Egypt-Japan University of Science and Technology
الجامعة المصرية اليابانية للعلوم و التكنولوجيا
エジプト日本科学技術大学

Database Systems CNC-314
Dr / Mohamed Issa
Airline Reservation Systems

Team Members:

- Nasser Hossam Hamed (ID: 320230051)
- Nour Mohamed Ashry (ID: 320230018)
- Mariam Ahmed Shawky (ID: 320230056)
- Malak Mohamed Aboelgheit (ID: 320230004)
- Youssef Ashraf Elmegharbel (ID: 320230026)

Table of Contents

- Flight
- Ticket Type
- Booking
- Payment Method
- Payment
- Flight_Seat
- Booking_Seat

11. **Relationship Details**
    11.1 User → Passenger
    11.2 Passenger → Booking
    11.3 Booking → Payment
    11.4 Booking → Flight
    11.5 Airline Company → Flight
    11.6 Flight → Route
    11.7 Flight → Aircraft
    11.8 Aircraft → Seat
    11.9 Aircraft → Crew
    11.10 Booking ↔ Seat

12. **SQL Queries and Analysis**
    12.1 Basic Flight Search
    12.2 Revenue Analysis by Route
    12.3 Top 10 Passengers by Total Spending
    12.4 Payment Method Analysis
    12.5 Route Distance and Flight Time Analysis
    12.6 Flight Occupancy Report
    12.7 Complete Passenger Booking Details
    12.8 Basic Booking and Passenger Information

13. **Graphical User Interface (GUI)**

1. Introduction

The Airline Reservation System is designed to streamline the process of booking, managing, and scheduling flights. It ensures efficient management of airline operations, including customer reservations, seat allocation, and fare calculation. This system will enable travelers to book flights easily while providing airlines with tools to manage schedules and passenger data effectively.

---

2. Scope

The system will provide end-to-end functionality for flight reservations, including:

- Managing flight schedules and seat availability.
- Allowing customers to book tickets and select seats.
- Handling fare calculations, including taxes, discounts, and multi-currency support.
- Supporting various user roles, such as administrators, agents, and travelers.
- Integrating with payment gateways for secure transactions.
- Maintaining records of airlines, airplanes, airports, and passengers.

The system will cover domestic and international flights, supporting a wide range of airlines and ensuring compliance with industry standards.

3.Problem Description

Airline operations require accurate management of flight schedules, seat availability, passenger data, and payments. Manual or poorly integrated systems can lead to booking errors, inefficient operations, and poor customer experience. A centralized airline reservation system is needed to manage these processes efficiently and securely using a structured database.

---

4. Purpose of the System

The purpose of the Airline Reservation System is to automate flight booking and airline management processes. It provides travelers with an easy way to search and reserve flights while enabling airline staff to manage flights, seats, fares, and passengers

efficiently through a reliable database system.

---

5. System Overview

5.1 Airline Reservation System Explanation

The Airline Reservation System is a database-driven application designed to manage flight operations, reservations, travelers, and payments. It provides real-time access to flight schedules, seat availability, and booking information while maintaining data integrity and security.

5.2 Main Functionalities:

- Flight search and scheduling
- Seat reservation and availability tracking
- Traveler registration and booking management
- Fare calculation and payment processing
- Role-based access for admins, agents, and travelers
- Reporting and analytics

---

6. Non-Functional Requirements

6.1 Performance Requirements

- The system should handle up to 10,000 simultaneous user sessions.
- Flight searches should be processed within 2 seconds.

6.2 Security Requirements

- Encrypt sensitive data, including personal details and payment information.
- Ensure role-based access to sensitive functionalities.

6.3 Usability Requirements

● Provide a simple, intuitive GUI for all user roles.

6.4 Availability Requirements

● The system should be operational 99.9% of the time.

6.5 Scalability Requirements

● Support the addition of new airports, airplanes, and routes without performance degradation.

---

7. System Architecture

The system will use a three-tier architecture:

1. Presentation Layer: Web-based GUI for travelers, admins, and agents.
2. Application Layer: Business logic for flight search, reservations, and payment processing.
3. Database Layer: Centralized storage for all flight, user, and booking data.

---

8. Stakeholders in the Airline Reservation System

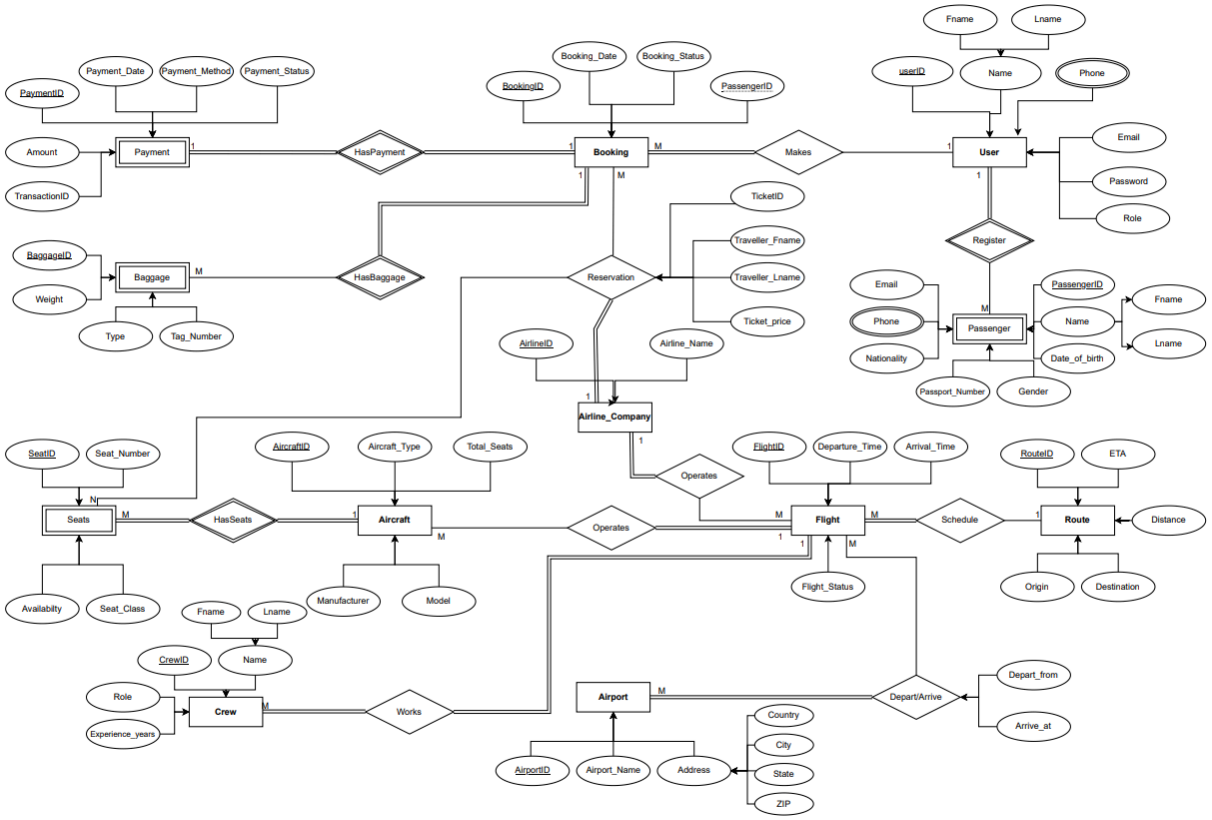
In the context of the Airline Reservation System, stakeholders include all parties who interact with the system or depend on it for airline operations, reservations, payments, and regulatory compliance. Each stakeholder has specific needs that the system must support to ensure efficiency, security, and reliability.

