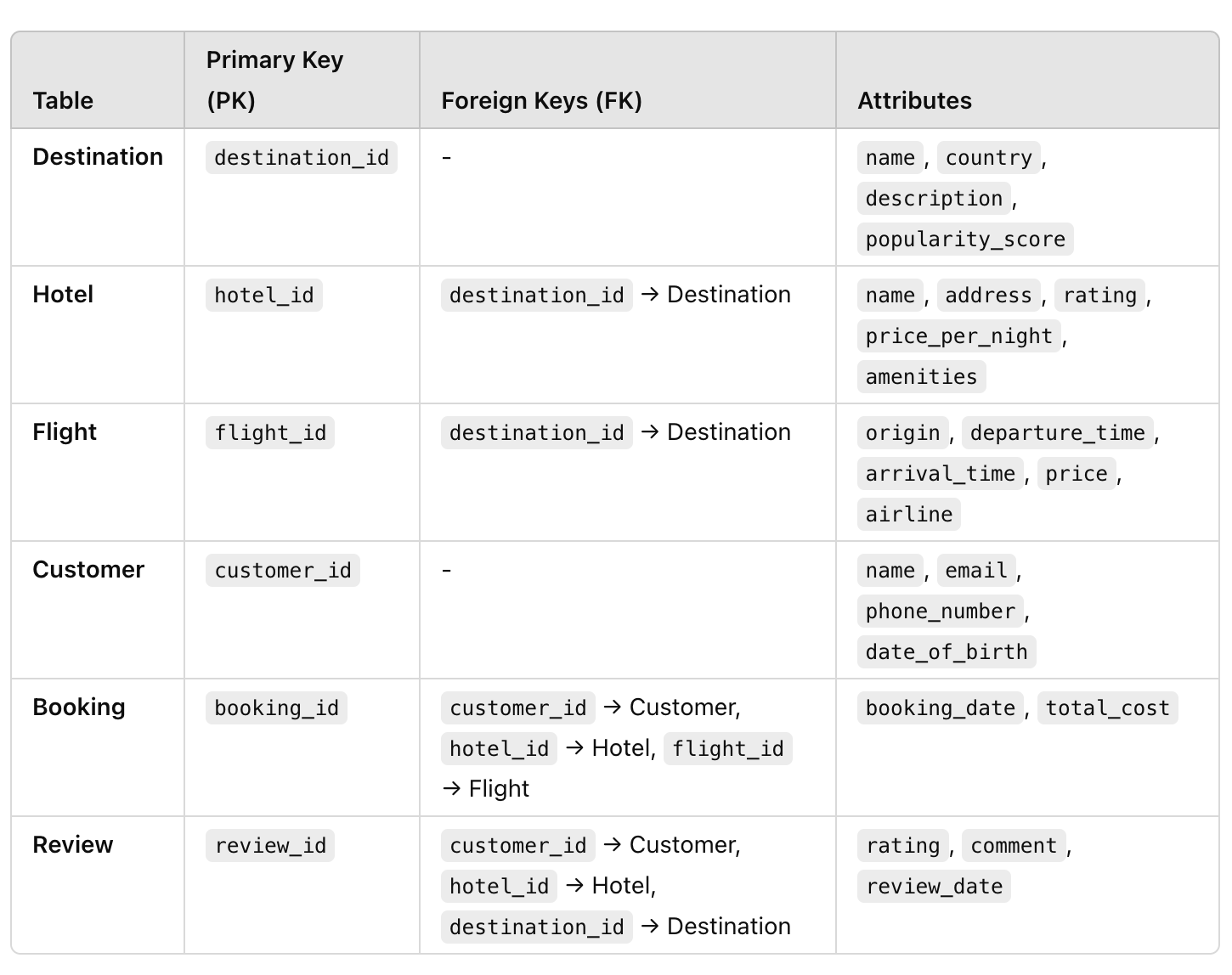
**192311404**

**ASSIGNMENT - 3**

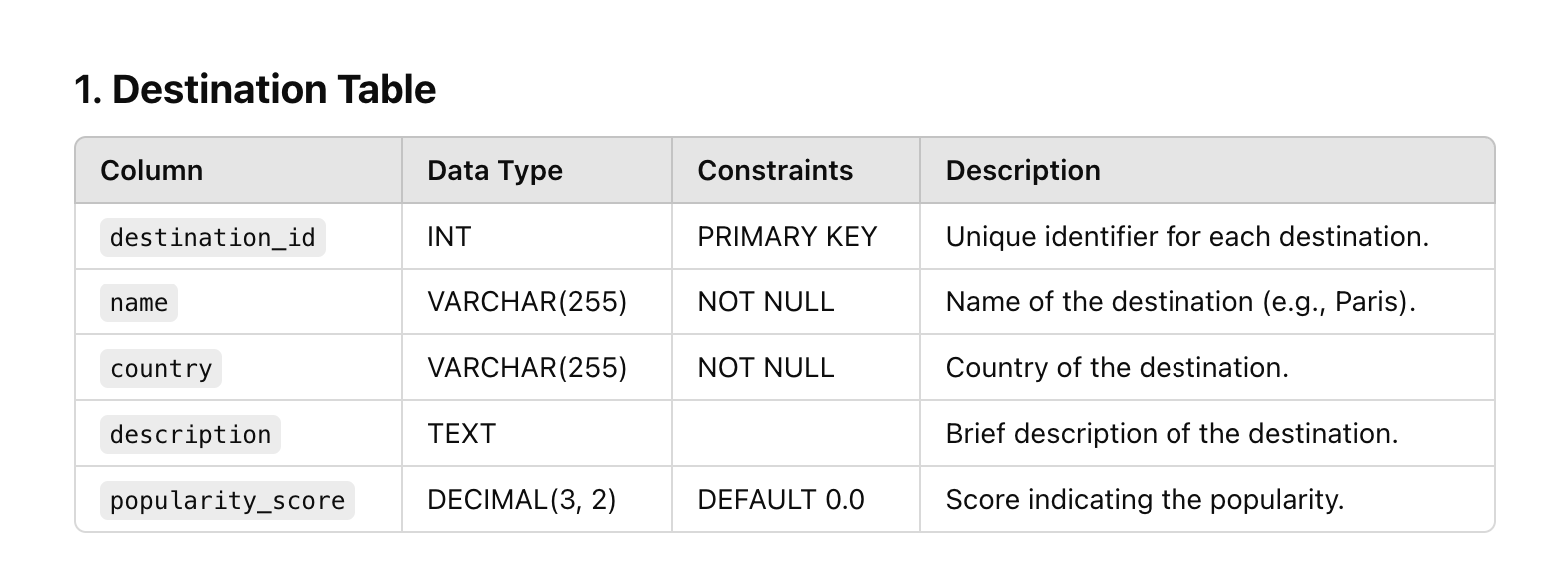
**Develop a database for a travel booking platform with destinations, hotels, flights, and customer reviews.  
  
-Model tables for destinations, hotels, flights, bookings, and customer reviews.  
-Write stored procedures for booking a flight, reserving a hotel room, and submitting a review.  
-Implement triggers to update destination popularity and flight availability after a booking.  
-Write SQL queries to find popular destinations, most-booked hotels, and highest-rated flights.**

