2nd Year 2nd Semester

University of Dhaka

Department of Computer Science and Engineering

Session: 2022-2023

CSE2211: Database Management Systems-1

Lab Project

Submitted to:

Mr. Abu Ahmed Ferdaus, Dept. of CSE, University of Dhaka

Mr. Redwan Ahmed Rizvee, Dept. of CSE, University of Dhaka

Submission Date:

17/07/2025

Submitted By:

Ahnaf Mahbub Khan (38)

Mahbmudul Hassan Arafat (60)

Contents

1 Project Objectives

- To understand and apply the principles of database design by developing a comprehensive relational database for an airline management system.
- To design and implement an efficient and normalized database schema that accurately models airline operations such as flights, bookings, passengers, and payments.
- To learn how to use SQL for creating, managing, and querying the airline database to facilitate smooth data management and retrieval.
- To develop a user-friendly interface for managing airline schedules, passenger reservations, and payment processing.
- To explore the concepts of entity-relationship modeling, including identification of weak entities, derived attributes, and relationships in the context of airline data.
- To understand transaction management and ensure data integrity during concurrent bookings and payments.
- To gain practical experience in integrating backend database systems with front-end applications.

2 Project Scope

- The system will manage flight scheduling, passenger information, seat reservations, and payment transactions.
- The project will focus on database design, including tables for flights, passengers, bookings, payments, and related constraints.
- The system will handle basic reporting features such as passenger lists for flights and booking summaries.

3 Project Description

This database system is designed for an airline booking and management system. It comprises several tables to store information about users, airports, airplanes, flights, bookings, passengers, payments, and reviews.

User and Administrative Data
This section handles all information related to individuals using the system, including their personal details and assigned roles. It differentiates between customers making bookings and administrators managing the system, ensuring appropriate access levels.

Airport and Flight Operations
This core component manages comprehensive data on airports, including their location and identification. It also meticulously tracks details about airplanes and individual flights, encompassing their routes, schedules, and pricing, with a built-in check to prevent flights to the same origin.

Booking and Passenger Management
This part of the system facilitates the entire booking process, associating users with specific flights and recording booking timestamps and statuses. It further manages detailed information for each passenger within a booking, including their assigned seating and personal demographics.

Financial and Feedback Mechanisms
This section is dedicated to handling monetary transactions, specifically tracking payment details for each booking, including the amount and method. Additionally, it incorporates a valuable feedback loop through user reviews, allowing for ratings and comments on flights to gauge satisfaction and improve services.

4 Database Diagram

4.1 Relational Schema

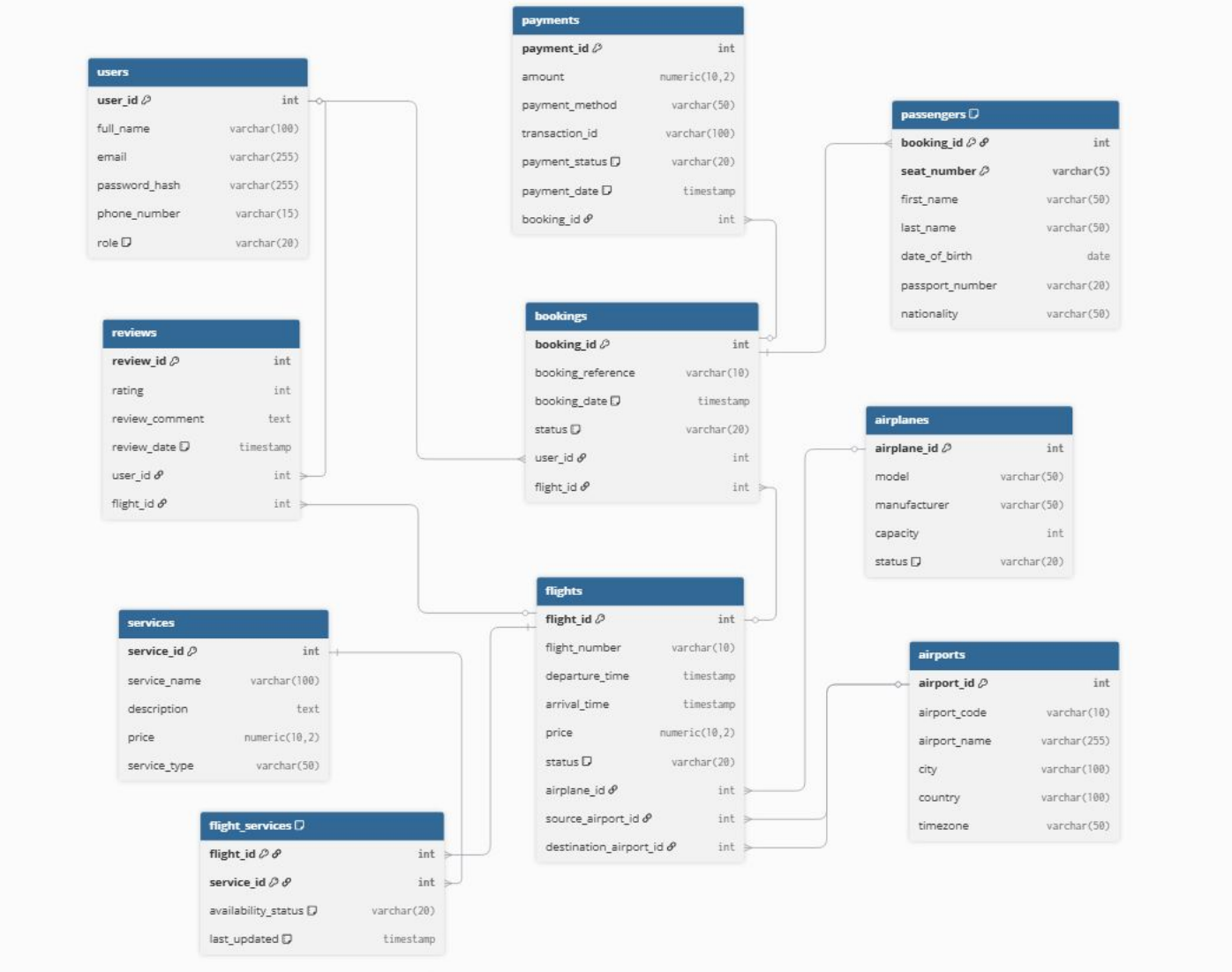

Figure 1: Relational Schema

4.2 Entity-Relationship Diagram

5 Schema

5.1 Users

This table stores all information about registered users including their personal details and system roles.

Attribute	Description	Data Type	Constraints
user_id	User ID	INTEGER	PRIMARY KEY
full_name	Full name	VARCHAR(100)	NOT NULL
email	Email address	VARCHAR(255)	UNIQUE, NOT NULL
password_hash	Password hash	VARCHAR(255)	NOT NULL, LENGTH ≥ 8
phone_number	Phone number	VARCHAR(15)	NOT NULL
role	User role	VARCHAR(20)	DEFAULT 'Customer', CHECK ('Customer','Admin')

5.2 Airports

This table contains information about airports including location and timezone details.

