

LLM Response

Copilot Link - [Link](#)

1. Database Schema

Here is the database schema for my DBMS Project -

```
DROP DATABASE IF EXISTS TravelEase;
CREATE DATABASE TravelEase;
USE TravelEase;

CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,

    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50),

    email VARCHAR(255) UNIQUE,
    mobile_number VARCHAR(20) UNIQUE,

    password VARCHAR(25) NOT NULL,

    CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)
);

CREATE TABLE Admin (
    admin_id INT AUTO_INCREMENT PRIMARY KEY,

    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50),

    email VARCHAR(255) UNIQUE,
    mobile_number VARCHAR(20) UNIQUE,

    password VARCHAR(25) NOT NULL,

    CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)
);

CREATE TABLE TourismAgency (
```

```

agency_id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,
location VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE,
mobile_number VARCHAR(20) UNIQUE,

password VARCHAR(25) NOT NULL,

CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)
);

CREATE TABLE Itinerary (
    itinerary_id INT AUTO_INCREMENT PRIMARY KEY,
    agency_id INT NOT NULL,

    description TEXT NOT NULL,

    duration_day INT NOT NULL,
    duration_night INT NOT NULL,

    price DECIMAL(10,2) NOT NULL,

    destination_city VARCHAR(50) NOT NULL,
    destination_state VARCHAR(50) NOT NULL,
    destination_country VARCHAR(50) NOT NULL,

    FOREIGN KEY (agency_id) REFERENCES TourismAgency(agency_id)
);

CREATE TABLE TransportProvider (
    provider_id INT AUTO_INCREMENT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    email VARCHAR(255) UNIQUE,
    mobile_number VARCHAR(20) UNIQUE,

    service_type ENUM('Train', 'Airplane') NOT NULL,

    password VARCHAR(25) NOT NULL,

    CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)
);