These stakeholders include:

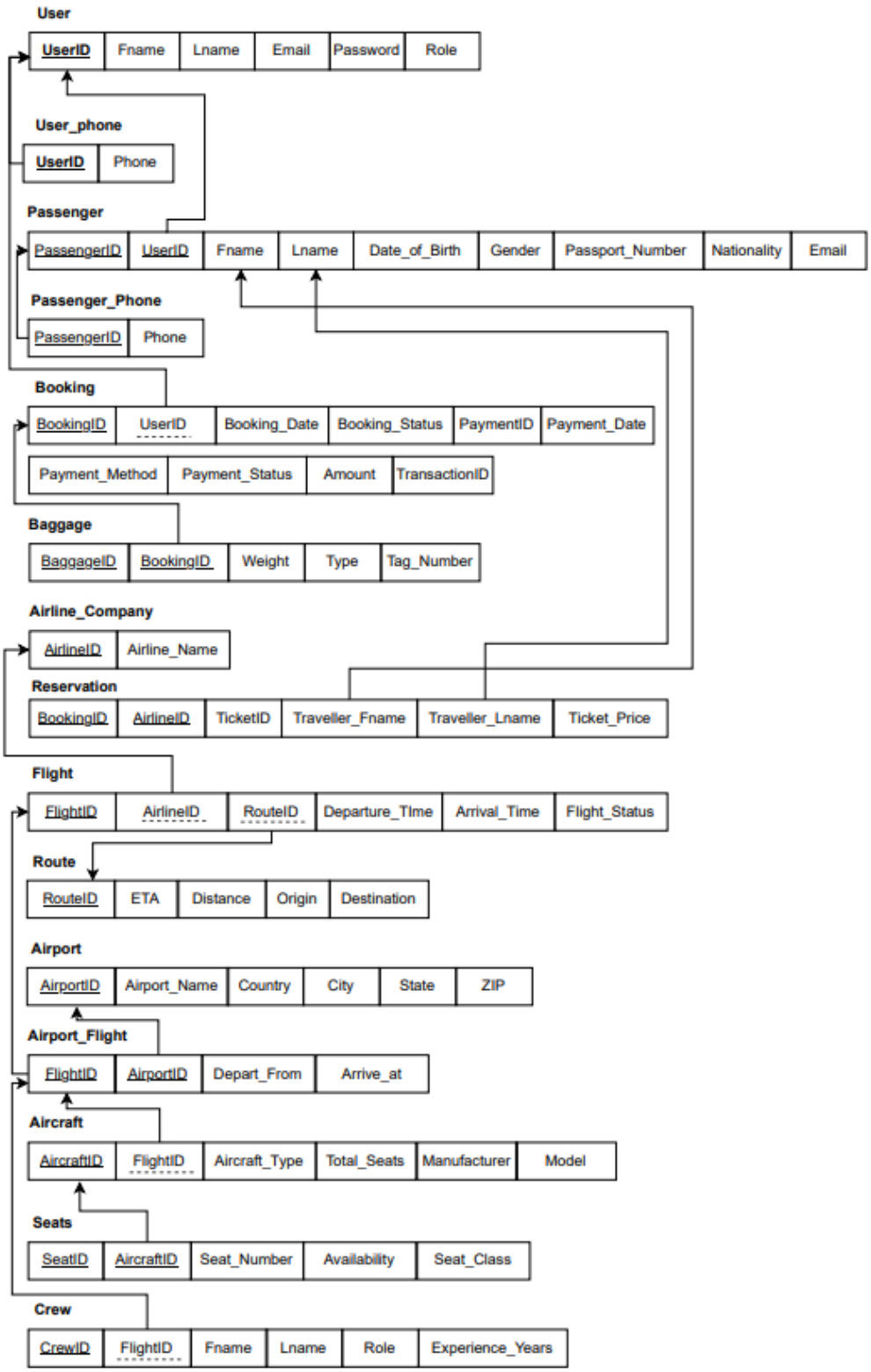
- Travelers who use the system to search, book, and manage flights.
- Airline staff and administrators responsible for managing flights, schedules, and reservations.

- Airline companies that operate flights and manage airplane fleets.
- System administrators who maintain system performance and security.
- Regulatory authorities that ensure compliance with aviation and payment standards.
- Payment gateway providers that process secure financial transactions.
- Travel agents who book flights on behalf of customers.
- Developers and vendors involved in building and maintaining the system.
- Airport authorities that coordinate flight operations and passenger handling.

ER Diagram



ER Mapping


User
UserID
Fname
Lname
Email
Password
Role
User_phone
UserID
Phone
Passenger
PassengerID
UserID
Fname
Lname
Date_of_Birth
Gender
Passport_Number
Nationality
Email
Passenger_Phone
PassengerID
Phone
Booking
BookingID
UserID
Booking_Date
Booking_Status
PaymentID
Payment_Date
Payment_Method
Payment_Status
Amount
TransactionID
Baggage
BaggageID
BookingID
Weight
Type
Tag_Number
Airline_Company
AirlineID
Airline_Name
Reservation
BookingID
AirlineID
TicketID
Traveller_Fname
Traveller_Lname
Ticket_Price
Flight
FlightID
AirlineID
RouteID
Departure_TIme
Arrival_Time
Flight_Status
Route
RouteID
ETA
Distance
Origin
Destination
Airport
AirportID
Airport_Name
Country
City
State
ZIP
Airport_Flight
FlightID
AirportID
Depart_From
Arrive_at
Aircraft
AircraftID
FlightID
Aircraft_Type
Total_Seats
Manufacturer
Model
Seats
SeatID
AircraftID
Seat_Number
Availability
Seat_Class
Crew
CrewID
FlightID
Fname
Lname
Role
Experience_Years

1. Entities and Relationships

1.1 User

SQL:

```sql
CREATE TABLE `User` (
UserID INT AUTO_INCREMENT PRIMARY KEY,
Fname VARCHAR(50) NOT NULL,
Lname VARCHAR(50) NOT NULL,
Email VARCHAR(100) NOT NULL UNIQUE,
Password VARCHAR(255) NOT NULL,
Phone VARCHAR(20) NULL,
Role VARCHAR(50) NOT NULL DEFAULT 'User',
Date_of_Birth DATE NULL,
Last_Login DATETIME NULL,
CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
UpdatedAt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
UserID	INT	PK, AUTO_INCREMENT	Unique identifier for system users.
Fname	VARCHAR(50)	NOT NULL	User's first name.
Lname	VARCHAR(50)	NOT NULL	User's last name.
Email	VARCHAR(100)	NOT NULL, UNIQUE	User's email address for login.
Password	VARCHAR(255)	NOT NULL	Encrypted password for authentication.
Phone	VARCHAR(20)	NULL	User's contact phone number.
Role	VARCHAR(50)	NOT NULL, DEFAULT 'User'	User role (Admin, User, etc.).

Date_of_Birth	DATE	NULL	User's date of birth.
Last_Login	DATETIME	NULL	Timestamp of last login.
CreatedAt	DATETIME	DEFAULT CURRENT_TIMESTAMP	Account creation timestamp.
UpdatedAt	DATETIME	DEFAULT CURRENT_TIMESTAMP ON UPDATE	Last update timestamp.

Purpose: Manages system user accounts for authentication and authorization.

1.2 Passenger

SQL:

```
sql
CREATE TABLE Passenger (
PassengerID INT AUTO_INCREMENT PRIMARY KEY,
Fname VARCHAR(50) NOT NULL,
Lname VARCHAR(50) NOT NULL,
Email VARCHAR(100) NULL,
Phone VARCHAR(20) NULL,
Date_of_Birth DATE NOT NULL,
Nationality VARCHAR(50) NOT NULL,
Passport_Number VARCHAR(50) NOT NULL UNIQUE,
Gender CHAR(1) NOT NULL,
CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
PassengerID	INT	PK, AUTO_INCREMENT	Unique passenger identifier.
Fname	VARCHAR(50)	NOT NULL	Passenger's first name.

Lname	VARCHAR(50)	NOT NULL	Passenger's last name.
Email	VARCHAR(100)	NULL	Passenger's email address.
Phone	VARCHAR(20)	NULL	Contact phone number.
Date_of_Birth	DATE	NOT NULL	Passenger's birth date.
Nationality	VARCHAR(50)	NOT NULL	Passenger's nationality.
Passport_Number	VARCHAR(50)	NOT NULL, UNIQUE	International travel document number.
Gender	CHAR(1)	NOT NULL	Gender (M/F/O).
CreatedAt	DATETIME	DEFAULT CURRENT_TIMESTAMP	Record creation timestamp.

Relationships:

- Makes multiple Bookings (1:M)

1.3 Airport

SQL:

```
sql
CREATE TABLE Airport (
AirportID INT AUTO_INCREMENT PRIMARY KEY,
Airport_Name VARCHAR(100) NOT NULL,
City VARCHAR(100) NOT NULL,
Country VARCHAR(100) NOT NULL,
IATA_Code CHAR(3) NOT NULL UNIQUE,
ICAO_Code CHAR(4) NULL UNIQUE,
Timezone VARCHAR(50) NULL
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
AirportID	INT	PK, AUTO_INCREMENT	Unique airport identifier.
Airport_Name	VARCHAR(100)	NOT NULL	Official airport name.
City	VARCHAR(100)	NOT NULL	City where airport is located.
Country	VARCHAR(100)	NOT NULL	Country of operation.
IATA_Code	CHAR(3)	NOT NULL, UNIQUE	3-letter IATA airport code (e.g., CAI).
ICAO_Code	CHAR(4)	NULL, UNIQUE	4-letter ICAO airport code (e.g., HECA).
Timezone	VARCHAR(50)	NULL	Airport timezone.

Sample Data:

- Cairo International Airport (CAI, HECA)
- Dubai International Airport (DXB, OMDB)
- London Heathrow (LHR, EGLL)
- New York JFK (JFK, KJFK)

Relationships:

- Origin for multiple Routes (1:M)
- Destination for multiple Routes (1:M)

1.4 Route

SQL:

sql

```
CREATE TABLE Route (
RouteID INT AUTO_INCREMENT PRIMARY KEY,
Origin_AirportID INT NOT NULL,
```

```
Destination_AirportID INT NOT NULL,
Distance DECIMAL(10, 2) NULL,
Estimated_Time TIME NULL,

FOREIGN KEY (Origin_AirportID) REFERENCES Airport(AirportID),
FOREIGN KEY (Destination_AirportID) REFERENCES Airport(AirportID)
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
RouteID	INT	PK, AUTO_INCREMENT	Unique route identifier.
Origin_AirportID	INT	NOT NULL, FK	Reference to origin airport.
Destination_AirportID	INT	NOT NULL, FK	Reference to destination airport.
Distance	DECIMAL(10, 2)	NULL	Distance in kilometers.
Estimated_Time	TIME	NULL	Estimated flight duration.