Attribute	Description	Data Type	Constraints
airport_id	Airport ID	INTEGER	PRIMARY KEY
airport_code	IATA/ICAO code	VARCHAR(10)	UNIQUE, NOT NULL
airport_name	Airport name	VARCHAR(255)	NOT NULL
city	City location	VARCHAR(100)	NOT NULL
country	Country location	VARCHAR(100)	NOT NULL
timezone	Timezone	VARCHAR(50)	NOT NULL

5.3 Airplanes

This table lists all airplanes in the fleet with their specifications and current status.

Attribute	Description	Data Type	Constraints
airplane_id	Airplane ID	INTEGER	PRIMARY KEY
model	Aircraft model	VARCHAR(50)	NOT NULL
manufacturer	Manufacturer name	VARCHAR(50)	NOT NULL
capacity	Seating capacity	INTEGER	NOT NULL, CHECK > 0
status	Current status	VARCHAR(20)	DEFAULT 'Active', CHECK ('Active', 'Maintenance', 'Retired')

5.4 Services

This table defines all services available on flights with pricing information.

Attribute	Description	Data Type	Constraints
service_id	Service ID	INTEGER	PRIMARY KEY
service_name	Service name	VARCHAR(100)	NOT NULL
description	Service description	TEXT	
price	Service price	NUMERIC(10,2)	NOT NULL, CHECK ≥ 0
service_type	Service category	VARCHAR(50)	NOT NULL, CHECK ('Food', 'Entertainment', 'Comfort', 'WiFi', 'Amenity')

5.5 Flights

This table records all scheduled flights with timing, pricing, and aircraft assignments.

Attribute	Description	Data Type	Constraints
flight_id	Flight ID	INTEGER	PRIMARY KEY
flight_number	Flight number	VARCHAR(10)	UNIQUE, NOT NULL
departure_time	Departure time	TIMESTAMP	NOT NULL
arrival_time	Arrival time	TIMESTAMP	NOT NULL, CHECK (departure $<$ arrival)
price	Ticket price	NUMERIC(10,2)	NOT NULL, CHECK > 0
status	Flight status	VARCHAR(20)	DEFAULT 'Scheduled', CHECK ('Scheduled', 'Delayed', 'Cancelled', 'Completed')
airplane_id	Assigned airplane	INTEGER	NOT NULL, FK $\rightarrow$ airplanes
source_airport_id	Source airport	INTEGER	NOT NULL, FK $\rightarrow$ airports
destination_airport_id	Destination airport	INTEGER	NOT NULL, FK $\rightarrow$ airports, CHECK ($\neq$ source)

5.6 Bookings

This table stores flight bookings made by users with booking details and status.

Attribute	Description	Data Type	Constraints
booking_id	Booking ID	INTEGER	PRIMARY KEY
booking_reference	Reference code	VARCHAR(10)	UNIQUE, NOT NULL
booking_date	Booking date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
status	Booking status	VARCHAR(20)	DEFAULT 'Confirmed', CHECK ('Confirmed', 'Cancelled', 'Pending')
user_id	User who booked	INTEGER	NOT NULL, FK $\rightarrow$ users
flight_id	Booked flight	INTEGER	NOT NULL, FK $\rightarrow$ flights

5.7 Payments

This table records payment transactions for bookings with payment methods and status.

Attribute	Description	Data Type	Constraints
payment_id	Payment ID	INTEGER	PRIMARY KEY
amount	Amount paid	NUMERIC(10,2)	NOT NULL, CHECK > 0
payment_method	Payment method	VARCHAR(50)	NOT NULL, CHECK ('Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer')
transaction_id	Transaction ID	VARCHAR(100)	UNIQUE
payment_status	Payment status	VARCHAR(20)	DEFAULT 'Pending', CHECK ('Pending', 'Completed', 'Failed', 'Refunded')
payment_date	Payment date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
booking_id	Associated booking	INTEGER	UNIQUE, NOT NULL, FK $\rightarrow$ bookings

5.8 Reviews

This table contains user reviews and ratings for completed flights.

Attribute	Description	Data Type	Constraints
review_id	Review ID	INTEGER	PRIMARY KEY
rating	Rating (1-5)	INTEGER	NOT NULL, CHECK (1-5)
review_comment	Review comment	TEXT	
review_date	Review date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
user_id	Reviewing user	INTEGER	NOT NULL, FK $\rightarrow$ users
flight_id	Reviewed flight	INTEGER	NOT NULL, FK $\rightarrow$ flights
(user_id, flight_id)	User/Flight pair		UNIQUE

5.9 Passengers

This table lists all passengers for each booking with their personal and travel details.

Attribute	Description	Data Type	Constraints
booking_id	Associated booking	INTEGER	PART OF PK, NOT NULL, FK → bookings
seat_number	Assigned seat	VARCHAR(5)	PART OF PK, NOT NULL
first_name	First name	VARCHAR(50)	NOT NULL
last_name	Last name	VARCHAR(50)	NOT NULL
date_of_birth	Date of birth	DATE	NOT NULL
passport_number	Passport number	VARCHAR(20)	NOT NULL
nationality	Nationality	VARCHAR(50)	NOT NULL

5.10 Flight Services

This table links flights and available services with their availability status.

Attribute	Description	Data Type	Constraints
flight_id	Flight	INTEGER	PART OF PK, NOT NULL, FK → flights
service_id	Service	INTEGER	PART OF PK, NOT NULL, FK → services
availability_status	Availability status	VARCHAR(20)	DEFAULT 'Available', CHECK ('Available', 'Unavailable', 'Limited')
last_updated	Last updated	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

6 SQL DDL Statements

6.1 Users Table

```sql
CREATE TABLE users (
    user_id INTEGER PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    phone_number VARCHAR(15) NOT NULL,
    role VARCHAR(20) DEFAULT 'Customer',
    CONSTRAINT check_user_role CHECK (role IN ('Customer',
        'Admin')),
    CONSTRAINT check_password_length CHECK
        (LENGTH(password_hash) >= 8)
);
```

6.2 Airports Table

```sql
CREATE TABLE airports (
    airport_id INTEGER PRIMARY KEY,
    airport_code VARCHAR(10) UNIQUE NOT NULL,
    airport_name VARCHAR(255) NOT NULL,
    city VARCHAR(100) NOT NULL,
    country VARCHAR(100) NOT NULL,
    timezone VARCHAR(50) NOT NULL
);
```

6.3 Airplanes Table

```sql
CREATE TABLE airplanes (
    airplane_id INTEGER PRIMARY KEY,
    model VARCHAR(50) NOT NULL,
    manufacturer VARCHAR(50) NOT NULL,
    capacity INTEGER NOT NULL,
    status VARCHAR(20) DEFAULT 'Active',
    CONSTRAINT check_airplane_capacity CHECK (capacity > 0),
    CONSTRAINT check_airplane_status CHECK (status IN ('Active',
        'Maintenance', 'Retired'))
);
```

6.4 Services Table

```sql
CREATE TABLE services (
    service_id INTEGER PRIMARY KEY,
    service_name VARCHAR(100) NOT NULL,
    description TEXT,
    price NUMERIC(10,2) NOT NULL,
    service_type VARCHAR(50) NOT NULL,
    CONSTRAINT check_service_price CHECK (price >= 0),
    CONSTRAINT check_service_type CHECK (service_type IN
       ('Food', 'Entertainment', 'Comfort', 'WiFi', 'Amenity'))
);
```