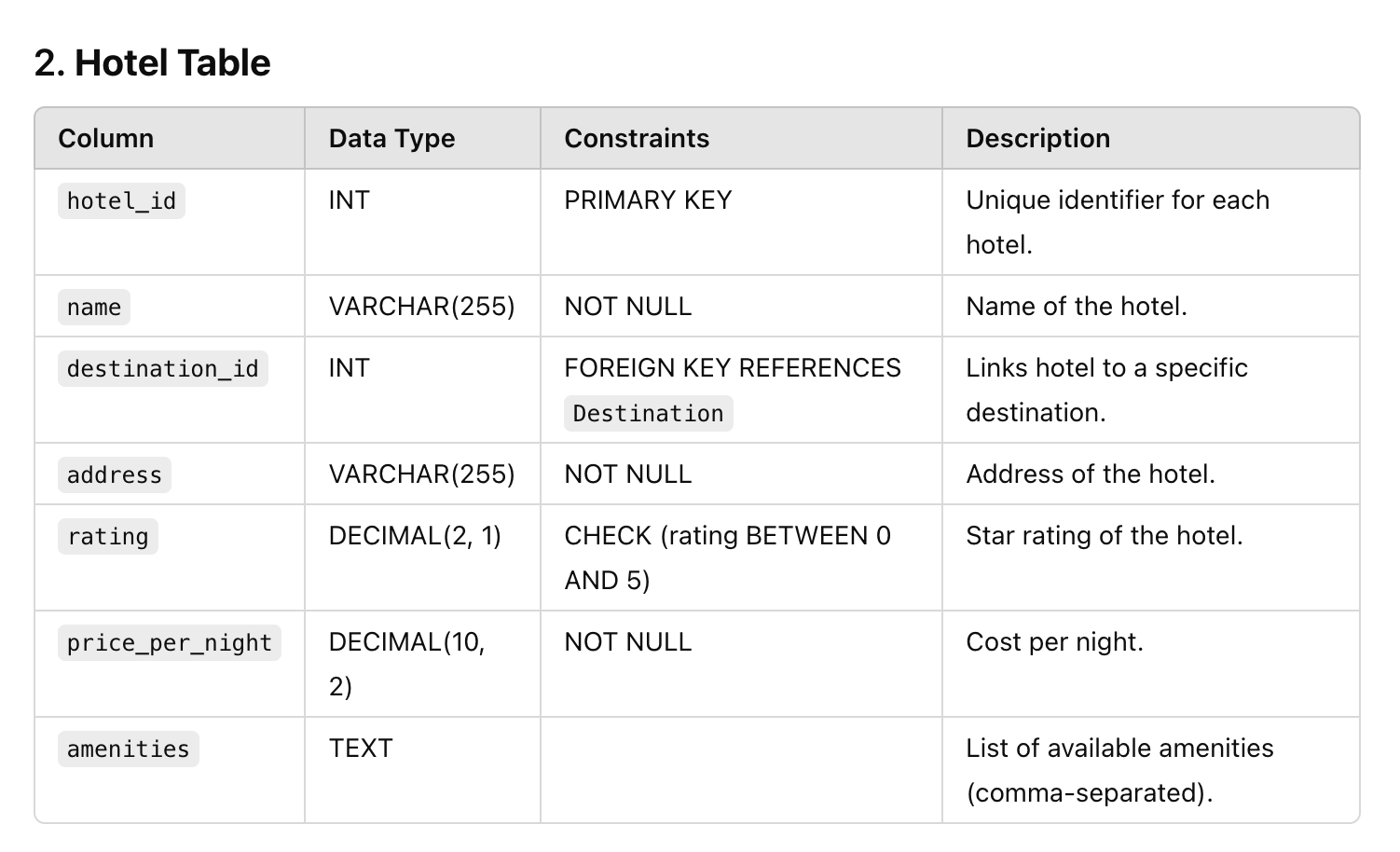
**DATABASE DESIGN :**

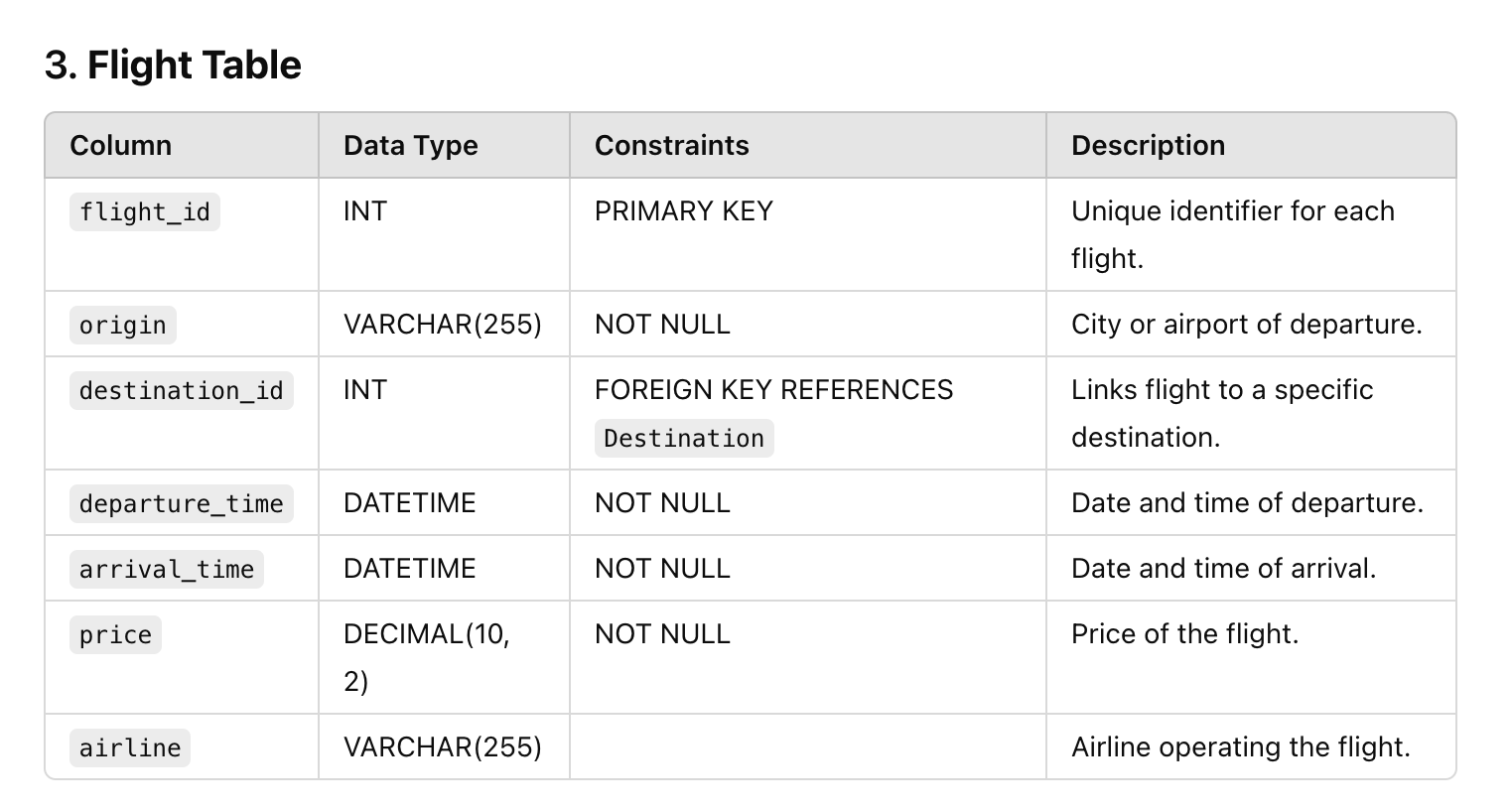
To design a database for a travel booking platform with destinations, hotels, flights, and customer reviews, we’ll focus on the primary entities and relationships that can support the platform’s key features.