Sample Routes:

- Alexandria → Jeddah (1200 km, 2:15)
- Cairo → Dubai (2500 km, 3:30)
- Cairo → London (3500 km, 5:00)
- Cairo → New York (9000 km, 11:00)

Relationships:

- Connected to Origin Airport (M:1)
- Connected to Destination Airport (M:1)
- Used by multiple Flights (1:M)

1.5 Aircraft

SQL:

```sql
CREATE TABLE Aircraft (
AircraftID INT AUTO_INCREMENT PRIMARY KEY,
Model VARCHAR(100) NOT NULL,
Manufacturer VARCHAR(100) NOT NULL,
Total_Seats INT NOT NULL,
Year_Manufactured INT NULL,
Aircraft_Type VARCHAR(50) NOT NULL,
Registration_Number VARCHAR(20) NOT NULL UNIQUE,
Status VARCHAR(20) DEFAULT 'Active'
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
AircraftID	INT	PK, AUTO_INCREMENT	Unique aircraft identifier.
Model	VARCHAR(100)	NOT NULL	Aircraft model (e.g., Boeing 737-800).
Manufacturer	VARCHAR(100)	NOT NULL	Aircraft manufacturer name.
Total_Seats	INT	NOT NULL	Maximum passenger capacity.
Year_Manufactured	INT	NULL	Year of manufacture.
Aircraft_Type	VARCHAR(50)	NOT NULL	Type (Wide-body, Narrow-body).
Registration_Number	VARCHAR(20)	NOT NULL, UNIQUE	Aircraft registration (e.g., SU-AAA).
Status	VARCHAR(20)	DEFAULT 'Active'	Current status (Active, Maintenance).

Sample Fleet:

- Boeing 737-800 (189 seats, Narrow-body, SU-AAA)
- Airbus A380 (525 seats, Wide-body, SU-DDD)

- Boeing 787-9 Dreamliner (296 seats, Wide-body, SU-EEE)

Relationships:

- Has multiple Seats (1:M)
- Assigned to multiple Flights (1:M)

1.6 Seat

SQL:

```
sql
CREATE TABLE Seat (
SeatID INT AUTO_INCREMENT PRIMARY KEY,
AircraftID INT NOT NULL,
Seat_Number VARCHAR(10) NOT NULL,
Seat_Class VARCHAR(30) NOT NULL,

FOREIGN KEY (AircraftID) REFERENCES Aircraft(AircraftID) ON DELETE
CASCADE,
UNIQUE KEY (AircraftID, Seat_Number)

);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
SeatID	INT	PK, AUTO_INCREMENT	Unique seat identifier.
AircraftID	INT	NOT NULL, FK	Reference to aircraft.
Seat_Number	VARCHAR(10)	NOT NULL	Seat number (e.g., 1A, 12F).
Seat_Class	VARCHAR(30)	NOT NULL	Class (First Class, Business, Premium Economy, Economy).

Seat Classes:

- First Class

- Business
- Premium Economy
- Economy

Relationships:

- Belongs to one Aircraft (M:1)
- Available on multiple Flights via Flight_Seat (1:M)
- Assigned in Bookings via Booking_Seat (1:M)

1.7 Flight_Status

SQL:

```
sql
CREATE TABLE Flight_Status (
FlightStatusID INT AUTO_INCREMENT PRIMARY KEY,
Status_Name VARCHAR(50) NOT NULL UNIQUE,
Description VARCHAR(255) NULL
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
FlightStatusID	INT	PK, AUTO_INCREMENT	Unique status identifier.
Status_Name	VARCHAR(50)	NOT NULL, UNIQUE	Status name.
Description	VARCHAR(255)	NULL	Status description.

Available Statuses:

- Scheduled: Flight is scheduled and on time
- Delayed: Flight is delayed
- Boarding: Passengers are boarding
- Departed: Flight has departed
- Cancelled: Flight has been cancelled

Relationships:

- Assigned to multiple Flights (1:M)

1.8 Flight

SQL:

```
sql
CREATE TABLE Flight (
FlightID INT AUTO_INCREMENT PRIMARY KEY,
Flight_Number VARCHAR(20) NOT NULL,
RouteID INT NOT NULL,
AircraftID INT NOT NULL,
Departure_Time DATETIME NOT NULL,
Arrival_Time DATETIME NOT NULL,
Flight_Date DATE NOT NULL,
FlightStatusID INT NOT NULL,
Gate_Number VARCHAR(10) NULL,

FOREIGN KEY (RouteID) REFERENCES Route(RouteID),
FOREIGN KEY (AircraftID) REFERENCES Aircraft(AircraftID),
FOREIGN KEY (FlightStatusID) REFERENCES Flight_Status(FlightStatusID),
UNIQUE KEY (Flight_Number, Flight_Date)
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
FlightID	INT	PK, AUTO_INCREMENT	Unique flight identifier.
Flight_Number	VARCHAR(20)	NOT NULL	Flight number (e.g., MS100).
RouteID	INT	NOT NULL, FK	Reference to route.
AircraftID	INT	NOT NULL, FK	Reference to aircraft.
Departure_Time	DATETIME	NOT NULL	Scheduled departure time.
Arrival_Time	DATETIME	NOT NULL	Scheduled arrival time.
Flight_Date	DATE	NOT NULL	Flight date.

FlightStatusID	INT	NOT NULL, FK	Reference to flight status.
Gate_Number	VARCHAR(10)	NULL	Departure gate number.

Sample Flights:

- MS100: Alexandria → Jeddah (2025-12-26)
- MS200: Cairo → Dubai (2025-12-27)
- MS300: Cairo → London (2025-12-28)
- MS400: Cairo → New York (2025-12-29)

Relationships:

- Uses one Route (M:1)
- Uses one Aircraft (M:1)
- Has one Flight_Status (M:1)
- Has multiple Bookings (1:M)
- Has multiple Flight_Seats (1:M)

1.9 Ticket_Type

SQL:

```
sql
CREATE TABLE Ticket_Type (
TicketTypeID INT AUTO_INCREMENT PRIMARY KEY,
Type_Name VARCHAR(50) NOT NULL UNIQUE,
Base_Fare DECIMAL(10, 2) NOT NULL,
Description VARCHAR(255) NULL
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
TicketTypeID	INT	PK, AUTO_INCREMENT	Unique ticket type identifier.
Type_Name	VARCHAR(50)	NOT NULL, UNIQUE	Ticket type name.

Base_Fare	DECIMAL(10,2)	NOT NULL	Base fare amount.
Description	VARCHAR(255)	NULL	Type description.

Available Types:

- Economy: $350.00 - Standard economy class ticket
- Premium Economy: $550.00 - Enhanced economy with extra legroom
- Business: $1,200.00 - Business class with premium service
- First Class: $2,500.00 - First class with luxury amenities

Relationships:

- Used in multiple Bookings (1:M)

1.10 Booking

SQL:

```
sql
CREATE TABLE Booking (
BookingID INT AUTO_INCREMENT PRIMARY KEY,
PassengerID INT NOT NULL,
FlightID INT NOT NULL,
Booking_Date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
Booking_Status VARCHAR(20) NOT NULL DEFAULT 'Pending',
Total_Amount DECIMAL(10, 2) NOT NULL,
TicketTypeID INT NOT NULL,

FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID),
FOREIGN KEY (FlightID) REFERENCES Flight(FlightID),
FOREIGN KEY (TicketTypeID) REFERENCES Ticket_Type(TicketTypeID)

);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
BookingID	INT	PK, AUTO_INCREMENT	Unique booking identifier.