6.5 Flights Table

```sql
CREATE TABLE flights (
    flight_id INTEGER PRIMARY KEY,
    flight_number VARCHAR(10) UNIQUE NOT NULL,
    departure_time TIMESTAMP NOT NULL,
    arrival_time TIMESTAMP NOT NULL,
    price NUMERIC(10,2) NOT NULL,
    status VARCHAR(20) DEFAULT 'Scheduled',
    airplane_id INTEGER NOT NULL,
    source_airport_id INTEGER NOT NULL,
    destination_airport_id INTEGER NOT NULL,
    CONSTRAINT fk_flight_airplane FOREIGN KEY (airplane_id)
       REFERENCES airplanes(airplane_id),
    CONSTRAINT fk_flight_source FOREIGN KEY (source_airport_id)
       REFERENCES airports(airport_id),
    CONSTRAINT fk_flight_dest FOREIGN KEY
       (destination_airport_id) REFERENCES airports(airport_id),
    CONSTRAINT check_flight_price CHECK (price > 0),
    CONSTRAINT check_flight_status CHECK (status IN
       ('Scheduled', 'Delayed', 'Cancelled', 'Completed')),
    CONSTRAINT check_departure_before_arrival CHECK
       (departure_time < arrival_time),
    CONSTRAINT check_different_airports CHECK (source_airport_id
       != destination_airport_id)
);
```

6.6 Bookings Table

```sql
CREATE TABLE bookings (
    booking_id INTEGER PRIMARY KEY,
    booking_reference VARCHAR(10) UNIQUE NOT NULL,
    booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'Confirmed',
    user_id INTEGER NOT NULL,
```

```
    flight_id INTEGER NOT NULL,
    CONSTRAINT check_booking_status CHECK (status IN
       ('Confirmed', 'Cancelled', 'Pending')),
    CONSTRAINT fk_booking_user FOREIGN KEY (user_id) REFERENCES
       users(user_id),
    CONSTRAINT fk_booking_flight FOREIGN KEY (flight_id)
       REFERENCES flights(flight_id)
);
```

6.7 Payments Table

```
CREATE TABLE payments (
    payment_id INTEGER PRIMARY KEY,
    amount NUMERIC(10,2) NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    transaction_id VARCHAR(100) UNIQUE,
    payment_status VARCHAR(20) DEFAULT 'Pending',
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    booking_id INTEGER UNIQUE NOT NULL,
    CONSTRAINT check_payment_amount CHECK (amount > 0),
    CONSTRAINT check_payment_method CHECK (payment_method IN
       ('Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer')),
    CONSTRAINT check_payment_status CHECK (payment_status IN
       ('Pending', 'Completed', 'Failed', 'Refunded')),
    CONSTRAINT fk_payment_booking FOREIGN KEY (booking_id)
       REFERENCES bookings(booking_id)
);
```

6.8 Reviews Table

```
CREATE TABLE reviews (
    review_id INTEGER PRIMARY KEY,
    rating INTEGER NOT NULL,
    review_comment TEXT,
    review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_id INTEGER NOT NULL,
    flight_id INTEGER NOT NULL,
    CONSTRAINT check_review_rating CHECK (rating >= 1 AND rating
       <= 5),
    CONSTRAINT fk_review_user FOREIGN KEY (user_id) REFERENCES
       users(user_id),
    CONSTRAINT fk_review_flight FOREIGN KEY (flight_id)
       REFERENCES flights(flight_id),
    CONSTRAINT unique_user_flight_review UNIQUE(user_id,
       flight_id)
);
```

6.9 Passengers Table

```
CREATE TABLE passengers (
    booking_id INTEGER NOT NULL,
    seat_number VARCHAR(5) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    date_of_birth DATE NOT NULL,
    passport_number VARCHAR(20) NOT NULL,
    nationality VARCHAR(50) NOT NULL,
    CONSTRAINT pk_passenger PRIMARY KEY (booking_id,
       seat_number),
    CONSTRAINT fk_passenger_booking FOREIGN KEY (booking_id)
       REFERENCES bookings(booking_id) ON DELETE CASCADE
);
```

6.10 Flight Services Table

```
CREATE TABLE flight_services (
    flight_id INTEGER NOT NULL,
    service_id INTEGER NOT NULL,
    availability_status VARCHAR(20) DEFAULT 'Available',
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT check_flight_service_availability CHECK
       (availability_status IN ('Available', 'Unavailable',
       'Limited')),
    CONSTRAINT pk_flight_services PRIMARY KEY (flight_id,
       service_id),
    CONSTRAINT fk_flight_services_flight FOREIGN KEY (flight_id)
       REFERENCES flights(flight_id) ON DELETE CASCADE,
    CONSTRAINT fk_flight_services_service FOREIGN KEY
       (service_id) REFERENCES services(service_id) ON DELETE
       CASCADE
);
```

7 Sample Data

7.1 Users Table

```
INSERT ALL
  INTO users VALUES (1, 'Alice Smith', 'alice@example.com',
     'password123', '1234567890', 'Customer')
  INTO users VALUES (2, 'Bob Johnson', 'bob@example.com',
     'securepass', '2345678901', 'Customer')
  INTO users VALUES (3, 'admin', 'admin@example.com',
     'admin123', '3456789012', 'Admin')
  INTO users VALUES (4, 'David Kim', 'david@example.com',
     'passw0rd!', '4567890123', 'Customer')
  INTO users VALUES (5, 'Eva Brown', 'eva@example.com',
     'brownpass', '5678901234', 'Customer')
  INTO users VALUES (6, 'Frank Green', 'frank@example.com',
     'greenpass', '6789012345', 'Customer')
  INTO users VALUES (7, 'Grace Hall', 'grace@example.com',
     'hallpass1', '7890123456', 'Customer')
  INTO users VALUES (8, 'Henry King', 'henry@example.com',
     'kingpass2', '8901234567', 'Customer')
  INTO users VALUES (9, 'Ivy Moore', 'ivy@example.com',
     'moorepass', '9012345678', 'Customer')
  INTO users VALUES (10, 'Jack White', 'jack@example.com',
     'whitepass', '1122334455', 'Customer')
SELECT 1 FROM DUAL;
```

7.2 Airports Table

```
INSERT ALL
  INTO airports VALUES (1, 'JFK', 'John F. Kennedy
     International', 'New York', 'USA', 'UTC-5')
  INTO airports VALUES (2, 'LAX', 'Los Angeles International',
     'Los Angeles', 'USA', 'UTC-8')
  INTO airports VALUES (3, 'ORD', 'O''Hare International',
     'Chicago', 'USA', 'UTC-6')
  INTO airports VALUES (4, 'ATL', 'Hartsfield-Jackson Atlanta',
     'Atlanta', 'USA', 'UTC-5')
  INTO airports VALUES (5, 'LHR', 'Heathrow', 'London', 'UK',
     'UTC+0')
  INTO airports VALUES (6, 'DXB', 'Dubai International',
     'Dubai', 'UAE', 'UTC+4')
SELECT 1 FROM DUAL;
```

