
# AIRLINE RESERVATION SYSTEM

**Submitted by:**

Name: Karan Dandora

UID:23BCA10315

Class 23BCA 4(B)

**Submitted to:**

Mr. Arvinder Singh

Subject Code: 23CAP-252

# Introduction

In today's digital era, the airline industry relies heavily on robust and efficient database systems to manage the growing demand for air travel services. From flight scheduling and passenger management to real-time ticket booking and secure payment processing, an organized and reliable **Database Management System (DBMS)** is essential.

This project, titled **"Airline Reservation System"**, focuses on designing and implementing a DBMS-based solution that simulates the core operations of an airline ticket booking system. The goal is to demonstrate how real-world airline functions can be efficiently handled through relational database concepts, normalized schemas, and structured SQL queries.

The system is designed with the following core modules:

- **User Management** – with support for both Admins and Passengers (using generalization).
- **Flight Scheduling** – including routes, timings, and seat availability.
- **Booking & Payment Processing** – allowing passengers to reserve and cancel flights.
- **Querying & Data Analysis** – providing insights through SELECT, JOIN, GROUP BY, and aggregate operations.

This case study will walk through every step of the DBMS development process — from requirement gathering and ER modeling to normalization, SQL implementation, and testing. It also showcases how theoretical DBMS principles like **normal forms**, **referential integrity**, and **generalization/specialization** are applied practically in building a real-life database system.

# Technique

The **Airline Reservation System** was designed and developed using a structured and methodical database development process. The techniques used throughout this project combine theoretical DBMS concepts with practical implementation strategies. Below is a breakdown of the techniques applied:

---

## 1. Entity-Relationship (ER/EER) Modeling

- **ER Diagram:** Initial high-level design captured all key entities (User, Flight, Booking, Payment).
- **EER Diagram:** Incorporated **generalization** for roles like *Admin* and *Passenger* derived from a common *User* entity.
- **Tool Used:** Mermaid Live Editor / Draw.io

---

## 2. Normalization

- Database design followed the rules of **1NF, 2NF, and 3NF** to eliminate redundancy and maintain data consistency.
- Attributes were separated into logically distinct tables, and foreign keys were used to maintain relationships.

---

## 3. Relational Database Design

- Converted the ER model into relational schema using **Primary Keys** and **Foreign Keys**.
- Each entity was represented as a table with appropriate data types, constraints, and indexes.

---

## 4. SQL-Based Implementation

- SQL queries were written to:
  - **Create tables** (`CREATE TABLE`)
  - **Insert sample data** (`INSERT INTO`)
  - **Retrieve data** (`SELECT`, `JOIN`, `GROUP BY`, `ORDER BY`)

  - **Modify data** (`UPDATE`, `DELETE`)
- Used **aggregate functions**, **nested subqueries**, and **UNION** operations for analysis.

---

### 5. CRUD Operations

- Implemented full **Create, Read, Update, Delete** functionality using SQL queries to demonstrate real-world use cases.

---

### 6. Validation & Testing

- Queries were tested on test data to ensure logical consistency.
- Verified data integrity using **foreign key constraints** and **NOT NULL checks**.

# System Configuration

This section outlines the hardware and software environment used during the development and execution of the **Airline Reservation System** project.

## Hardware Requirements:

- **Processor:** Intel Core i5 / Ryzen 5 or higher
- **RAM:** 8 GB or higher
- **Storage:** Minimum 500 GB (HDD or SSD recommended)
- **Display:** 13” or larger screen for better workspace management
- **Network:** Internet access for installing tools and testing optional web features

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.
NAAC GRADE A+
ACCREDITED UNIVERSITY

# Acknowledgement

**I would like to express my heartfelt gratitude to my respected guide, Arvinder Singh for their valuable guidance, support, and encouragement throughout the completion of this case study.**

**I am also thankful to the faculty members of the Department of Computer Science, for providing a strong academic foundation that helped in conceptualizing this project.**

**Last but not least, I would like to thank my friends and family for their support and motivation.**

## INPUT

**AIM: The aim of this project is to design and implement a Database Management System (DBMS) for an Airline Reservation System that allows passengers to search for available flights, make bookings, and manage their travel details efficiently and securely. The system will provide a structured relational database model that ensures data consistency, supports efficient query processing, and reduces data redundancy. It will facilitate the airline staff in managing flight schedules, seat availability, passenger records, and booking statuses while maintaining data integrity and accessibility. By applying key DBMS concepts such as relational schema design, normalization, and SQL-based data manipulation, this project aims to demonstrate the practical implementation of a real-world database application.**