```

```
CREATE TABLE Payment (  
    payment_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    amount DECIMAL(10,2) NOT NULL,  
  
    payment_method ENUM('Credit Card', 'Debit Card', 'Net Banking', 'UPI') NOT  
NULL,  
    payment_status ENUM('Confirmed', 'Pending', 'Cancelled') NOT NULL  
);
```

```
CREATE TABLE Coupon (  
    coupon_code VARCHAR(10) PRIMARY KEY,  
    discount_percentage INT NOT NULL,  
    expiry_date DATETIME  
);
```

```
CREATE TABLE Booking (  
    booking_id INT AUTO_INCREMENT PRIMARY KEY,  
    customer_id INT NOT NULL,  
    payment_id INT NOT NULL UNIQUE,  
    coupon_code VARCHAR(10),  
  
    transport_type ENUM('Train', 'Airplane'),  
    status ENUM('Confirmed', 'Pending', 'Cancelled') NOT NULL,  
    booking_date DATE NOT NULL,  
  
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),  
    FOREIGN KEY (payment_id) REFERENCES Payment(payment_id),  
    FOREIGN KEY (coupon_code) REFERENCES Coupon(coupon_code)  
);
```

```
CREATE TABLE Airplane (  
    airplane_id VARCHAR(10) PRIMARY KEY,  
  
    name VARCHAR(100) NOT NULL,  
    capacity INT NOT NULL,  
    provider_id INT NOT NULL,  
  
    FOREIGN KEY (provider_id) REFERENCES TransportProvider(provider_id)  
);
```

```
CREATE TABLE AirplaneRoute (  
    route_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    arrival_location VARCHAR(100) NOT NULL,  
    departure_location VARCHAR(100) NOT NULL
```

```
);
```

```
CREATE TABLE Train (  
    train_id VARCHAR(10) PRIMARY KEY,  
  
    name VARCHAR(100) NOT NULL,  
    capacity INT NOT NULL,  
    provider_id INT NOT NULL,  
  
    FOREIGN KEY (provider_id) REFERENCES TransportProvider(provider_id)  
);
```

```
CREATE TABLE TrainRoute (  
    route_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    arrival_location VARCHAR(100) NOT NULL,  
    departure_location VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE AccommodationProvider (  
    provider_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    name VARCHAR(100) NOT NULL,  
  
    email VARCHAR(255) UNIQUE,  
    mobile_number VARCHAR(20) UNIQUE,  
  
    password VARCHAR(25) NOT NULL,  
  
    CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)  
);
```

```
CREATE TABLE Hotel (  
    hotel_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    name VARCHAR(100) NOT NULL,  
    location VARCHAR(100) NOT NULL,  
    price_per_night DECIMAL(10,2) NOT NULL,  
  
    email VARCHAR(255) UNIQUE,  
    mobile_number VARCHAR(20) UNIQUE,  
  
    total_rooms INT NOT NULL,  
  
    hotel_description TEXT NOT NULL,  
    provider_id INT NOT NULL,
```

```
FOREIGN KEY (provider_id) REFERENCES AccommodationProvider(provider_id),  
CHECK (email IS NOT NULL OR mobile_number IS NOT NULL)  
);
```

```
CREATE TABLE Reviews (  
    review_id INT AUTO_INCREMENT PRIMARY KEY,  
  
    comment TEXT,  
    rating INT NOT NULL,  
    booking_id INT NOT NULL,  
  
    item_type ENUM('Hotel', 'Itinerary') NOT NULL,  
    item_id INT NOT NULL,  
  
    CHECK (rating BETWEEN 1 AND 5),  
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id)  
);
```

```
CREATE TABLE A_Route_Follows (  
    arf_pkey INT AUTO_INCREMENT PRIMARY KEY,  
    route_id INT NOT NULL UNIQUE,  
  
    price DECIMAL(10,2) NOT NULL,  
    available_seats INT NOT NULL,  
  
    arrival_time TIME NOT NULL,  
    arrival_date DATE NOT NULL,  
  
    departure_time TIME NOT NULL,  
    departure_date DATE NOT NULL,  
  
    airplane_id VARCHAR(10) NOT NULL,  
  
    FOREIGN KEY (airplane_id) REFERENCES Airplane(airplane_id),  
    FOREIGN KEY (route_id) REFERENCES AirplaneRoute(route_id)  
);
```

```
CREATE TABLE T_Route_Follows (  
    trf_pkey INT AUTO_INCREMENT PRIMARY KEY,  
    route_id INT NOT NULL UNIQUE,  
  
    price DECIMAL(10,2) NOT NULL,  
    available_seats INT NOT NULL,  
  
    arrival_time TIME NOT NULL,
```

```

arrival_date DATE NOT NULL,

departure_time TIME NOT NULL,
departure_date DATE NOT NULL,

train_id VARCHAR(10) NOT NULL,

FOREIGN KEY (train_id) REFERENCES Train(train_id),
FOREIGN KEY (route_id) REFERENCES TrainRoute(route_id)
);

CREATE TABLE A_Book_Includes (
    booking_id INT NOT NULL,
    arf_pkey INT NOT NULL,
    tickets_booked INT NOT NULL,

    PRIMARY KEY (booking_id, arf_pkey),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id),
    FOREIGN KEY (arf_pkey) REFERENCES A_Route_Follows(arf_pkey)
);

CREATE TABLE T_Book_Includes (
    booking_id INT NOT NULL,
    trf_pkey INT NOT NULL,
    tickets_booked INT NOT NULL,

    PRIMARY KEY (booking_id, trf_pkey),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id),
    FOREIGN KEY (trf_pkey) REFERENCES T_Route_Follows(trf_pkey)
);

CREATE TABLE H_Book_Includes (
    booking_id INT NOT NULL,
    hotel_id INT NOT NULL,

    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,

    room_booked INT NOT NULL,

    PRIMARY KEY (booking_id, hotel_id),
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id),
    FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);

```

```
CREATE TABLE I_Book_Includes (  
    booking_id INT NOT NULL,  
    itinerary_id INT NOT NULL,  
    itinerary_start_date DATE NOT NULL,  
  
    PRIMARY KEY (booking_id, itinerary_id),  
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id),  
    FOREIGN KEY (itinerary_id) REFERENCES Itinerary(itinerary_id)  
);
```

Response

Your database schema for "TravelEase" is well-structured and provides a comprehensive system to manage travel-related operations. Here's a quick overview:

Strengths of the Schema:

1. Comprehensive Coverage:

- Manages customers, admins, tourism agencies, transport providers, and accommodation providers.
- Includes detailed entities for itineraries, bookings, hotels, routes, and reviews.