PassengerID	INT	NOT NULL, FK	Reference to passenger.
FlightID	INT	NOT NULL, FK	Reference to flight.
Booking_Date	DATETIME	NOT NULL, DEFAULT NOW	Booking creation timestamp.
Booking_Status	VARCHAR(20)	NOT NULL, DEFAULT 'Pending'	Status (Pending, Confirmed, Cancelled).
Total_Amount	DECIMAL(10, 2)	NOT NULL	Total booking amount.
TicketTypeID	INT	NOT NULL, FK	Reference to ticket type.

Relationships:

- Made by one Passenger (M:1)
- For one Flight (M:1)
- Has one Ticket_Type (M:1)
- Has one Payment (1:1)
- Has multiple Booking_Seats (1:M)

1.11 Payment_Method

SQL:

```
sql
CREATE TABLE Payment_Method (
PaymentMethodID INT AUTO_INCREMENT PRIMARY KEY,
Method_Name VARCHAR(50) NOT NULL UNIQUE,
Description VARCHAR(255) NULL
);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
PaymentMethodID	INT	PK, AUTO_INCREMENT	Unique payment method identifier.

Method_Name	VARCHAR(50)	NOT NULL, UNIQUE	Payment method name.
Description	VARCHAR(255)	NULL	Method description.

Available Methods:

- Credit Card
- Debit Card
- PayPal
- Bank Transfer
- Cash

Relationships:

- Used in multiple Payments (1:M)

1.12 Payment

SQL:

```
sql
CREATE TABLE Payment (
PaymentID INT AUTO_INCREMENT PRIMARY KEY,
BookingID INT NOT NULL,
Payment_Date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
Amount DECIMAL(10, 2) NOT NULL,
PaymentMethodID INT NOT NULL,
TransactionID VARCHAR(100) NULL UNIQUE,
Payment_Status VARCHAR(20) NOT NULL DEFAULT 'Pending',

FOREIGN KEY (BookingID) REFERENCES Booking(BookingID),
FOREIGN KEY (PaymentMethodID) REFERENCES
Payment_Method(PaymentMethodID)

);
```

Attributes:

Attribute Name	Data Type	Constraints	Description

PaymentID	INT	PK, AUTO_INCREMENT	Unique payment identifier.
BookingID	INT	NOT NULL, FK	Reference to booking.
Payment_Date	DATETIME	NOT NULL, DEFAULT NOW	Payment timestamp.
Amount	DECIMAL(10,2)	NOT NULL	Payment amount.
PaymentMethodID	INT	NOT NULL, FK	Reference to payment method.
TransactionID	VARCHAR(100)	NULL, UNIQUE	Transaction reference number.
Payment_Status	VARCHAR(20)	NOT NULL, DEFAULT 'Pending'	Status (Pending, Completed, Failed).

Relationships:

- For one Booking (M:1)
- Uses one Payment_Method (M:1)

1.13 Flight_Seat

SQL:

```
sql
CREATE TABLE Flight_Seat (
FlightSeatID INT AUTO_INCREMENT PRIMARY KEY,
FlightID INT NOT NULL,
SeatID INT NOT NULL,
Is_Booked BOOLEAN NOT NULL DEFAULT 0,

FOREIGN KEY (FlightID) REFERENCES Flight(FlightID) ON DELETE CASCADE,
FOREIGN KEY (SeatID) REFERENCES Seat(SeatID),
UNIQUE KEY (FlightID, SeatID)

);
```

Attributes:

Attribute Name	Data Type	Constraints	Description
FlightSeatID	INT	PK, AUTO_INCREMENT	Unique flight seat identifier.
FlightID	INT	NOT NULL, FK	Reference to flight.
SeatID	INT	NOT NULL, FK	Reference to seat.
Is_Booked	BOOLEAN	NOT NULL, DEFAULT 0	Booking status (0=Available, 1=Booked).

Purpose: Links seats to specific flights and tracks their availability status.

Relationships:

- Belongs to one Flight (M:1)
- References one Seat (M:1)

1.14 Booking_Seat

SQL:

```
sql
CREATE TABLE Booking_Seat (
BookingSeatID INT AUTO_INCREMENT PRIMARY KEY,
BookingID INT NOT NULL,
SeatID INT NOT NULL,

FOREIGN KEY (BookingID) REFERENCES Booking(BookingID) ON DELETE
CASCADE,
FOREIGN KEY (SeatID) REFERENCES Seat(SeatID),
UNIQUE KEY (BookingID, SeatID)

);
```

Attributes:

Attribute Name	Data Type	Constraints	Description

BookingSeatID	INT	PK, AUTO_INCREMENT	Unique booking seat identifier.
BookingID	INT	NOT NULL, FK	Reference to booking.
SeatID	INT	NOT NULL, FK	Reference to seat.

Purpose: Links specific seats to passenger bookings.

Relationships:

- Belongs to one Booking (M:1)
- References one Seat (M:1)

2. Relationship Details

2.1 User → Passenger (Register)

- **Cardinality:** 1:M (One-to-Many)
- **Relationship Name:** Register
- **Description:** One user can register multiple passengers in the system. This allows a single user account to manage bookings for family members or other travelers. The relationship enables users to create and maintain passenger profiles for booking purposes.

2.2 Passenger → Booking (Makes)

- **Cardinality:** 1:M (One-to-Many)
- **Relationship Name:** Makes
- **Foreign Key:** Booking.PassengerID → Passenger.PassengerID
- **Description:** One passenger can make multiple bookings for different flights over time. This relationship tracks the complete travel history of each passenger, including all their flight reservations.

2.3 Booking → Payment (HasPayment)

- **Cardinality:** 1:1 (One-to-One)

- **Relationship Name:** HasPayment
- **Foreign Key:** Payment.BookingID → Booking.BookingID
- **Description:** Each booking has exactly one payment transaction associated with it. This ensures that every flight reservation is properly paid for, and payment details are tracked for each booking.

2.4 Booking → Flight (Makes)

- **Cardinality:** M:1 (Many-to-One)
- **Relationship Name:** Makes
- **Foreign Key:** Booking.FlightID → Flight.FlightID
- **Description:** Multiple bookings can be made for the same flight by different passengers. Each flight can accommodate many passengers up to its seat capacity.

2.5 Airline_Company → Flight (Operates)

- **Cardinality:** 1:M (One-to-Many)
- **Relationship Name:** Operates
- **Description:** One airline company operates multiple flights. This relationship establishes which airline is responsible for each flight service.

2.6 Flight → Route (Schedule)

- **Cardinality:** M:1 (Many-to-One)
- **Relationship Name:** Schedule
- **Foreign Key:** Flight.RouteID → Route.RouteID
- **Description:** Multiple flights can follow the same route on different dates and times. For example, the Cairo to Dubai route may have several flights per day, each represented as a separate flight record.

2.7 Flight → Aircraft (Operates)

- **Cardinality:** M:1 (Many-to-One)
- **Relationship Name:** Operates
- **Foreign Key:** Flight.AircraftID → Aircraft.AircraftID
- **Description:** Multiple flights can use the same aircraft over different time periods. One aircraft serves many flights throughout its operational schedule.

2.8 Aircraft → Seat (HasSeats)

- **Cardinality:** 1:M (One-to-Many)
- **Relationship Name:** HasSeats
- **Foreign Key:** Seat.AircraftID → Aircraft.AircraftID

- **Description:** Each aircraft has a fixed configuration of multiple seats. The seat arrangement includes different classes (First Class, Business, Premium Economy, Economy) with specific seat numbers (e.g., 1A, 12F, 25C).

2.9 Aircraft → Crew (Works)

- **Cardinality:** 1:M (One-to-Many)
- **Relationship Name:** Works
- **Description:** Multiple crew members work on one aircraft. This relationship assigns flight crew (pilots, flight attendants, engineers) to specific aircraft.

2.10 Booking ↔ Seat (via Booking_Seat)

- **Cardinality:** M:N (Many-to-Many)
- **Junction Table:** Booking_Seat
- **Foreign Keys:**
    - Booking_Seat.BookingID → Booking.BookingID
    - Booking_Seat.SeatID → Seat.SeatID

Query 1: Basic Flight Search

Description

Find all flights from a specific origin to destination on a given date

Complexity: Simple - Basic JOIN
Tables: Flight, Route, Airport (2x), Flight_Status, Aircraft

SQL Code

```
sql
SELECT
    f.Flight_Number,
    orig.Airport_Name AS Origin_Airport,
    orig.IATA_Code AS Origin_Code,
    dest.Airport_Name AS Destination_Airport,
    dest.IATA_Code AS Destination_Code,
    f.Departure_Time,
    f.Arrival_Time,
    fs.Status_Name AS Flight_Status,
    a.Model AS Aircraft_Model,
    a.Total_Seats
FROM Flight f
INNER JOIN `Route` r ON f.RouteID = r.RouteID
INNER JOIN Airport orig ON r.Origin_AirportID = orig.AirportID
INNER JOIN Airport dest ON r.Destination_AirportID = dest.AirportID
INNER JOIN Flight_Status fs ON f.FlightStatusID = fs.FlightStatusID
INNER JOIN Aircraft a ON f.AircraftID = a.AircraftID
ORDER BY f.Departure_Time;
```

Relational Algebra