**Objectives**: The primary objectives of the Airline Reservation System are:

1. **To design a relational database** that stores and manages airline operations data such as flights, passengers, bookings, payments, and schedules.
2. **To allow users to search for available flights** based on criteria like source, destination, and travel date.
3. **To enable secure ticket booking** and passenger registration within the system.
4. **To manage real-time seat availability** for different flights and prevent overbooking.
5. **To provide administrative functionality** for airline staff to add, update, or delete flight information.
6. **To ensure data consistency and accuracy** using database normalization and integrity constraints.
7. **To implement SQL queries** for performing CRUD (Create, Read, Update, Delete) operations.
8. **To provide a reliable and efficient way** to retrieve booking history and generate reports if needed.

## Scope of the Project

This DBMS-based project focuses solely on the **backend database system** of an Airline Reservation System. It does **not** include a complete frontend interface (like a mobile or web app), but it provides:

- A well-structured **EER (Enhanced Entity-Relationship) model**
- A **normalized relational schema**
- **SQL code** to implement tables and sample queries
- **Support for flight booking, cancellation, and availability checking**
- Scalability to include additional modules like staff management, notifications, or loyalty programs in the future

The system is designed for academic and conceptual understanding of DBMS principles applied to a real-world scenario.

# Entity-Relationship Model

## Main Entities and Their Attributes

### *1. Passenger*

- **PassengerID (PK)**
- **Name**
- **Email**
- **PhoneNumber**
- **PassportNumber**

### *2. Flight*

- **FlightID (PK)**
- **AirlineName**
- **Source**
- **Destination**
- **DepartureTime**
- **ArrivalTime**
- **TotalSeats**

### *3. Booking*

- **BookingID (PK)**
- **PassengerID (FK)**
- **FlightID (FK)**
- **BookingDate**
- **SeatNumber**
- **Status (Confirmed / Cancelled)**

### *4. Payment*

- **PaymentID (PK)**
- **BookingID (FK)**

- **PaymentDate**
- **Amount**
- **PaymentMode (Card, UPI, Wallet, etc.)**

### *5. Admin*

- **AdminID (PK)**
- **Username**
- **Password**

……

## Relationships

- **Passenger** ↔ **Booking** → One-to-Many
  (A passenger can have many bookings)
- **Flight** ↔ **Booking** → One-to-Many
  (Each flight can have multiple bookings)
- **Booking** ↔ **Payment** → One-to-One
  (Each booking has exactly one payment)
- **Admin** ↔ **Flight** → Optional (One-to-Many, if tracking which admin added/updated the flight)