7.3 Airplanes Table

```
INSERT ALL
  INTO airplanes VALUES (1, '737-800', 'Boeing', 160, 'Active')
  INTO airplanes VALUES (2, 'A320', 'Airbus', 150, 'Active')
  INTO airplanes VALUES (3, '777-300', 'Boeing', 300,
     'Maintenance')
  INTO airplanes VALUES (4, 'A350', 'Airbus', 325, 'Active')
  INTO airplanes VALUES (5, '787-9', 'Boeing', 280, 'Retired')
  INTO airplanes VALUES (6, 'A321', 'Airbus', 185, 'Active')
SELECT 1 FROM DUAL;
```

7.4 Services Table

```
INSERT ALL
  INTO services VALUES (1, 'In-flight Meal', 'Meal service',
     25.00, 'Food')
  INTO services VALUES (2, 'WiFi', 'Internet access', 15.00,
     'WiFi')
  INTO services VALUES (3, 'Entertainment', 'Movies and TV',
     0.00, 'Entertainment')
  INTO services VALUES (4, 'Extra Legroom', 'More space', 35.00,
     'Comfort')
  INTO services VALUES (5, 'Priority Boarding', 'Board early',
     20.00, 'Amenity')
SELECT 1 FROM DUAL;
```

7.5 Flights Table

```
INSERT ALL
  INTO flights VALUES (1, 'AA101', TO_TIMESTAMP('2025-08-01
     08:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-01 12:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 300.00, 'Scheduled', 1, 1, 2)
  INTO flights VALUES (2, 'UA202', TO_TIMESTAMP('2025-08-02
     09:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-02 13:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 350.00, 'Scheduled', 2, 2, 3)
  INTO flights VALUES (3, 'DL303', TO_TIMESTAMP('2025-08-03
     10:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-03 14:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 400.00, 'Delayed', 3, 3, 4)
  INTO flights VALUES (4, 'BA404', TO_TIMESTAMP('2025-08-04
     11:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-04 15:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 450.00, 'Scheduled', 4, 4, 5)
  INTO flights VALUES (5, 'EK505', TO_TIMESTAMP('2025-08-05
     12:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-05 20:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 600.00, 'Scheduled', 5, 5, 6)
  INTO flights VALUES (6, 'LH606', TO_TIMESTAMP('2025-08-06
     13:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-06 17:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 320.00, 'Cancelled', 6, 6, 1)
  INTO flights VALUES (7, 'AF707', TO_TIMESTAMP('2025-08-07
     14:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-07 18:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 370.00, 'Scheduled', 1, 1, 3)
  INTO flights VALUES (8, 'SQ808', TO_TIMESTAMP('2025-08-08
     15:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-08 19:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 420.00, 'Scheduled', 2, 2, 4)
  INTO flights VALUES (9, 'QF909', TO_TIMESTAMP('2025-08-09
     16:00:00', 'YYYY-MM-DD HH24:S:SS'),
     TO_TIMESTAMP('2025-08-09 20:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 470.00, 'Scheduled', 3, 3, 5)
  INTO flights VALUES (10, 'AI010', TO_TIMESTAMP('2025-08-10
     17:00:00', 'YYYY-MM-DD HH24:MI:SS'),
     TO_TIMESTAMP('2025-08-10 21:00:00', 'YYYY-MM-DD
     HH24:MI:SS'), 520.00, 'Scheduled', 4, 4, 6)
SELECT 1 FROM DUAL;
```

7.6 Bookings Table

```
INSERT ALL
```

```
  INTO bookings VALUES (1, 'BK001', TO_TIMESTAMP('2025-07-01
    10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Confirmed', 1, 1)
  INTO bookings VALUES (2, 'BK002', TO_TIMESTAMP('2025-07-02
    11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Confirmed', 2, 2)
  INTO bookings VALUES (3, 'BK003', TO_TIMESTAMP('2025-07-03
    12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Confirmed', 3, 3)
  INTO bookings VALUES (4, 'BK004', TO_TIMESTAMP('2025-07-04
    13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Cancelled', 4, 4)
  INTO bookings VALUES (5, 'BK005', TO_TIMESTAMP('2025-07-05
    14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Confirmed', 5, 5)
SELECT 1 FROM DUAL;
```

7.7 Passengers Table

```
INSERT ALL
  INTO passengers VALUES (1, '12A', 'Alice', 'Smith',
    TO_DATE('1990-01-01', 'YYYY-MM-DD'), 'P123456', 'USA')
  INTO passengers VALUES (2, '14B', 'Bob', 'Johnson',
    TO_DATE('1985-02-02', 'YYYY-MM-DD'), 'P234567', 'USA')
  INTO passengers VALUES (3, '15C', 'Carol', 'Lee',
    TO_DATE('1992-03-03', 'YYYY-MM-DD'), 'P345678', 'UK')
  INTO passengers VALUES (3, '15D', 'David', 'Kim',
    TO_DATE('1991-04-04', 'YYYY-MM-DD'), 'P456789', 'KOR')
  INTO passengers VALUES (4, '16A', 'Eva', 'Brown',
    TO_DATE('1988-05-05', 'YYYY-MM-DD'), 'P567890', 'USA')
  INTO passengers VALUES (4, '16B', 'Frank', 'Green',
    TO_DATE('1987-06-06', 'YYYY-MM-DD'), 'P678901', 'USA')
  INTO passengers VALUES (4, '16C', 'Grace', 'Hall',
    TO_DATE('1993-07-07', 'YYYY-MM-DD'), 'P789012', 'USA')
  INTO passengers VALUES (5, '17A', 'Henry', 'King',
    TO_DATE('1986-08-08', 'YYYY-MM-DD'), 'P890123', 'UK')
SELECT 1 FROM DUAL;
```

7.8 Payments Table

```
INSERT ALL
  INTO payments VALUES (1, 300.00, 'Credit Card', 'TXN001',
    'Completed', TO_TIMESTAMP('2025-07-01 10:05:00',
    'YYYY-MM-DD HH24:MI:SS'), 1)
  INTO payments VALUES (2, 350.00, 'Debit Card', 'TXN002',
    'Completed', TO_TIMESTAMP('2025-07-02 11:05:00',
    'YYYY-MM-DD HH24:MI:SS'), 2)
  INTO payments VALUES (3, 400.00, 'PayPal', 'TXN003',
    'Completed', TO_TIMESTAMP('2025-07-03 12:05:00',
    'YYYY-MM-DD HH24:MI:SS'), 3)
```

```
  INTO payments VALUES (4, 450.00, 'Credit Card', 'TXN004',
    'Refunded', TO_TIMESTAMP('2025-07-04 13:05:00', 'YYYY-MM-DD
    HH24:MI:SS'), 4)
  INTO payments VALUES (5, 600.00, 'Bank Transfer', 'TXN005',
    'Completed', TO_TIMESTAMP('2025-07-05 14:05:00',
    'YYYY-MM-DD HH24:MI:SS'), 5)
SELECT 1 FROM DUAL;
```