2. Data Integrity:

- Use of `CHECK`, `NOT NULL`, and `UNIQUE` constraints ensures the validity of data.
- Clear relationships established through `FOREIGN KEY` constraints.

3. Normalization:

- Reduces redundancy by using junction tables for many-to-many relationships (e.g., bookings with itineraries, hotels, or transportation).

4. Flexibility for Growth:

- Enums (e.g., `payment_method`, `status`, etc.) allow for easy adjustments without overcomplicating the schema.

Potential Enhancements:

1. Address Fields:

- Adding fields for address in tables like `Customer`, `TourismAgency`, and `AccommodationProvider` could provide more detailed information.

2. Time Zones:

- Including time zones for arrival and departure times in the route tables would be crucial for global travel services.

3. Cancellation Policies:

- Tables like `Booking` and `Itinerary` could include fields for cancellation and refund policies.

4. Images:

- Adding an image field (or a related table) for itineraries, hotels, and agencies could make the system more visually descriptive.

5. Advanced Room/Seat Selection:

- Fields or mechanisms to enable specific room or seat selection during booking would enhance the user experience.

Your schema sets a strong foundation for building a full-fledged travel management system. If you'd like, I can assist in crafting queries or exploring ways to implement some of these enhancements!

2. Query - 1

Write a SQL query to number of Rooms Available from start date, end date on a particular hotel id start date = '2025-04-29' end date = '2025-05-02' hotel_id = 1

Response

Here's the SQL query to calculate the number of rooms available at a specific hotel for the given date range:

```
SELECT
    h.total_rooms - COALESCE(SUM(hbi.room_booked), 0) AS available_rooms
FROM
    Hotel h
LEFT JOIN
    H_Book_Includes hbi ON h.hotel_id = hbi.hotel_id
WHERE
    h.hotel_id = 1
    AND (hbi.hotel_id IS NULL -- Include hotels with no bookings
        OR NOT (
            hbi.check_in_date < '2025-05-02'
            AND hbi.check_out_date > '2025-04-29'
        ));
```

Explanation:

1. Select Clause:

- `h.total_rooms` : Fetches the total number of rooms in the hotel.

- `COALESCE(SUM(hbi.room_booked), 0)` : Calculates the total number of rooms booked for overlapping dates. If there are no bookings, it returns `0`.

2. FROM and LEFT JOIN:

- Starts with the `Hotel` table (`h`) and joins it with the `H_Book_Includes` table (`hbi`) to get booking data.
- `LEFT JOIN` ensures hotels with no bookings are still included in the results.

3. WHERE Clause:

- `h.hotel_id = 1` : Filters for the specific hotel by its ID.
- `(hbi.hotel_id IS NULL OR NOT (hbi.check_in_date < '2025-05-02' AND hbi.check_out_date > '2025-04-29'))` :
 - The first part (`hbi.hotel_id IS NULL`) includes hotels with no bookings.
 - The second part (`NOT (hbi.check_in_date < 'end_date' AND hbi.check_out_date > 'start_date')`) excludes bookings that overlap with the given date range.

This query calculates the difference between total rooms and the number of rooms booked during the specified date range. If there's anything else you'd like to tweak or optimize, let me know!

Why is this query is wrong?