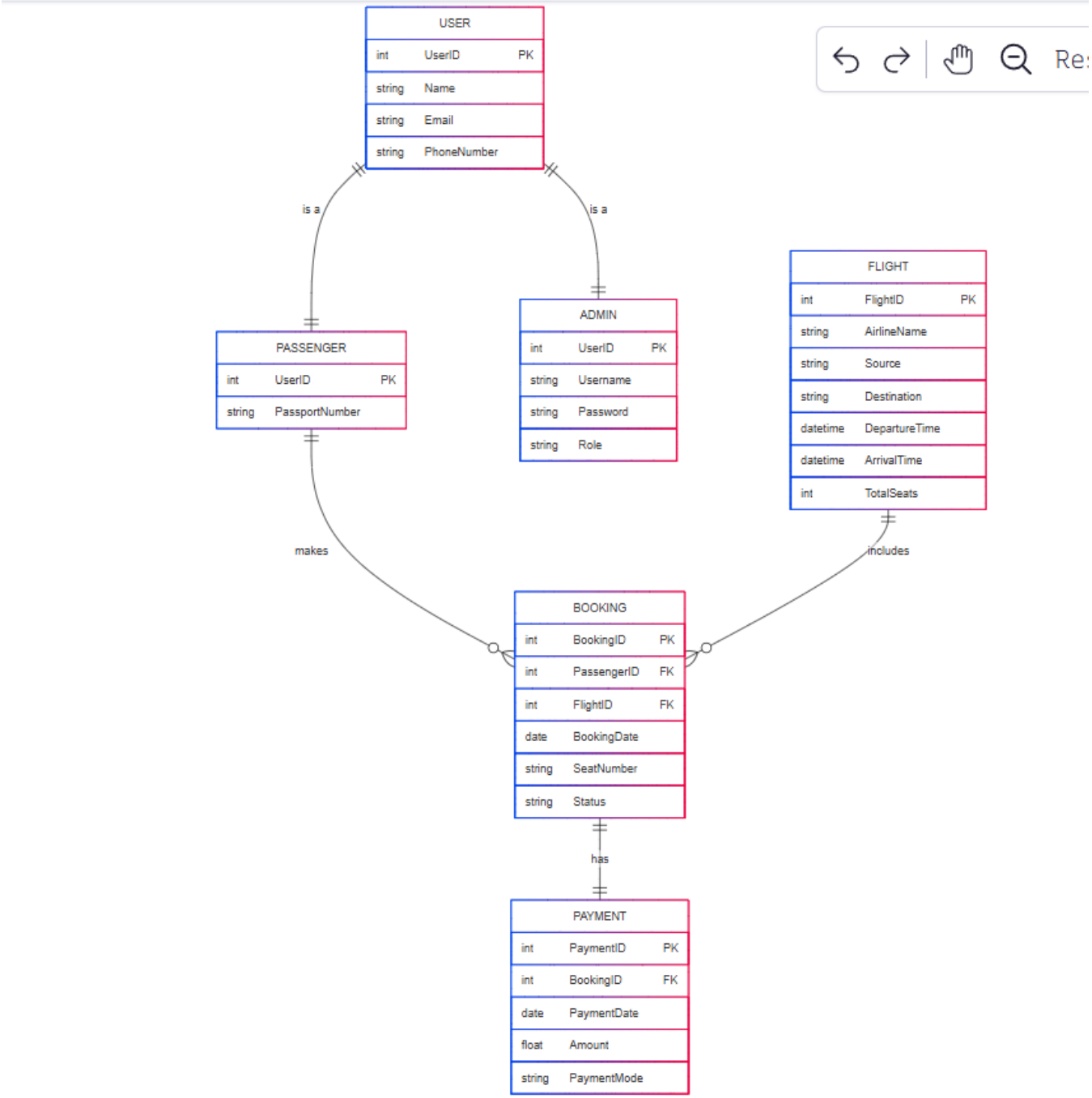
---

## Enhanced Features (EER)

In the EER diagram, we add:

- **Generalization (Optional):**
  `User` → `Passenger` and `Admin`
  *(You can use this if you want to showcase EER capabilities)*
- **Weak Entity:**
  If you want to show `Seat` as a weak entity dependent on `Flight`.
- **Multivalued Attribute (Optional):**
  For example, if a passenger can have multiple contact numbers.

# EER diagram image



## Relational Schema

This step translates the **EER diagram** into actual **normalized relational tables** with **Primary Keys (PK)** and **Foreign Keys (FK)**.

# TABLE

| Relationship | Entities Involved | Type | Cardinality | Description |
|---|---|---|---|---|
| User–Passenger | User, Passenger | Inheritance | 1:1 | Each passenger is a user |
| User–Admin | User, Admin | Inheritance | 1:1 | Each admin is a user |
| Passenger–Booking | Passenger, Booking | One-to-Many | 1:N | A passenger can make multiple bookings |
| Flight–Booking | Flight, Booking | One-to-Many | 1:N | A flight can have many bookings |
| Booking–Payment | Booking, Payment | One-to-One | 1:1 | Each booking has one payment |
| Admin–Flight *(optional)* | Admin, Flight | One-to-Many | 1:N | Admin can manage/add multiple flights (optional use) |

## TABLE CREATION:

```
create database airline;
use airline;


-- Table: User
CREATE TABLE User (
    UserID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15)
);

-- Table: Passenger (inherits from User)
CREATE TABLE Passenger (
    UserID INT PRIMARY KEY,
    PassportNumber VARCHAR(20),
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);

-- Table: Admin (inherits from User)
CREATE TABLE Admin (
    UserID INT PRIMARY KEY,
    Username VARCHAR(50) UNIQUE,
    Password VARCHAR(100),
    Role VARCHAR(50),
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);

-- Table: Flight
CREATE TABLE Flight (
    FlightID INT PRIMARY KEY,
    AirlineName VARCHAR(100),
    Source VARCHAR(100),
    Destination VARCHAR(100),
    DepartureTime DATETIME,
    ArrivalTime DATETIME,
    TotalSeats INT
);

-- Table: Booking
CREATE TABLE Booking (
    BookingID INT PRIMARY KEY,
    PassengerID INT,
```