```
π Flight_Number, orig.Airport_Name→Origin_Airport, orig.IATA_Code→Origin_Code,
  dest.Airport_Name→Destination_Airport, dest.IATA_Code→Destination_Code,
  Departure_Time, Arrival_Time, Status_Name→Flight_Status,
  Model→Aircraft_Model, Total_Seats
(
  σ orig.IATA_Code='JFK' ∧ dest.IATA_Code='LAX' ∧ Flight_Date='2025-12-10'
  (
    Flight ⋈ Flight.RouteID=Route.RouteID Route
          ⋈ Route.Origin_AirportID=Airport.AirportID (ρ orig (Airport))
          ⋈ Route.Destination_AirportID=Airport.AirportID (ρ dest (Airport))
          ⋈ Flight.FlightStatusID=Flight_Status.FlightStatusID Flight_Status
          ⋈ Flight.AircraftID=Aircraft.AircraftID Aircraft
```

)

)

Flight_Number	Origin_Airport	Origin_Code	Destination_Airport	Destination_Code	Flight_Date	Departure_Time	Arrival_Time	Flight_Status
MS100	Alexandria Borg El Arab Airport	HBE	Jeddah King Abdulaziz Airport	JED	2025-12-26	2025-12-26 08:00:00	2025-12-26 10:15:00	Scheduled
MS200	Cairo International Airport	CAI	Dubai International Airport	DXB	2025-12-27	2025-12-27 14:00:00	2025-12-27 17:30:00	Scheduled
MS300	Cairo International Airport	CAI	London Heathrow	LHR	2025-12-28	2025-12-28 10:00:00	2025-12-28 15:00:00	Scheduled
MS400	Cairo International Airport	CAI	New York JFK	JFK	2025-12-29	2025-12-28 20:00:00	2025-12-29 07:00:00	Scheduled
MS150	Dubai International Airport	DXB	London Heathrow	LHR	2025-12-30	2025-12-30 07:00:00	2025-12-30 07:45:00	Scheduled
MS151	Jeddah King Abdulaziz Airport	JED	Alexandria Borg El Arab Airport	HBE	2025-12-31	2025-12-31 18:00:00	2025-12-31 18:45:00	Scheduled
MS503	Cairo International Airport	CAI	Alexandria Borg El Arab Airport	HBE	2026-01-01	2026-01-01 08:00:00	2026-01-01 10:15:00	Scheduled
MS451	Paris Charles de Gaulle	CDG	Cairo International Airport	CAI	2026-01-02	2026-01-02 19:00:00	2026-01-02 23:30:00	Scheduled
MS550	London Heathrow	LHR	New York JFK	JFK	2026-01-03	2026-01-03 10:00:00	2026-01-03 18:00:00	Scheduled
MS551	New York JFK	JFK	London Heathrow	LHR	2026-01-04	2026-01-03 21:00:00	2025-01-04 05:00:00	Scheduled
MS350	Jeddah King Abdulaziz Airport	JED	Dubai International Airport	DXB	2026-01-05	2026-01-05 10:00:00	2026-01-05 12:00:00	Scheduled
MS351	Dubai International Airport	DXB	Jeddah King Abdulaziz Airport	JED	2026-01-06	2026-01-06 16:00:00	2026-01-06 18:00:00	Scheduled

---

Query 2: Revenue Analysis by Route

Description

Analyze revenue, bookings, and average fare by route

Complexity: Medium - Multiple JOINs with GROUP BY and aggregation
Tables: Booking, Booking_Seat, Flight_Seat, Flight, Route, Airport (2x)

SQL Code

sql

```
SELECT
    orig.City AS Origin_City,
    orig.IATA_Code AS Origin_Code,
    dest.City AS Destination_City,
    dest.IATA_Code AS Destination_Code,
    COUNT(DISTINCT b.BookingID) AS Total_Bookings,
    SUM(b.Total_Amount) AS Total_Revenue,
    AVG(b.Total_Amount) AS Average_Fare,
    MIN(b.Total_Amount) AS Min_Fare,
    MAX(b.Total_Amount) AS Max_Fare,
    COUNT(DISTINCT f.FlightID) AS Flights_On_Route
FROM Booking b
INNER JOIN Booking_Seat bs ON b.BookingID = bs.BookingID
INNER JOIN Flight_Seat fls ON bs.SeatID = fls.SeatID
INNER JOIN Flight f ON fls.FlightID = f.FlightID
```

```
INNER JOIN `Route` r ON f.RouteID = r.RouteID
INNER JOIN Airport orig ON r.Origin_AirportID = orig.AirportID
INNER JOIN Airport dest ON r.Destination_AirportID = dest.AirportID
WHERE b.Booking_Status = 'Confirmed'
GROUP BY orig.City, orig.IATA_Code, dest.City, dest.IATA_Code
ORDER BY Total_Revenue DESC;
```

Relational Algebra

τ Total_Revenue DESC
(
  γ orig.City → Origin_City, orig.IATA_Code → Origin_Code,
    dest.City → Destination_City, dest.IATA_Code → Destination_Code;
    COUNT(DISTINCT BookingID) → Total_Bookings,
    SUM(Total_Amount) → Total_Revenue,
    AVG(Total_Amount) → Average_Fare,
    MIN(Total_Amount) → Min_Fare,
    MAX(Total_Amount) → Max_Fare,
    COUNT(DISTINCT FlightID) → Flights_On_Route
  (
    σ Booking_Status='Confirmed'
    (
      Booking ⋈ Booking.BookingID=Booking_Seat.BookingID Booking_Seat
            ⋈ Booking_Seat.SeatID=Flight_Seat.SeatID Flight_Seat
            ⋈ Flight_Seat.FlightID=Flight.FlightID Flight
            ⋈ Flight.RouteID=Route.RouteID Route
            ⋈ Route.Origin_AirportID=Airport.AirportID (ρ orig (Airport))
            ⋈ Route.Destination_AirportID=Airport.AirportID (ρ dest (Airport))
)
)
)

Origin_City	Origin_Code	Destination_City	Destination_Code	Total_Bookings	Total_Revenue	Average_Fare	Min_Fare	Max_Fare	Flights_On_Route
Alexandria	HBE	Jeddah	JED	1	350.00	350.000000	350.00	350.00	1
Paris	CDG	Cairo	CAI	1	350.00	350.000000	350.00	350.00	1

Query 3: Top 10 Passengers by Total Spending

Description

Identify VIP passengers based on total confirmed bookings

Complexity: Medium - Aggregation with ranking
Tables: Passenger, Booking

SQL Code

```
sql
SELECT
    p.PassengerID,
    CONCAT(p.Fname, ' ', p.Lname) AS Passenger_Name,
    p.Email,
    p.Nationality,
    COUNT(b.BookingID) AS Total_Bookings,
    SUM(b.Total_Amount) AS Total_Spent,
    AVG(b.Total_Amount) AS Avg_Booking_Value,
    MAX(b.Booking_Date) AS Last_Booking_Date
FROM Passenger p
INNER JOIN Booking b ON p.PassengerID = b.PassengerID
WHERE b.Booking_Status = 'Confirmed'
GROUP BY p.PassengerID, p.Fname, p.Lname, p.Email, p.Nationality
ORDER BY Total_Spent DESC
LIMIT 10;
```

Relational Algebra

```
LIMIT 10
(
  τ Total_Spent DESC
  (
    γ PassengerID, Fname, Lname, Email, Nationality;
      COUNT(BookingID)→Total_Bookings,
      SUM(Total_Amount)→Total_Spent,
      AVG(Total_Amount)→Avg_Booking_Value,
      MAX(Booking_Date)→Last_Booking_Date
    (
      σ Booking_Status='Confirmed'
      (
        Passenger ⋈ Passenger.PassengerID=Booking.PassengerID Booking
      )
    )
  )
)
```

PassengerID	Passenger_Name	Email	Nationality	BookingID	Booking_Date	Total_Amount	Flight_Number	Flight_Date	Departure_Time	Origin	Destination	Seat_Number	Seat_Class
2	Nasser Hossam	NasserHossam@gmail.com	Egyptian	1	2025-12-24 01:30:03	350.00	MS100	2025-12-26	2025-12-26 08:00:00	Alexandria	Jeddah	10A	Economy
4	Hoss Hoss	hoss@gmail.com	American	2	2025-12-24 01:31:54	350.00	MS451	2026-01-02	2026-01-02 19:00:00	Paris	Cairo	25A	Economy

---

Query 4: Payment Method Analysis

Description

Analyze payment methods used and their totals

Complexity: Simple - GROUP BY with payment details
Tables: Payment, Payment_Method

SQL Code

sql