7.9 Reviews Table

```
INSERT ALL
  INTO reviews VALUES (1, 5, 'Great flight!',
    TO_TIMESTAMP('2025-08-01 18:00:00', 'YYYY-MM-DD
    HH24:MI:SS'), 1, 1)
  INTO reviews VALUES (2, 4, 'Comfortable and on time.',
    TO_TIMESTAMP('2025-08-01 18:30:00', 'YYYY-MM-DD
    HH24:MI:SS'), 2, 1)
  INTO reviews VALUES (3, 3, 'Average experience.',
    TO_TIMESTAMP('2025-08-02 19:00:00', 'YYYY-MM-DD
    HH24:MI:SS'), 3, 2)
  INTO reviews VALUES (4, 2, 'Delayed flight.',
    TO_TIMESTAMP('2025-08-03 20:00:00', 'YYYY-MM-DD
    HH24:MI:SS'), 4, 3)
  INTO reviews VALUES (5, 5, 'Excellent service!',
    TO_TIMESTAMP('2025-08-04 21:00:00', 'YYYY-MM-DD
    HH24:MI:SS'), 5, 4)
SELECT 1 FROM DUAL;
```

7.10 Flight Services Table

```
INSERT ALL
  INTO flight_services VALUES (1, 1, 'Available',
    TO_TIMESTAMP('2025-07-01 09:00:00', 'YYYY-MM-DD
    HH24:MI:SS'))
  INTO flight_services VALUES (1, 2, 'Available',
    TO_TIMESTAMP('2025-07-01 09:00:00', 'YYYY-MM-DD
    HH24:MI:SS'))
  INTO flight_services VALUES (1, 3, 'Available',
    TO_TIMESTAMP('2025-07-01 09:00:00', 'YYYY-MM-DD
    HH24:MI:SS'))
  INTO flight_services VALUES (2, 1, 'Available',
    TO_TIMESTAMP('2025-07-02 09:00:00', 'YYYY-MM-DD
    HH24:MI:SS'))
  INTO flight_services VALUES (2, 2, 'Limited',
    TO_TIMESTAMP('2025-07-02 09:00:00', 'YYYY-MM-DD
    HH24:MI:SS'))
```

```
  INTO flight_services VALUES (3, 3, 'Unavailable',
     TO_TIMESTAMP('2025-07-03 09:00:00', 'YYYY-MM-DD
     HH24:MI:SS'))
  INTO flight_services VALUES (3, 4, 'Available',
     TO_TIMESTAMP('2025-07-03 09:00:00', 'YYYY-MM-DD
     HH24:MI:SS'))
  INTO flight_services VALUES (4, 5, 'Available',
     TO_TIMESTAMP('2025-07-04 09:00:00', 'YYYY-MM-DD
     HH24:MI:SS'))
  INTO flight_services VALUES (5, 1, 'Available',
     TO_TIMESTAMP('2025-07-05 09:00:00', 'YYYY-MM-DD
     HH24:MI:SS'))
SELECT 1 FROM DUAL;
```

8 Query Examples

This section demonstrates various SQL queries that can be executed on our database to retrieve useful information.

8.1 Retrieve booking and payment information for all bookings that have associated payments.

Query Statement:

```
SELECT booking_reference, amount, payment_method, payment_status
FROM bookings
NATURAL JOIN payments;
```

Relational Algebra:

$$\Pi_{\text{booking\_reference, amount, payment\_method, payment\_status}}(\text{bookings} \bowtie \text{payments})$$

BOOKING_REFERENCE	AMOUNT	PAYMENT_METHOD	PAYMENT_STATUS
BK001	300	Credit Card	Completed
BK002	350	Debit Card	Completed
BK003	400	PayPal	Completed
BK004	450	Credit Card	Refunded
BK005	600	Bank Transfer	Completed

Figure 2: Natural Join Query Result

8.2 Find all bookings with user and flight details using explicit JOIN

Query Statement:

```sql
SELECT b.booking_id, u.full_name, u.email, f.flight_number,
   f.departure_time, f.price, b.status
FROM bookings b
JOIN users u ON b.user_id = u.user_id
JOIN flights f ON b.flight_id = f.flight_id
WHERE b.status = 'Confirmed';
```

Relational Algebra:

$$\Pi_{\text{booking\_id, full\_name, email, flight\_number, departure\_time, price, status}} (\sigma_{\text{status='Confirmed'}} (\text{bookings} \bowtie \text{users} \bowtie \text{flights}))$$

BOOKING_ID	FULL_NAME	EMAIL	FLIGHT_NUMBER	DEPARTURE_TIME	PRICE	STATUS
1	Alice Smith	alice@example.com	AA101	01-AUG-25 08.00.00.000000 AM	300	Confirmed
2	Bob Johnson	bob@example.com	UA202	02-AUG-25 09.00.00.000000 AM	350	Confirmed
3	admin	admin@example.com	DL303	03-AUG-25 10.00.00.000000 AM	400	Confirmed
5	Eva Brown	eva@example.com	EK505	05-AUG-25 12.00.00.000000 PM	600	Confirmed

Figure 3: Explicit JOIN Query Result

8.3 All possible combinations of payment methods and payment statuses

Query Statement:

```sql
SELECT pm.payment_method, ps.payment_status
FROM (SELECT DISTINCT payment_method FROM payments) pm
CROSS JOIN (SELECT DISTINCT payment_status FROM payments) ps
ORDER BY pm.payment_method, ps.payment_status;
```

Relational Algebra:

$$\Pi_{\text{payment\_method, payment\_status}} (\text{payment\_methods} \times \text{payment\_statuses})$$

PAYMENT_METHOD	PAYMENT_STATUS
Bank Transfer	Completed
Bank Transfer	Refunded
Credit Card	Completed
Credit Card	Refunded
Debit Card	Completed
Debit Card	Refunded
PayPal	Completed
PayPal	Refunded

Figure 4: CROSS JOIN of Payment Methods and Statuses

8.4 All users with their bookings (including users who haven't booked)

Query Statement:

```sql
SELECT u.full_name, u.email, b.booking_reference,
   b.booking_date, b.status
FROM bookings b
RIGHT OUTER JOIN users u ON b.user_id = u.user_id
ORDER BY u.full_name;
```

Relational Algebra:

$$\Pi_{\text{full\_name, email, booking\_reference, booking\_date, status}}\Big(\big(\text{bookings} \ \bowtie^{r}_{\text{bookings.user\_id=users.user\_id}} \ \text{users}\big)\Big)$$

FULL_NAME	EMAIL	BOOKING_REFERENCE	STATUS
Alice Smith	alice@example.com	BK001	Confirmed
Bob Johnson	bob@example.com	BK002	Confirmed
David Kim	david@example.com	BK004	Cancelled
Eva Brown	eva@example.com	BK005	Confirmed
Frank Green	frank@example.com	-	-
Grace Hall	grace@example.com	-	-
Henry King	henry@example.com	-	-
Ivy Moore	ivy@example.com	-	-
Jack White	jack@example.com	-	-

Figure 5: Right Outer Join: All Users with Bookings (NULLs for users without bookings)

8.5 All flights with their reviews (including flights with no reviews)

Query Statement:

```sql
SELECT f.flight_number, r.rating, r.review_comment, u.full_name
   as reviewer
FROM flights f
LEFT OUTER JOIN reviews r ON f.flight_id = r.flight_id
LEFT OUTER JOIN users u ON r.user_id = u.user_id
ORDER BY f.flight_number;
```