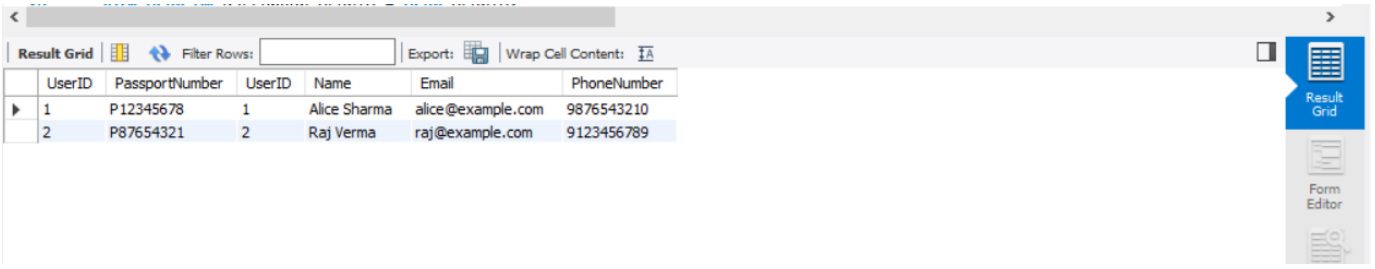
```
    FlightID INT,
    BookingDate DATE,
    SeatNumber VARCHAR(10),
    Status VARCHAR(20),
    FOREIGN KEY (PassengerID) REFERENCES Passenger(UserID),
    FOREIGN KEY (FlightID) REFERENCES Flight(FlightID)
);

-- Table: Payment
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY,
    BookingID INT,
    PaymentDate DATE,
    Amount DECIMAL(10,2),
    PaymentMode VARCHAR(50),
    FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)
);
INSERT INTO User (UserID, Name, Email, PhoneNumber)
VALUES
(1, 'Alice Sharma', 'alice@example.com', '9876543210'),
(2, 'Raj Verma', 'raj@example.com', '9123456789');
INSERT INTO Passenger (UserID, PassportNumber)
VALUES
(1, 'P12345678'),
(2, 'P87654321');
INSERT INTO Admin (UserID, Username, Password, Role)
VALUES
(3, 'admin01', 'admin@123', 'Manager');

INSERT INTO Flight (FlightID, AirlineName, Source, Destination, DepartureTime, ArrivalTime,
TotalSeats)
VALUES
(101, 'IndiGo', 'Delhi', 'Mumbai', '2025-05-10 10:00:00', '2025-05-10 12:00:00', 180),
(102, 'Air India', 'Chennai', 'Kolkata', '2025-05-12 15:00:00', '2025-05-12 18:00:00', 160);

INSERT INTO Booking (BookingID, PassengerID, FlightID, BookingDate, SeatNumber, Status)
VALUES
(501, 1, 101, '2025-04-10', '12A', 'Confirmed'),
(502, 2, 102, '2025-04-11', '16B', 'Confirmed');

INSERT INTO Payment (PaymentID, BookingID, PaymentDate, Amount, PaymentMode)
VALUES
(901, 501, '2025-04-10', 4500.00, 'UPI'),
(902, 502, '2025-04-11', 5200.00, 'Card');

SELECT * FROM Passenger
JOIN User ON Passenger.UserID = User.UserID;

SELECT U.Name, F.AirlineName, F.Source, F.Destination, B.SeatNumber, B.Status
```

```
FROM Booking B
JOIN Passenger P ON B.PassengerID = P.UserID
JOIN User U ON P.UserID = U.UserID
JOIN Flight F ON B.FlightID = F.FlightID;

SELECT U.Name, P.PaymentDate, P.Amount, P.PaymentMode
FROM Payment P
JOIN Booking B ON P.BookingID = B.BookingID
JOIN Passenger Pa ON B.PassengerID = Pa.UserID
JOIN User U ON Pa.UserID = U.UserID;

SELECT * FROM FlightZ
WHERE Source = 'Delhi' AND Destination = 'Mumbai';

UPDATE User
SET PhoneNumber = '9988776655'
WHERE UserID = 1;

UPDATE Booking
SET Status = 'Cancelled'
WHERE BookingID = 501;

UPDATE Flight
SET DepartureTime = '2025-05-10 11:00:00'
WHERE FlightID = 101;

UPDATE Admin
SET Password = 'newSecurePassword123'
WHERE UserID = 3;

UPDATE Payment
SET PaymentMode = 'Credit Card'
WHERE PaymentID = 901;
--
DELETE FROM Payment
WHERE PaymentID = 901;

DELETE FROM Booking
WHERE BookingID = 501;

DELETE FROM User
WHERE UserID = 1;

DELETE FROM Passenger
WHERE UserID = 1;
DELETE FROM Flight
WHERE FlightID = 101;
```