```
SELECT
    pm.Method_Name,
    COUNT(DISTINCT p.PaymentID) AS Total_Transactions,
    COUNT(DISTINCT p.BookingID) AS Unique_Bookings,
    SUM(p.Amount) AS Total_Amount,
    AVG(p.Amount) AS Average_Transaction,
    MIN(p.Payment_Date) AS First_Transaction,
    MAX(p.Payment_Date) AS Latest_Transaction
FROM Payment p
INNER JOIN Payment_Method pm ON p.PaymentMethodID = pm.PaymentMethodID
WHERE p.Payment_Status = 'Completed'
GROUP BY pm.PaymentMethodID, pm.Method_Name

ORDER BY Total_Amount DESC;
```

Relational Algebra

```
τ Total_Amount DESC
(
  γ PaymentMethodID, Method_Name;
    COUNT(DISTINCT PaymentID)→Total_Transactions,
    COUNT(DISTINCT BookingID)→Unique_Bookings,
    SUM(Amount)→Total_Amount,
    AVG(Amount)→Average_Transaction,
    MIN(Payment_Date)→First_Transaction,
    MAX(Payment_Date)→Latest_Transaction
  (
    σ Payment_Status='Completed'
```

```
    (
      Payment ⋈ Payment.PaymentMethodID=Payment_Method.PaymentMethodID
Payment_Method
    )
  )
)
```

Method_Name	Total_Transactions	Unique_Bookings	Total_Amount	Average_Transaction	First_Transaction	Latest_Transaction
Credit Card	2	2	700.00	350.000000	2025-12-24 01:30:03	2025-12-24 01:31:54

Query 5: Route Distance and Flight Time Analysis

Description

Compare actual flight times with estimated times

Complexity: Medium - Calculated fields with time analysis
Tables: Route, Airport (2x), Flight

SQL Code

sql

```
SELECT
    orig.City AS Origin,
    dest.City AS Destination,
    r.Distance AS Distance_KM,
    r.Estimated_Time,
    COUNT(f.FlightID) AS Number_Of_Flights,
    AVG(TIMESTAMPDIFF(MINUTE, f.Departure_Time, f.Arrival_Time)) AS
Avg_Actual_Duration_Minutes,
    MIN(TIMESTAMPDIFF(MINUTE, f.Departure_Time, f.Arrival_Time)) AS
Min_Duration_Minutes,
    MAX(TIMESTAMPDIFF(MINUTE, f.Departure_Time, f.Arrival_Time)) AS
Max_Duration_Minutes,
    CASE
        WHEN AVG(TIMESTAMPDIFF(MINUTE, f.Departure_Time, f.Arrival_Time)) >
TIMESTAMPDIFF(MINUTE, '00:00:00', r.Estimated_Time)
        THEN 'Behind Schedule'
        ELSE 'On Time'
    END AS Performance_Status
FROM `Route` r
INNER JOIN Airport orig ON r.Origin_AirportID = orig.AirportID
```

```
INNER JOIN Airport dest ON r.Destination_AirportID = dest.AirportID
INNER JOIN Flight f ON r.RouteID = f.RouteID
GROUP BY r.RouteID, orig.City, dest.City, r.Distance, r.Estimated_Time
ORDER BY r.Distance DESC;
```

Relational Algebra

τ Distance DESC
(
  γ RouteID, orig.City→Origin, dest.City→Destination,
    Distance→Distance_KM, Estimated_Time;
    COUNT(FlightID)→Number_Of_Flights,
    AVG(TIMESTAMPDIFF(MINUTE, Departure_Time,
Arrival_Time))→Avg_Actual_Duration_Minutes,
    MIN(TIMESTAMPDIFF(MINUTE, Departure_Time, Arrival_Time))→Min_Duration_Minutes,
    MAX(TIMESTAMPDIFF(MINUTE, Departure_Time, Arrival_Time))→Max_Duration_Minutes,
    CASE (AVG > Estimated_Time) THEN 'Behind Schedule' ELSE 'On
Time'→Performance_Status
  (
    Route ⋈ Route.Origin_AirportID=Airport.AirportID (ρ orig (Airport))
          ⋈ Route.Destination_AirportID=Airport.AirportID (ρ dest (Airport))
          ⋈ Route.RouteID=Flight.RouteID Flight
)
)

Origin	Destination	Distance_KM	Estimated_Time	Number_Of_Flights	Avg_Actual_Duration_Minutes	Min_Duration_Minutes	Max_Duration_Minutes	Performance_Status
Cairo	New York	9000.00	11:00:00	1	660.0000	660	660	On Time
Dubai	London	5500.00	07:00:00	1	45.0000	45	45	On Time
Cairo	London	3500.00	05:00:00	1	300.0000	300	300	On Time
Cairo	Dubai	2500.00	03:30:00	1	210.0000	210	210	On Time
Alexandria	Jeddah	1200.00	02:15:00	1	135.0000	135	135	On Time
Jeddah	Alexandria	1200.00	02:15:00	1	45.0000	45	45	On Time
Cairo	Alexandria	220.00	00:45:00	1	135.0000	135	135	On Time
Jeddah	Dubai	220.00	00:45:00	1	120.0000	120	120	On Time
Dubai	Jeddah	220.00	00:45:00	1	120.0000	120	120	On Time
London	New York	220.00	00:45:00	1	480.0000	480	480	On Time
New York	London	220.00	00:45:00	1	-525120.0000	-525120	-525120	On Time
Paris	Cairo	220.00	00:45:00	1	270.0000	270	270	On Time

Query 6: Flight Occupancy Report

Description

Calculate seat occupancy percentage for each flight

Complexity: Medium - Aggregation with calculated percentages
Tables: Flight, Aircraft, Flight_Seat, Route, Airport (2x)

SQL Code

```
sql
SELECT
    f.FlightID,
    f.Flight_Number,
    f.Flight_Date,
    CONCAT(orig.City, ' → ', dest.City) AS Route,
    a.Model AS Aircraft,
    a.Total_Seats,
    COUNT(CASE WHEN fs.Is_Booked = 1 THEN 1 END) AS Booked_Seats,
    a.Total_Seats - COUNT(CASE WHEN fs.Is_Booked = 1 THEN 1 END) AS Available_Seats,
    CAST(COUNT(CASE WHEN fs.Is_Booked = 1 THEN 1 END) AS DECIMAL(10,2)) /
a.Total_Seats * 100 AS Occupancy_Percentage
FROM Flight f
INNER JOIN Aircraft a ON f.AircraftID = a.AircraftID
INNER JOIN Flight_Seat fs ON f.FlightID = fs.FlightID
INNER JOIN `Route` r ON f.RouteID = r.RouteID
INNER JOIN Airport orig ON r.Origin_AirportID = orig.AirportID
INNER JOIN Airport dest ON r.Destination_AirportID = dest.AirportID
GROUP BY
    f.FlightID, f.Flight_Number, f.Flight_Date,
    orig.City, dest.City, a.Model, a.Total_Seats

ORDER BY Occupancy_Percentage DESC;
```

Relational Algebra

```
τ Occupancy_Percentage DESC
(
  γ FlightID, Flight_Number, Flight_Date,
    CONCAT(orig.City, '→', dest.City)→Route, Model→Aircraft, Total_Seats;
    COUNT(CASE Is_Booked=1)→Booked_Seats,
    (Total_Seats - COUNT(CASE Is_Booked=1))→Available_Seats,
    (COUNT(CASE Is_Booked=1) / Total_Seats × 100)→Occupancy_Percentage
  (
    Flight ⋈ Flight.AircraftID=Aircraft.AircraftID Aircraft
          ⋈ Flight.FlightID=Flight_Seat.FlightID Flight_Seat
          ⋈ Flight.RouteID=Route.RouteID Route
          ⋈ Route.Origin_AirportID=Airport.AirportID (ρ orig (Airport))
          ⋈ Route.Destination_AirportID=Airport.AirportID (ρ dest (Airport))
  )
```

)

FlightID	Flight_Number	Flight_Date	Route	Aircraft	Total_Seats	Booked_Seats	Available_Seats	Occupancy_Percentage
1	MS100	2025-12-26	Alexandria → Jeddah	Boeing 737-800	189	1	188	0.529101
8	MS451	2026-01-02	Paris → Cairo	Airbus A380	525	1	524	0.190476
2	MS200	2025-12-27	Cairo → Dubai	Airbus A320	180	0	180	0.000000
3	MS300	2025-12-28	Cairo → London	Boeing 777-300ER	396	0	396	0.000000
4	MS400	2025-12-29	Cairo → New York	Boeing 777-300ER	396	0	396	0.000000
7	MS503	2026-01-01	Cairo → Alexandria	Boeing 737 MAX 8	178	0	178	0.000000
6	MS151	2025-12-31	Jeddah → Alexandria	Airbus A350-900	325	0	325	0.000000
11	MS350	2026-01-05	Jeddah → Dubai	Boeing 737-800	189	0	189	0.000000
5	MS150	2025-12-30	Dubai → London	Boeing 787-9 Dreamliner	296	0	296	0.000000
12	MS351	2026-01-06	Dubai → Jeddah	Boeing 737-800	189	0	189	0.000000
9	MS550	2026-01-03	London → New York	Boeing 737 MAX 8	178	0	178	0.000000
10	MS551	2026-01-04	New York → London	Airbus A350-900	325	0	325	0.000000

Query 7: Complete Passenger Booking Details

Description

Retrieve comprehensive passenger information with all confirmed booking details, flight information, and seat assignments

Complexity: Complex - Multiple JOINs with extensive filtering
Tables: Passenger, Booking, Booking_Seat, Seat, Flight_Seat, Flight, Route, Airport (2x)

SQL Code

sql