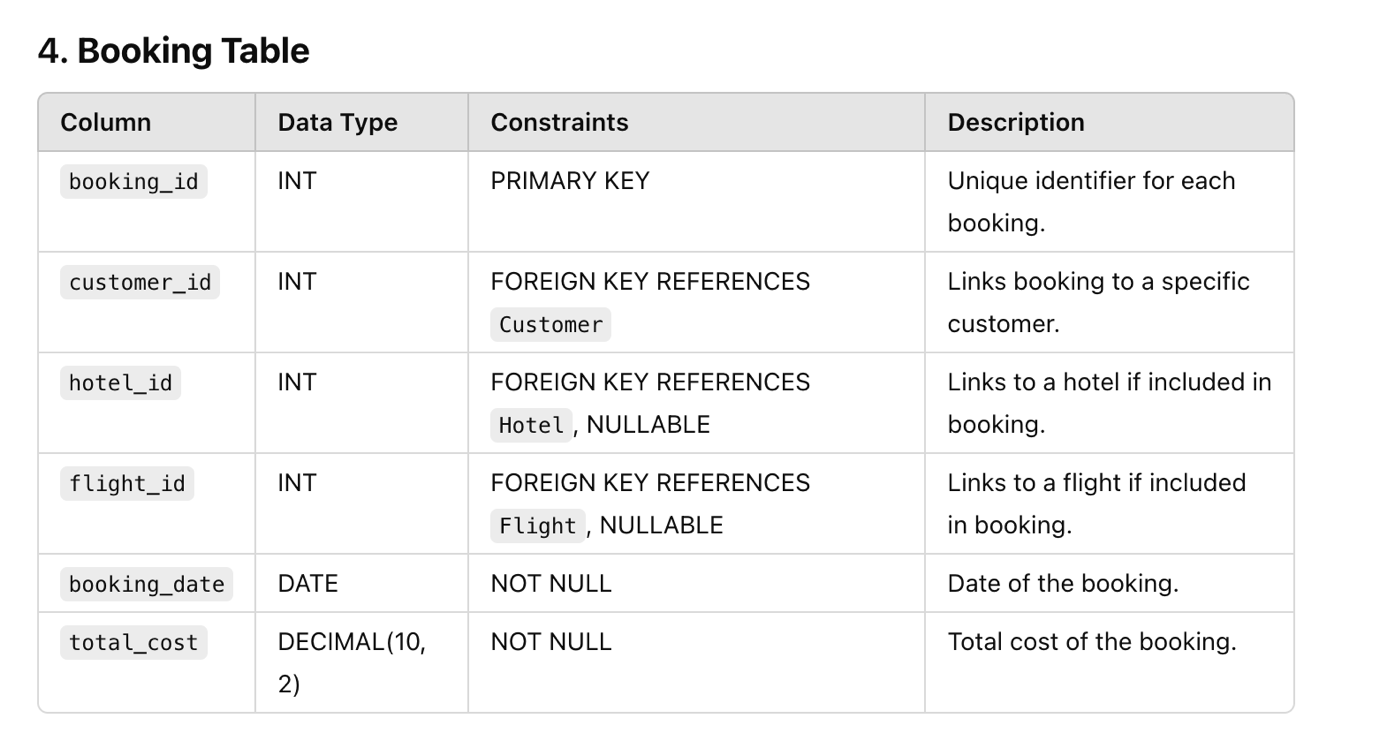
****

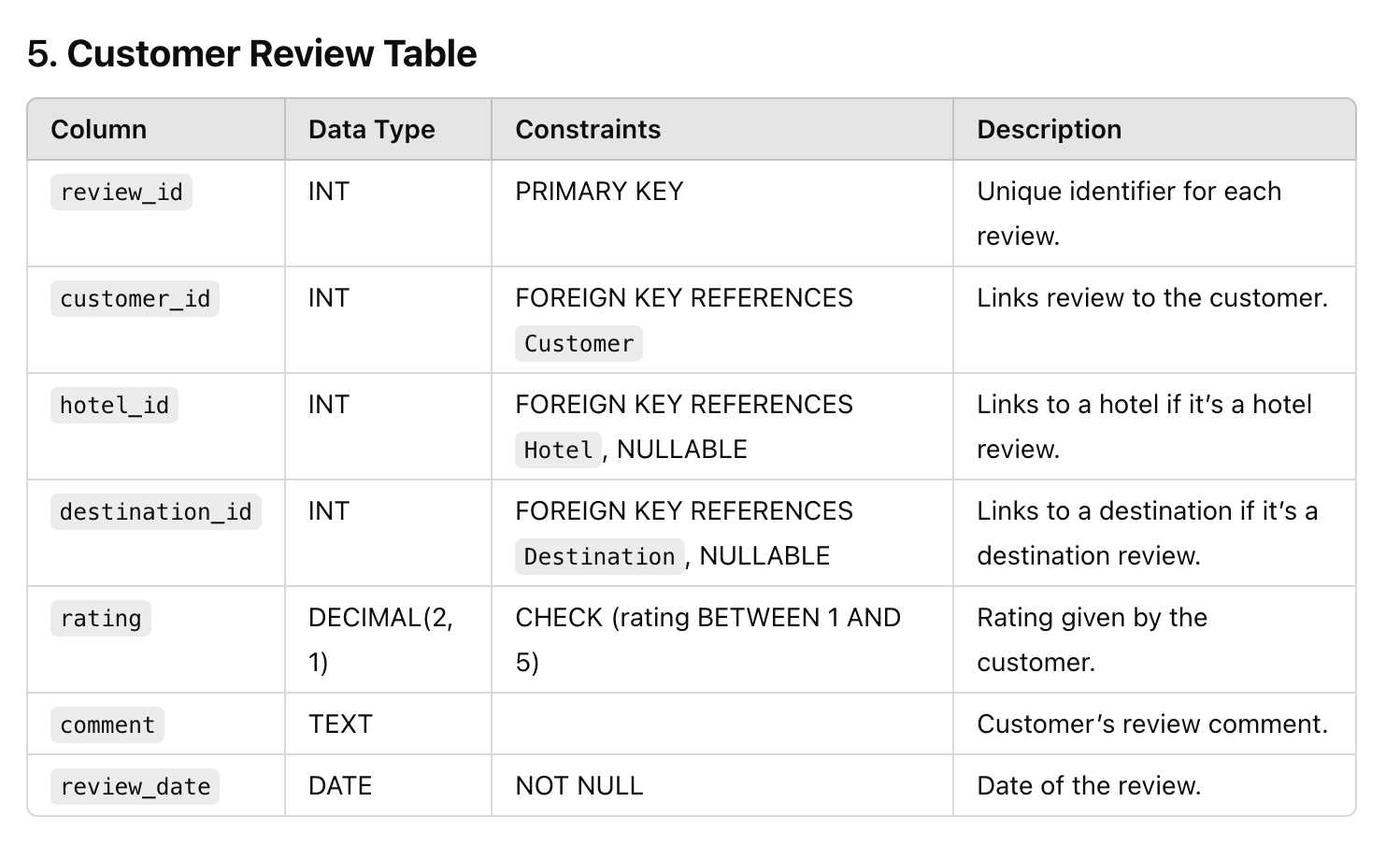
1. **Model tables for destinations, hotels, flights, bookings, and customer reviews:**