1. Strict inequality is used for the check-in and check-out dates. This will exclude other bookings made by other users for which either or both the check-in and check-out dates are exactly equal to the current user's check-in and check-out dates. Weak inequalities should be used instead.
2. The only case for which the WHERE clause is true is when another user's hotel booking totally overlaps inside the current user's booking dates. And therefore, the number of rooms booked corresponding to only this case are subtracted from the total number of rooms. Cases in which another user's booking partially but not totally overlaps with the current user's booking are not subtracted from the total number of rooms. This is incorrect as both total and partial bookings reduce the number of rooms invalidated. For example, consider another user's booking from 2025-04-28 to 2025-05-01. `NOT (hbi.check_in_date < '2025-05-02' AND hbi.check_out_date > '2025-04-29')` evaluates to false and therefore this booking with partial overlap is excluded and not counted as an occupied room.

The correct query

```
SELECT h.total_rooms - COALESCE(SUM(hbi.room_booked), 0) AS available_rooms
FROM Hotel h LEFT JOIN H_Book_Includes hbi ON h.hotel_id = hbi.hotel_id
```

```
WHERE h.hotel_id = 1 AND hbi.check_in_date <= '2025-05-02' AND  
hbi.check_out_date >= '2025-04-29';
```

- This query uses weak inequalities and make sures that both partial and total overlaps are handled properly.

3. Query - 2

Write a query to Find all airplane available and travel time (in min) Given :
arrival location = 'New Delhi' , departure location = 'Mumbai', departure date =
'2025-03-01'

Response

Here's a SQL query to find all available airplanes along with their travel time in minutes for the given criteria:

```
SELECT  
    a.airplane_id,  
    a.name AS airplane_name,  
    arf.price,  
    arf.available_seats,  
    arf.arrival_time,  
    arf.departure_time,  
    TIMESTAMPDIFF(MINUTE, arf.departure_time, arf.arrival_time) AS  
travel_time_minutes  
FROM  
    Airplane a  
JOIN  
    A_Route_Follows arf ON a.airplane_id = arf.airplane_id  
JOIN  
    AirplaneRoute ar ON arf.route_id = ar.route_id  
WHERE  
    ar.arrival_location = 'New Delhi'  
    AND ar.departure_location = 'Mumbai'  
    AND arf.departure_date = '2025-03-01'  
    AND arf.available_seats > 0;
```

Explanation:

1. Selected Fields:

- a.airplane_id : Fetches the unique airplane ID.
- a.name AS airplane_name : The name of the airplane, with an alias for clarity.

- `arf.price` : The price of the flight.
- `arf.available_seats` : The number of available seats.
- `arf.arrival_time` and `arf.departure_time` : For schedule details.
- `TIMESTAMPDIFF(MINUTE, arf.departure_time, arf.arrival_time)` :
 - This calculates the travel time in minutes using the departure and arrival times.

2. FROM and JOINS:

- The `Airplane`, `A_Route_Follows`, and `AirplaneRoute` tables are joined to retrieve all relevant data.
- `ON` clauses ensure that the data from these tables is properly connected via foreign keys.

3. WHERE Clause:

- Filters the results based on:
 - `ar.arrival_location = 'New Delhi'` and `ar.departure_location = 'Mumbai'` for the desired route.
 - `arf.departure_date = '2025-03-01'` for the specific departure date.
 - `arf.available_seats > 0` to include only flights with available seats.

This query will return all airplanes meeting your criteria, along with their travel time in minutes. If you'd like adjustments or need further assistance, feel free to ask!

Why is this query wrong?

- The duration of the flight in minutes is not correctly computed. Only the `arrival_time` and `departure_time` attributes are passed to `TIMESTAMPDIFF()`. This assumes that arrival and departure happens on the same date. It does not account for different arrival and departure dates.

The correct query

```
SELECT *, TIMESTAMPDIFF ( minute, CONCAT(departure_date, " ",
departure_time), CONCAT(arrival_date, " ", arrival_time) ) AS 'travel_time
(min)'
FROM A_Route_Follows NATURAL JOIN AirplaneRoute NATURAL JOIN Airplane
WHERE departure_date = '2025-03-01' AND departure_location = 'Mumbai' AND
arrival_location = 'New Delhi';
```

- We create DATETIME objects by concatenating the dates and times of arrival and departure, which are then passed to TIMESTAMPDIFF. This ensures that arrivals and departures on different dates are correctly handled.