```
SELECT
    p.PassengerID,
    p.Fname,
    p.Lname,
    p.Email,
    p.Phone,
    p.Date_of_Birth,
```

```
    p.Nationality,
    p.Passport_Number,
    p.Gender,
    p.CreatedAt,
    b.BookingID,
    b.Booking_Date,
    b.Total_Amount,
    b.Booking_Status,
    f.Flight_Number,
    f.Flight_Date,
    f.Departure_Time,
    orig.City AS Origin,
    dest.City AS Destination,
    s.Seat_Number,
    s.Seat_Class
FROM Passenger p
INNER JOIN Booking b ON p.PassengerID = b.PassengerID
INNER JOIN Booking_Seat bs ON b.BookingID = bs.BookingID
INNER JOIN Seat s ON bs.SeatID = s.SeatID
INNER JOIN Flight_Seat fs ON b.FlightID = fs.FlightID AND fs.SeatID = s.SeatID
INNER JOIN Flight f ON b.FlightID = f.FlightID
INNER JOIN Route r ON f.RouteID = r.RouteID
INNER JOIN Airport orig ON r.Origin_AirportID = orig.AirportID
INNER JOIN Airport dest ON r.Destination_AirportID = dest.AirportID
WHERE b.Booking_Status = 'Confirmed'
  AND fs.Is_Booked = 1
```

```
  AND p.Fname IS NOT NULL
  AND p.Lname IS NOT NULL
ORDER BY p.PassengerID, b.Booking_Date DESC, f.Flight_Date, s.Seat_Number;
```

Relational Algebra

τ PassengerID, Booking_Date DESC, Flight_Date, Seat_Number

(

  π PassengerID, Fname, Lname, Email, Phone, Date_of_Birth, Nationality,

    Passport_Number, Gender, CreatedAt, BookingID, Booking_Date, Total_Amount,

    Booking_Status, Flight_Number, Flight_Date, Departure_Time,

    orig.City→Origin, dest.City→Destination, Seat_Number, Seat_Class

  (

    σ Booking_Status='Confirmed' ∧ Is_Booked=1 ∧ Fname IS NOT NULL ∧ Lname IS NOT NULL

    (

      Passenger ⋈ Passenger.PassengerID=Booking.PassengerID Booking

            ⋈ Booking.BookingID=Booking_Seat.BookingID Booking_Seat

            ⋈ Booking_Seat.SeatID=Seat.SeatID Seat

            ⋈ Booking.FlightID=Flight_Seat.FlightID ∧ Seat.SeatID=Flight_Seat.SeatID Flight_Seat

            ⋈ Booking.FlightID=Flight.FlightID Flight

            ⋈ Flight.RouteID=Route.RouteID Route

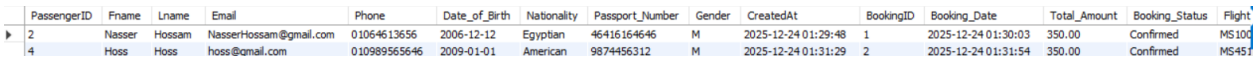
            ⋈ Route.Origin_AirportID=Airport.AirportID (ρ orig (Airport))

            ⋈ Route.Destination_AirportID=Airport.AirportID (ρ dest (Airport))

)

)

)



PassengerID	Fname	Lname	Email	Phone	Date_of_Birth	Nationality	Passport_Number	Gender	CreatedAt	BookingID	Booking_Date	Total_Amount	Booking_Status	Flight
2	Nasser	Hossam	NasserHossam@gmail.com	01064613656	2006-12-12	Egyptian	46416164646	M	2025-12-24 01:29:48	1	2025-12-24 01:30:03	350.00	Confirmed	MS100
4	Hoss	Hoss	hoss@gmail.com	010989565646	2009-01-01	American	9874456312	M	2025-12-24 01:31:29	2	2025-12-24 01:31:54	350.00	Confirmed	MS451

Query 8: Basic Booking and Passenger Information

Description

Simple query to retrieve booking information with associated passenger details

Complexity: Simple - Basic JOIN

Tables: Booking, Passenger

SQL Code

sql