Relational Algebra:

$$\Pi_{\text{flight\_number, rating, review\_comment, full\_name}}\Big(\big(\text{flights} \ \bowtie^{l}_{\text{flights.flight\_id=reviews.flight\_id}} \ \text{reviews}\big) \ \bowtie^{l}_{\text{reviews.user\_id=users.user\_id}} \ \text{users}\Big)$$

FLIGHT_NUMBER	RATING	REVIEW_COMMENT	REVIEWER
AA101	5	Great flight!	Alice Smith
AA101	4	Comfortable and on time.	Bob Johnson
AF707	-	-	-
AI010	-	-	-
BA404	5	Excellent service!	Eva Brown
DL303	2	Delayed flight.	David Kim
EK505	-	-	-
LH606	-	-	-
QF909	-	-	-
SQ808	-	-	-

Figure 6: Left Outer Join: All Flights with Reviews (NULLs for flights without reviews)

8.6 Query: Confirmed Bookings with Passenger Details (Sorted)

```sql
SELECT booking_reference, status, first_name, last_name,
   seat_number, nationality
FROM bookings b
JOIN passengers p USING (booking_id)
WHERE b.status = 'Confirmed'
ORDER BY booking_id, p.seat_number;
```

Relational Algebra:

$$\tau_{\text{booking\_id,seat\_number}}\Big(\Pi_{\text{booking\_reference, status, first\_name, last\_name, seat\_number, nationality}}\Big(\sigma_{\text{status='Confirmed'}}\big(\text{bookings} \bowtie_{\text{booking\_id}} \text{passengers}\big)\Big)\Big)$$

BOOKING_REFERENCE	STATUS	FIRST_NAME	LAST_NAME	SEAT_NUMBER	NATIONALITY
BK001	Confirmed	Alice	Smith	12A	USA
BK002	Confirmed	Bob	Johnson	14B	USA
BK003	Confirmed	Carol	Lee	15C	UK
BK003	Confirmed	David	Kim	15D	KOR
BK005	Confirmed	Henry	King	17A	UK

Figure 7: Using syntax)

8.7 Find flights that have no reviews

Query Statement:

```sql
SELECT flight_id, flight_number, departure_time, price
FROM flights f
WHERE NOT EXISTS (
    SELECT 1 FROM reviews r
    WHERE r.flight_id = f.flight_id
)
ORDER BY departure_time;
```

Relational Algebra:

$$\Pi_{\text{flight\_id, flight\_number, departure\_time, price}}\Big(\sigma_{\neg\exists r \in \text{reviews}\ (r.\text{flight\_id}=f.\text{flight\_id})}(\text{flights } f)\Big)$$

FLIGHT_ID	FLIGHT_NUMBER	DEPARTURE_TIME	PRICE
5	EK505	05-AUG-25 12.00.00.000000 PM	600
6	LH606	06-AUG-25 01.00.00.000000 PM	320
7	AF707	07-AUG-25 02.00.00.000000 PM	370
8	SQ808	08-AUG-25 03.00.00.000000 PM	420
9	QF909	09-AUG-25 04.00.00.000000 PM	470
10	AI010	10-AUG-25 05.00.00.000000 PM	520

Figure 8: Flights with No Reviews

8.8 Find users who have made at least one booking

Query Statement:

```sql
SELECT user_id, full_name, email, role
FROM users u
WHERE EXISTS (
    SELECT 1 FROM bookings b
    WHERE b.user_id = u.user_id
)
ORDER BY full_name;
```

Relational Algebra:

$$\Pi_{\text{user\_id, full\_name, email, role}}\Big(\sigma_{\exists b \in \text{bookings}\ (b.\text{user\_id}=u.\text{user\_id})}(\text{users } u)\Big)$$

USER_ID	FULL_NAME	EMAIL	ROLE
1	Alice Smith	alice@example.com	Customer
2	Bob Johnson	bob@example.com	Customer
4	David Kim	david@example.com	Customer
5	Eva Brown	eva@example.com	Customer

Figure 9: Users who have made at least one booking

8.9 Find flights more expensive than ANY Boeing aircraft flight

Query Statement:

```sql
SELECT f.flight_number, f.price, a.model, a.manufacturer
FROM flights f
JOIN airplanes a ON f.airplane_id = a.airplane_id
WHERE f.price > ANY (
    SELECT f2.price
    FROM flights f2
    JOIN airplanes a2 ON f2.airplane_id = a2.airplane_id
    WHERE a2.manufacturer = 'Boeing'
)
ORDER BY f.price DESC;
```

Relational Algebra:

$$\tau_{\text{price DESC}}\, \Pi_{\text{flight\_number, price, model, manufacturer}} \Big(\sigma_{\text{price} > \text{SOME}\left(\Pi_{\text{price}}\left(\sigma_{\text{manufacturer}='Boeing'}(\text{flights} \bowtie \text{airplanes})\right)\right)} (\text{flights} \bowtie \text{airplanes}) \Big)$$

FLIGHT_NUMBER	PRICE	MODEL	MANUFACTURER
EK505	600	787-9	Boeing
AI010	520	A350	Airbus
QF909	470	777-300	Boeing
BA404	450	A350	Airbus
SQ808	420	A320	Airbus
DL303	400	777-300	Boeing
AF707	370	737-800	Boeing
UA202	350	A320	Airbus
LH606	320	A321	Airbus

Figure 10: Flights more expensive than ANY Boeing aircraft flight

8.10 Find flights with above-average price

Query Statement:

```sql
SELECT flight_number, price
FROM flights
WHERE price > (
    SELECT AVG(price)
    FROM flights
);
```

Relational Algebra:

$$\Pi_{\text{flight\_number,price}} \left(\sigma_{\text{price} > \text{avg\_price}} \left(\text{flights} \times \gamma_{\text{avg\_price}:=\text{AVG(price)}}(\text{flights}) \right) \right)$$

FLIGHT_NUMBER	PRICE
BA404	450
EK505	600
QF909	470
AI010	520

Figure 11: Flights with price above average

8.11 Find the most expensive flights (more expensive than ALL other flights)

Query Statement:

```sql
SELECT f.flight_number, f.price, src.city, dest.city
FROM flights f
JOIN airports src ON f.source_airport_id = src.airport_id
JOIN airports dest ON f.destination_airport_id = dest.airport_id
WHERE f.price >= ALL (SELECT price FROM flights)
ORDER BY f.price DESC;
```

Relational Algebra:

$$\tau_{\text{price DESC}}\,\Pi_{\text{flight\_number,price,source,destination}}\Big(\sigma_{\text{price}\geq\text{ALL}\left(\Pi_{\text{price}}(\text{flights})\right)}\Big(\big(\text{flights}\bowtie_{\text{flights.source\_airport\_id}=\text{src.airport\_id}}\text{airports as src}\big)\bowtie_{\text{flights.destination\_airport\_id}=\text{dest.airport\_id}}\text{airports as dest}\Big)\Big)$$

FLIGHT_NUMBER	PRICE	SOURCE	DESTINATION
EK505	600	London	Dubai

Figure 12: Most expensive flights (price greater than or equal to all others)