****







1. **Write stored procedures for booking a flight, reserving a hotel room, and submitting a review:**

**1. Stored Procedure to Book a Flight**

This procedure inserts a new flight booking for a specific customer, ensuring the flight exists and calculating the total cost for the booking.

DELIMITER //

CREATE PROCEDURE BookFlight (

IN p\_customer\_id INT,

IN p\_flight\_id INT,

OUT p\_booking\_id INT,

OUT p\_total\_cost DECIMAL(10, 2)

)

BEGIN

DECLARE flight\_price DECIMAL(10, 2);

SELECT price INTO flight\_price

FROM Flight

WHERE flight\_id = p\_flight\_id;

IF flight\_price IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Flight not found';

ELSE

INSERT INTO Booking (customer\_id, flight\_id, booking\_date, total\_cost)

VALUES (p\_customer\_id, p\_flight\_id, CURDATE(), flight\_price);

SET p\_booking\_id = LAST\_INSERT\_ID();

SET p\_total\_cost = flight\_price;

END IF;

END //

DELIMITER ;

### 2. ****Stored Procedure to Reserve a Hotel Room****

This procedure reserves a hotel room for a specific customer by creating a new hotel booking entry. It checks if the hotel exists and calculates the total cost based on the number of nights.

DELIMITER //

CREATE PROCEDURE ReserveHotel (

IN p\_customer\_id INT,

IN p\_hotel\_id INT,

IN p\_nights INT,

OUT p\_booking\_id INT,

OUT p\_total\_cost DECIMAL(10, 2)

)

BEGIN

DECLARE hotel\_price DECIMAL(10, 2);

SELECT price\_per\_night INTO hotel\_price

FROM Hotel

WHERE hotel\_id = p\_hotel\_id;

IF hotel\_price IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Hotel not found';

ELSE

SET p\_total\_cost = hotel\_price \* p\_nights;

INSERT INTO Booking (customer\_id, hotel\_id, booking\_date, total\_cost)

VALUES (p\_customer\_id, p\_hotel\_id, CURDATE(), p\_total\_cost);

SET p\_booking\_id = LAST\_INSERT\_ID();

END IF;

END //

DELIMITER ;

### 3. ****Stored Procedure to Submit a Review****

This procedure allows a customer to submit a review for a hotel or destination. It checks for valid references and inserts the review, allowing either a hotel or destination review but not both in the same review entry.