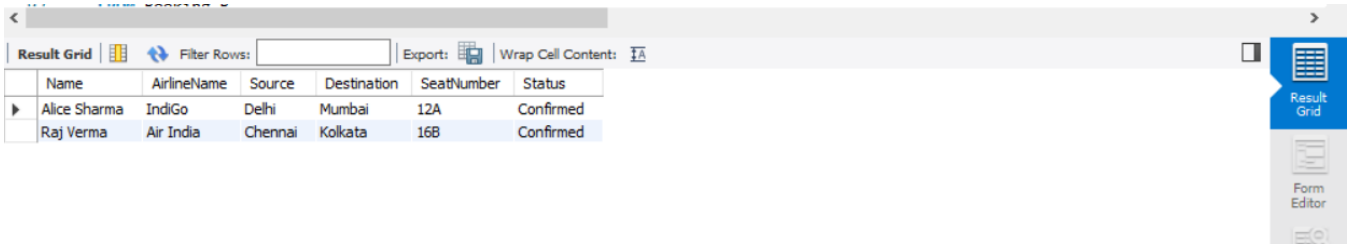
```
SELECT F.FlightID, F.DAirlineName, COUNT(B.BookingID) AS BookingCount
FROM Flight F
LEFT JOIN Booking B ON F.FlightID = B.FlightID
GROUP BY F.FlightID, F.AirlineName;

SELECT F.FlightID, F.AirlineName, SUM(P.Amount) AS TotalPayments
FROM Flight F
JOIN Booking B ON F.FlightID = B.FlightID
JOIN Payment P ON B.BookingID = P.BookingID
GROUP BY F.FlightID, F.AirlineName;

SELECT F.FlightID, F.AirlineName, F.Source, F.Destination
FROM Flight F
LEFT JOIN Booking B ON F.FlightID = B.FlightID
WHERE B.BookingID IS NULL;

SELECT U.UserID, U.Name, U.Email,
    (SELECT COUNT(*) FROM Booking WHERE PassengerID = U.UserID) AS TotalBookings
FROM User U
JOIN Passenger P ON U.UserID = P.UserID;
```

# SQL Queries & Output

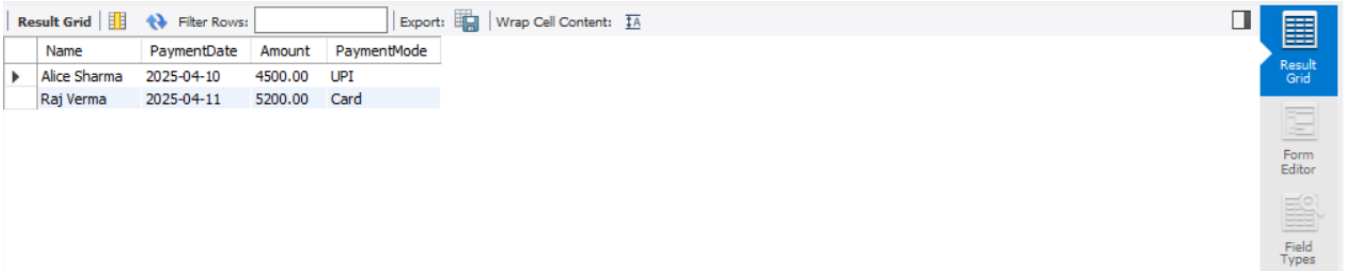
## --INSERT Queries



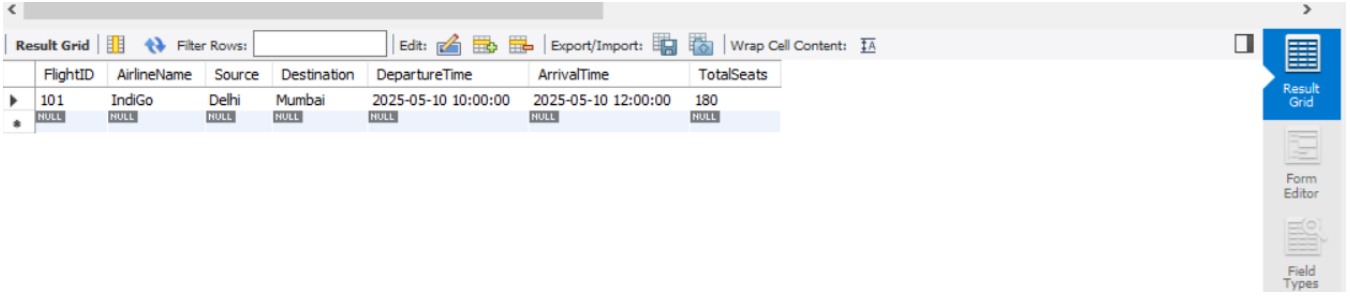
| UserID | PassportNumber | UserID | Name | Email | PhoneNumber |
|---|---|---|---|---|---|
| 1 | P12345678 | 1 | Alice Sharma | alice@example.com | 9876543210 |
| 2 | P87654321 | 2 | Raj Verma | raj@example.com | 9123456789 |