8.12 Show each user with their total number of bookings and average payment

Query Statement:

```sql
SELECT u.full_name, u.email,
       (SELECT COUNT(*) FROM bookings b WHERE b.user_id =
          u.user_id) as total_bookings,
       (SELECT COALESCE(AVG(p.amount), 0) FROM bookings b2
        JOIN payments p ON b2.booking_id = p.booking_id
        WHERE b2.user_id = u.user_id) as avg_payment
FROM users u
ORDER BY total_bookings DESC;
```

Relational Algebra:

$$\tau_{\text{total\_bookings DESC}}\,\Pi_{\text{full\_name, email, total\_bookings, avg\_payment}}\Big(\text{users} \;\overset{l}{\bowtie}_{\text{users.user\_id=b.user\_id}}\; \gamma_{\text{user\_id; COUNT(*)}\to\text{total\_bookings}}(\text{bookings as b}) \;\overset{l}{\bowtie}_{\text{users.user\_id=b2.user\_id}}\; \gamma_{\text{user\_id; AVG(amount)}\to\text{avg\_payment}}(\text{bookings as b2} \bowtie \text{payments})\Big)$$

FULL_NAME	EMAIL	TOTAL_BOOKINGS	AVG_PAYMENT
Alice Smith	alice@example.com	1	300
Bob Johnson	bob@example.com	1	350
admin	admin@example.com	1	400
David Kim	david@example.com	1	450
Eva Brown	eva@example.com	1	600
Frank Green	frank@example.com	0	0
Grace Hall	grace@example.com	0	0
Henry King	henry@example.com	0	0
Ivy Moore	ivy@example.com	0	0
Jack White	jack@example.com	0	0

Figure 13: Users with total bookings and average payment

8.13 Airports with more than 1 departure (simple GROUP BY and HAVING)

Query Statement:

```sql
SELECT src.airport_code, src.city, COUNT(f.flight_id) AS
   departing_flights
FROM flights f
JOIN airports src ON f.source_airport_id = src.airport_id
GROUP BY src.airport_code, src.city
HAVING COUNT(f.flight_id) > 1
```

Relational Algebra:

$$\Pi_{\text{airport\_code, city, departing\_flights}}\Big(\sigma_{\text{departing\_flights}>1}\Big(\gamma_{\text{src.airport\_code, src.city; departing\_flights}\leftarrow\text{COUNT}(f.\text{flight\_id})}\big(\text{flights} \bowtie_{\text{flights.source\_airport\_id}=\text{src.airport\_id}} \text{airports as src}\big)\Big)\Big)$$

AIRPORT_CODE	CITY	DEPARTING_FLIGHTS
JFK	New York	2
ATL	Atlanta	2
ORD	Chicago	2
LAX	Los Angeles	2

Figure 14: Airports with more than 1 departure

8.14 Boeing flights with price above 500 (using WITH clause)

Query Statement:

```sql
WITH boeing_flights AS (
    SELECT flight_id, flight_number, price
```

```
    FROM flights f
    JOIN airplanes a ON f.airplane_id = a.airplane_id
    WHERE a.manufacturer = 'Boeing'
)
SELECT flight_number, price
FROM boeing_flights
WHERE price > 500;
```

Relational Algebra:

$$\text{Let } B \leftarrow \sigma_{\text{manufacturer}='Boeing'}(\text{flights} \bowtie \text{airplanes})$$
$$\Pi_{\text{flight\_number, price}}\left(\sigma_{\text{price}>500}(B)\right)$$

FLIGHT_NUMBER	PRICE
EK505	600

Figure 15: Boeing flights with price above 500

8.15 Find airports with 'International' in their name

Query Statement:

```
SELECT airport_code, airport_name, city
FROM airports
WHERE airport_name LIKE '%International%';
```

Relational Algebra:

$$\Pi_{\text{airport\_code, airport\_name, city}}\Big(\sigma_{\text{airport\_name LIKE '\%International\%'}}(\text{airports})\Big)$$

AIRPORT_CODE	AIRPORT_NAME	CITY
JFK	John F. Kennedy International	New York
LAX	Los Angeles International	Los Angeles
ORD	O'Hare International	Chicago
DXB	Dubai International	Dubai

Figure 16: Airports with 'International' in their name

8.16 Find users who have both bookings AND reviews

Query Statement:

```sql
SELECT user_id, full_name
FROM users
WHERE EXISTS (SELECT 1 FROM bookings WHERE bookings.user_id =
   users.user_id)
  AND EXISTS (SELECT 1 FROM reviews WHERE reviews.user_id =
     users.user_id);
```

Relational Algebra:

$$\Pi_{\text{user\_id, full\_name}}\Big(\sigma_{\exists b \in \text{bookings}(b.\text{user\_id}=u.\text{user\_id}) \wedge \exists r \in \text{reviews}(r.\text{user\_id}=u.\text{user\_id})}(\text{users } u)\Big)$$

USER_ID	FULL_NAME
1	Alice Smith
2	Bob Johnson
3	admin
4	David Kim
5	Eva Brown

Figure 17: Users who have both bookings and reviews

8.17 Increase Flight Price by 100 for Flights Below 350

Query Statement:

```
UPDATE flights
SET base_price = base_price + 100
WHERE base_price < 350;
```

8.18 Delete Flights Below 350

Query Statement:

```
DELETE FROM flights
WHERE base_price < 350;
```

9 Views

Views provide a way to create virtual tables that can simplify complex queries and provide an additional layer of security.

9.1 Creating Views

9.1.1 flight_with_airport_view

This view provides flight details along with the source airport code and city.

View Definition:

```
CREATE VIEW flight_with_airport_view AS
SELECT
    f.flight_id,
    f.flight_number,
    f.departure_time,
    f.price,
    f.status,
    a.airport_code AS source_code,
    a.city AS source_city
FROM flights f
JOIN airports a ON f.source_airport_id = a.airport_id;
```

9.1.2 booking_details_view

This view provides booking details along with the user's name and email.

View Definition:

```
CREATE VIEW booking_details_view AS
SELECT
    b.booking_id,
    b.booking_reference,
    b.booking_date,
    b.status,
    u.full_name,
    u.email
FROM bookings b
JOIN users u ON b.user_id = u.user_id;
```

9.2 Querying Views

Example: Query confirmed bookings using the booking_details_view.

Query Statement:

```
SELECT booking_reference, booking_date, status, full_name
FROM booking_details_view
WHERE status = 'Confirmed';
```

BOOKING_REFERENCE	BOOKING_DATE	STATUS	FULL_NAME
BK001	01-JUL-25 10.00.00.000000 AM	Confirmed	Alice Smith
BK002	02-JUL-25 11.00.00.000000 AM	Confirmed	Bob Johnson
BK003	03-JUL-25 12.00.00.000000 PM	Confirmed	admin
BK005	05-JUL-25 02.00.00.000000 PM	Confirmed	Eva Brown

Figure 18: Querying confirmed bookings from the booking_details_view

10 Functional Dependencies and Normalization

10.1 Functional Dependencies

The following functional dependencies exist in our database:

10.1.1 Users Table

- user_id $\rightarrow$ full_name, email, password_hash, phone_number, role

  The user_id uniquely identifies each user, and all other attributes depend solely on it.

10.1.2 Airports Table

- airport_id $\rightarrow$ airport_code, airport_name, city, country, timezone

  The airport_id uniquely identifies each airport, and all other attributes depend solely on it.