```
SELECT
    b.BookingID,
    b.PassengerID,
    p.Email,
    p.Fname,
    b.Total_Amount
FROM Booking b
INNER JOIN Passenger p ON b.PassengerID = p.PassengerID;
```

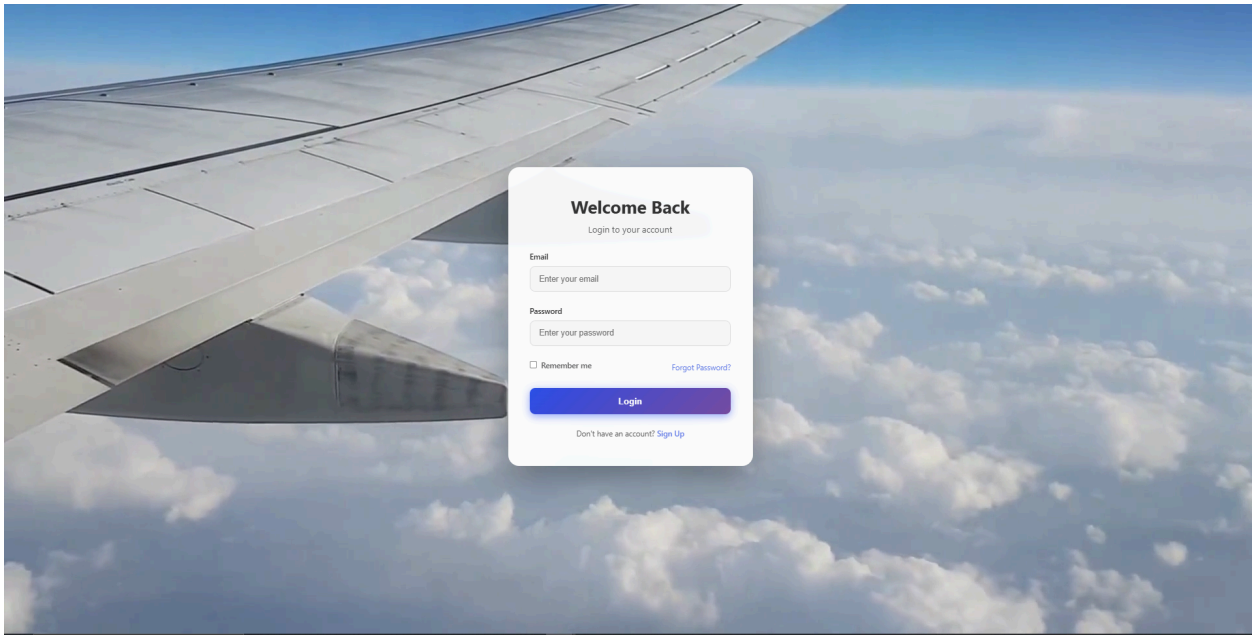
Relational Algebra

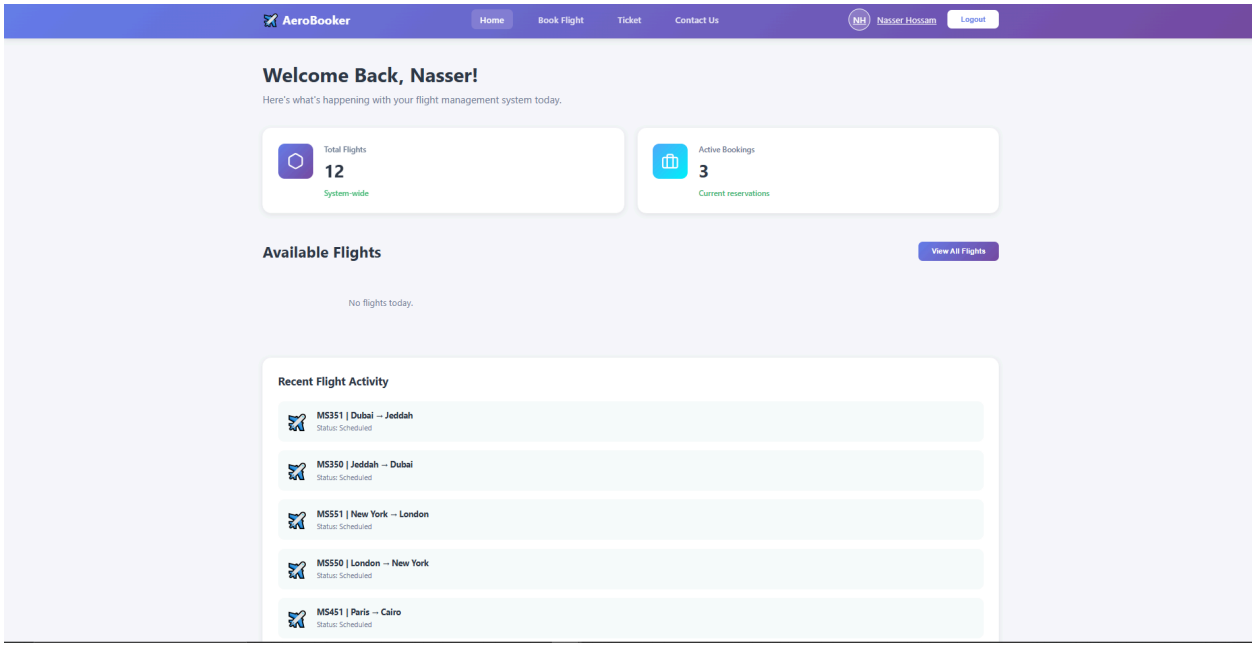
$$\pi \text{ BookingID, PassengerID, Email, Fname, Total\_Amount}$$

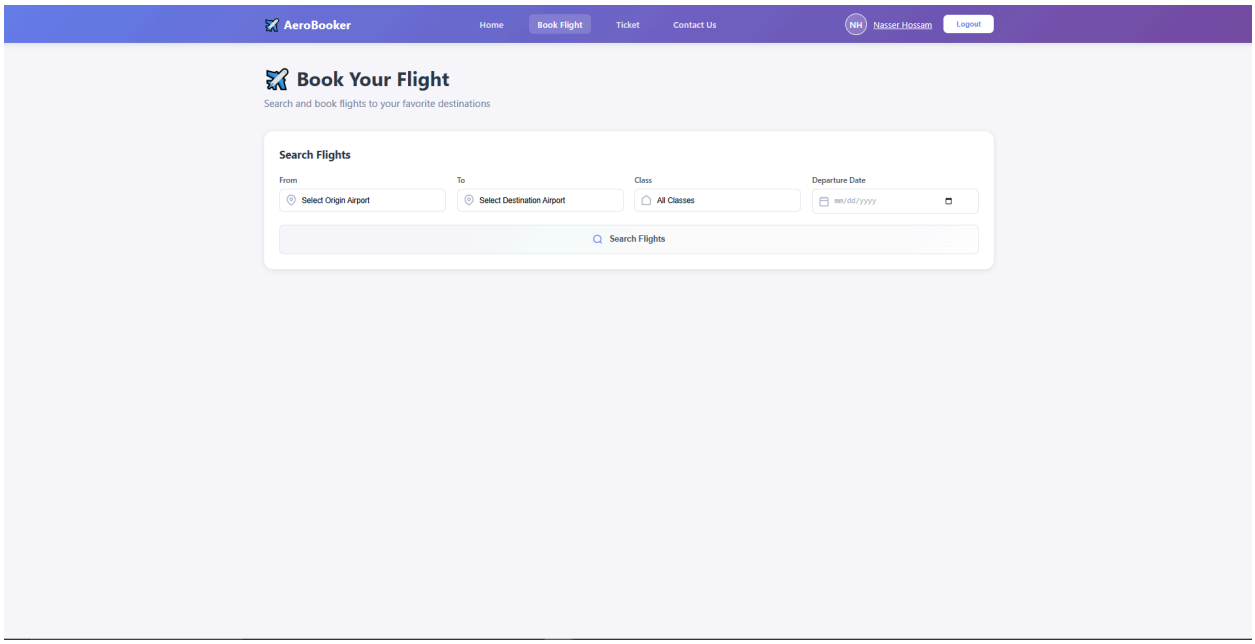
$$(\text{Booking} \bowtie \text{Booking.PassengerID=Passenger.PassengerID Passenger})$$

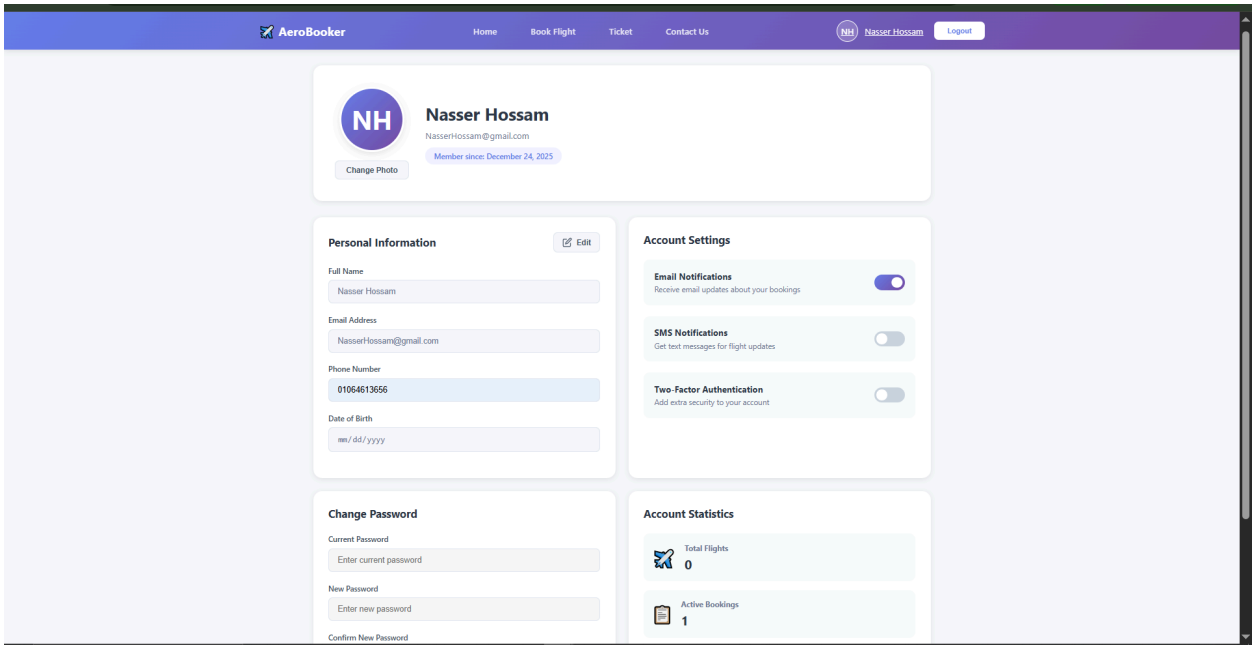
	BookingID	PassengerID	Email	Fname	Total_Amount
▶	1	2	NasserHossam@gmail.com	Nasser	350.00
	2	4	hoss@gmail.com	Hoss	350.00

GUI






AeroBooker
Home
Book Flight
Ticket
Contact Us
NH
Nasser Hossam
Logout
Book Your Flight
Search and book flights to your favorite destinations
Search Flights
From
Select Origin Airport
To
Select Destination Airport
Class
All Classes
Departure Date
mm/dd/yyyy
Search Flights


AeroBooker
Home
Book Flight
Ticket
Contact Us
NH
Nasser Hossam
Logout
NH
Change Photo
Nasser Hossam
NasserHossam@gmail.com
Member since: December 24, 2025
Personal Information
Edit
Full Name
Nasser Hossam
Email Address
NasserHossam@gmail.com
Phone Number
01064613656
Date of Birth
mm/dd/yyyy
Account Settings
Email Notifications
Receive email updates about your bookings
SMS Notifications
Get text messages for flight updates
Two-Factor Authentication
Add extra security to your account
Change Password
Current Password
Enter current password
New Password
Enter new password
Confirm New Password
Account Statistics
Total Flights
0
Active Bookings
1