DELIMITER //

CREATE PROCEDURE SubmitReview (

IN p\_customer\_id INT,

IN p\_hotel\_id INT,

IN p\_destination\_id INT,

IN p\_rating DECIMAL(2, 1),

IN p\_comment TEXT,

OUT p\_review\_id INT

)

BEGIN

IF p\_hotel\_id IS NOT NULL AND p\_destination\_id IS NOT NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Please provide either a hotel\_id or destination\_id, not both';

END IF;

IF NOT EXISTS (SELECT 1 FROM Hotel WHERE hotel\_id = p\_hotel\_id) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Hotel not found';

END IF;

END IF;

IF p\_destination\_id IS NOT NULL THEN

IF NOT EXISTS (SELECT 1 FROM Destination WHERE destination\_id = p\_destination\_id) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Destination not found';

END IF;

END IF;

INSERT INTO Review (customer\_id, hotel\_id, destination\_id, rating, comment, review\_date)

VALUES (p\_customer\_id, p\_hotel\_id, p\_destination\_id, p\_rating, p\_comment, CURDATE());

SET p\_review\_id = LAST\_INSERT\_ID();

END //

DELIMITER ;

**C) Implement triggers to update destination popularity and flight availability after a booking:**

### 1. ****Trigger to Update Destination Popularity Score After a Review****

This trigger updates the popularity\_score in the Destination table based on the average rating of all reviews for that destination each time a new review is added or an existing review is updated.

DELIMITER //

CREATE TRIGGER UpdateDestinationPopularity

AFTER INSERT ON Review

FOR EACH ROW

BEGIN

DECLARE avg\_rating DECIMAL(3, 2);

IF NEW.destination\_id IS NOT NULL THEN

SELECT AVG(rating) INTO avg\_rating

FROM Review

WHERE destination\_id = NEW.destination\_id;

UPDATE Destination

SET popularity\_score = avg\_rating

WHERE destination\_id = NEW.destination\_id;

END IF;

END //

DELIMITER ;

### 2. ****Trigger to Update Flight Availability After a Booking****

Assuming there’s a seats\_available column in the Flight table, this trigger will decrease the seat count whenever a new booking is made for a flight.

DELIMITER //

CREATE TRIGGER UpdateFlightAvailability

AFTER INSERT ON Booking

FOR EACH ROW

BEGIN

IF NEW.flight\_id IS NOT NULL THEN

UPDATE Flight

SET seats\_available = seats\_available - 1

WHERE flight\_id = NEW.flight\_id AND seats\_available > 0;

END IF;

END //

DELIMITER ;

1. **Write SQL queries to find popular destinations, most-booked hotels, and highest-rated flights:**

### 1. ****Query to Find Popular Destinations****

This query retrieves destinations ordered by their popularity score (average rating), from highest to lowest. It assumes that the Destination table has a popularity\_score attribute that reflects the average rating from customer reviews.

SELECT

destination\_id,

name,

country,

popularity\_score

FROM

Destination

ORDER BY

popularity\_score DESC

LIMIT 10;

### 2. ****Query to Find Most-Booked Hotels****

This query counts the number of bookings for each hotel and orders the hotels by the number of bookings in descending order, showing the most-booked hotels first.

SELECT

h.hotel\_id,

h.name AS hotel\_name,

d.name AS destination\_name,

COUNT(b.booking\_id) AS booking\_count

FROM

Hotel h

JOIN

Booking b ON h.hotel\_id = b.hotel\_id

JOIN

Destination d ON h.destination\_id = d.destination\_id

GROUP BY

h.hotel\_id, h.name, d.name

ORDER BY

booking\_count DESC

LIMIT 10;

### 3. ****Query to Find Highest-Rated Flights****

This query finds flights with the highest average rating from customer reviews, ordered from highest to lowest rating. It assumes the Review table has a flight\_id column for reviews associated with flights

SELECT

f.flight\_id,

f.origin,

d.name AS destination\_name,

f.airline,

AVG(r.rating) AS average\_rating

FROM

Flight f

JOIN

Review r ON f.flight\_id = r.flight\_id

JOIN

Destination d ON f.destination\_id = d.destination\_id

GROUP BY

f.flight\_id, f.origin, d.name, f.airline

ORDER BY

average\_rating DESC

LIMIT 10;

**CONCLUSION:**

In conclusion, the SQL queries and database features enable the travel booking platform to provide valuable insights into popular destinations, most-booked hotels, and highest-rated flights. By leveraging real-time customer reviews and booking data, the platform can recommend trending locations, popular accommodations, and highly-rated flights, enhancing user experience. These automated processes, supported by triggers and stored procedures, ensure up-to-date information, streamline operations, and improve customer satisfaction, contributing to the platform’s growth and competitiveness.