- airport_code $\rightarrow$ airport_id, airport_name, city, country, timezone

  The airport_code is unique and determines all other airport attributes.

10.1.3 Airplanes Table

- airplane_id $\rightarrow$ model, manufacturer, capacity, status

  The airplane_id uniquely identifies each airplane, and all other attributes depend solely on it.

10.1.4 Services Table

- service_id $\rightarrow$ service_name, description, price, service_type

  The service_id uniquely identifies each service, and all other attributes depend solely on it.

10.1.5 Flights Table

- flight_id $\rightarrow$ flight_number, departure_time, arrival_time,
  price, status, airplane_id, source_airport_id, destination_airport_id

  The flight_id uniquely identifies each flight, and all other attributes depend solely on it.

- flight_number $\rightarrow$ flight_id, departure_time, arrival_time,
  price, status, airplane_id, source_airport_id, destination_airport_id

  The flight_number is unique and determines all other flight attributes.

10.1.6 Bookings Table

- booking_id $\rightarrow$ booking_reference, booking_date, status, user_id, flight_id

  The booking_id uniquely identifies each booking, and all other attributes depend solely on it.

- booking_reference $\rightarrow$ booking_id, booking_date, status, user_id, flight_id

  The booking_reference is unique and determines all other booking attributes.

10.1.7 Payments Table

- payment_id $\rightarrow$ amount, payment_method, transaction_id, payment_status, payment_date, booking_id

  The payment_id uniquely identifies each payment, and all other attributes depend solely on it.

- transaction_id $\rightarrow$ payment_id, amount, payment_method, payment_status, payment_date, booking_id

  The transaction_id is unique and determines all other payment attributes.

10.1.8 Reviews Table

- review_id $\rightarrow$ rating, review_comment, review_date, user_id, flight_id

  The review_id uniquely identifies each review, and all other attributes depend solely on it.

- $(\text{user\_id}, \text{flight\_id}) \rightarrow$ review_id, rating, review_comment, review_date

  Each user can review a flight only once, so the pair $(\text{user\_id}, \text{flight\_id})$ determines the review.

10.1.9 Passengers Table

- $(\text{booking\_id}, \text{seat\_number}) \rightarrow$ first_name, last_name, date_of_birth, passport_number, nationality

  The composite key $(\text{booking\_id}, \text{seat\_number})$ uniquely identifies each passenger and all other attributes depend solely on it.

10.1.10 Flight_Services Table

- $(\text{flight\_id}, \text{service\_id}) \rightarrow \text{availability\_status}, \text{last\_updated}$

  The composite key $(\text{flight\_id}, \text{service\_id})$ uniquely identifies each flight-service relationship and all other attributes depend solely on it.

10.2 Normalization Analysis

Our database schema is designed to be in Boyce-Codd Normal Form (BCNF) and Third Normal Form (3NF). Here's the detailed analysis:

10.2.1 First Normal Form (1NF)

All tables satisfy 1NF because:

- Each table has a primary key or composite key.
  - For example, the Users table has `user_id` as its primary key.
  - The Passengers table has (`booking_id`, `seat_number`) as its composite primary key.
- All attributes contain atomic (indivisible) values.
  - For example, the `full_name` attribute in the Users table stores a single value (e.g., "John Doe").
- No repeating groups exist.
  - For example, the Bookings table does not store multiple booking references in a single cell.

10.2.2 Second Normal Form (2NF)

All tables satisfy 2NF because:

- They are in 1NF.
- All non-key attributes are fully functionally dependent on the whole primary key.

- For example, in the Flights table, all attributes (`flight_number`, `departure_time`, etc.) depend entirely on the primary key `flight_id`.
- Junction tables (e.g., Passengers, Flight_Services) have composite primary keys with no partial dependencies.

10.2.3 Third Normal Form (3NF)

All tables satisfy 3NF because:

- They are in 2NF.
- No transitive dependencies exist.
  - Non-key attributes depend only on the primary key, not on other non-key attributes.
  - For example, in the Airports table, `city` depends only on `airport_id`, not on `airport_code` or `country`.

10.2.4 Proof of BCNF

To prove that the database is in BCNF, consider the following:

- **Users Table:**
  - All attributes (`full_name`, `email`, etc.) depend directly on `user_id`, which is the only candidate key.
  - No non-trivial functional dependency has a determinant that is not a superkey.
- **Flights Table:**
  - All attributes (`flight_number`, `departure_time`, etc.) depend directly on `flight_id`, which is the only candidate key.
  - No non-trivial functional dependency has a determinant that is not a superkey.
- **Passengers Table:**
  - All attributes (`first_name`, `last_name`, etc.) depend directly on the composite key (`booking_id`, `seat_number`).
  - No non-trivial functional dependency has a determinant that is not a superkey.

- **Reviews Table:**
  - All attributes (`rating`, `review_comment`, etc.) depend directly on the candidate key (`user_id`, `flight_id`).
  - No non-trivial functional dependency has a determinant that is not a superkey.
- All other tables follow the same pattern: every non-trivial functional dependency has a superkey as its determinant.

Thus, the database schema is fully normalized to BCNF (and therefore also 3NF), ensuring data integrity, minimizing redundancy, and optimizing query performance.

11 Frontend Designing Tools and Techniques

11.1 Technologies Used

The frontend development of the Airline Management System utilized modern web technologies including **React.js** as the primary framework, **Tailwind CSS** for styling and responsive design, and **Context API** for state management across components.

11.2 Implementation Techniques

The development approach focused on **component architecture** principles, ensuring modular and reusable code structures. **User experience (UX) design principles** were applied throughout the interface design to create intuitive and accessible user interactions.

11.3 Development Tools

The development environment included **Create React App** for project scaffolding and build configuration, **Visual Studio Code** as the primary integrated development environment, and comprehensive use of **browser developer tools** for debugging and optimization.

11.4 Advantages and Disadvantages

11.4.1 Advantages

- **Component reusability and Virtual DOM efficiency**: React's component-based architecture promotes code reuse and ensures optimal rendering performance through the Virtual DOM implementation.
- **Rapid development with Tailwind CSS**: The utility-first CSS framework accelerates the development process while maintaining consistent design patterns.
- **Modern development practices and scalable architecture**: The chosen technology stack follows current industry standards and supports application scalability.

11.4.2 Disadvantages

- **Learning curve and bundle size concerns**: The complexity of modern frontend frameworks requires significant learning investment, and can result in larger application bundles.
- **State management complexity**: Managing application state across multiple components can become challenging as the application grows.
- **Browser compatibility challenges**: Ensuring consistent functionality across different browsers and versions requires additional testing and polyfills.

12 Conclusion

The airline management database system was successfully designed to model a real-world flight booking platform, incorporating key entities such as users, flights, airports, bookings, payments, and services. Each table was created with meaningful constraints and normalization principles to ensure data integrity, consistency, and efficient querying.

Through the use of SQL and relational algebra, various complex queries—such as aggregations, nested subqueries, joins, and outer joins—were effectively translated and analyzed. This not only demonstrated a practical understanding of database operations but also reinforced the theoretical foundations of relational models.

Overall, the lab work provided valuable experience in schema design, query formulation, and optimization techniques, all of which are essential for managing real-world databases in airline or other logistics-based domains.