| Name | AirlineName | Source | Destination | SeatNumber | Status |
|---|---|---|---|---|---|
| Alice Sharma | IndiGo | Delhi | Mumbai | 12A | Confirmed |
| Raj Verma | Air India | Chennai | Kolkata | 16B | Confirmed |

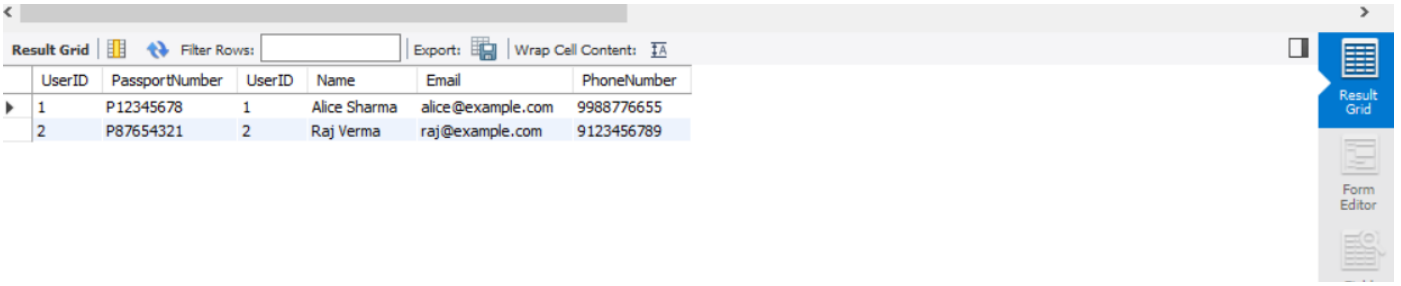


| Name | PaymentDate | Amount | PaymentMode |
|---|---|---|---|
| Alice Sharma | 2025-04-10 | 4500.00 | UPI |
| Raj Verma | 2025-04-11 | 5200.00 | Card |



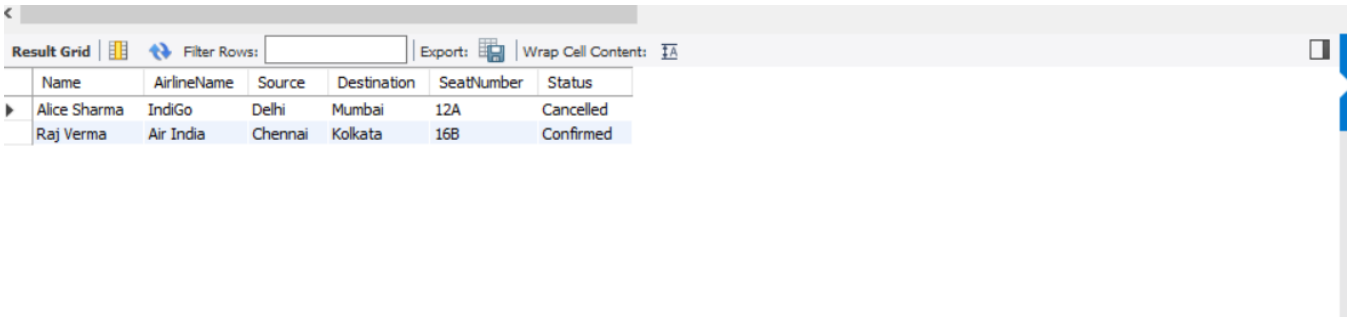
| FlightID | AirlineName | Source | Destination | DepartureTime | ArrivalTime | TotalSeats |
|---|---|---|---|---|---|---|
| 101 | IndiGo | Delhi | Mumbai | 2025-05-10 10:00:00 | 2025-05-10 12:00:00 | 180 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

# --UPDATE Queries



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

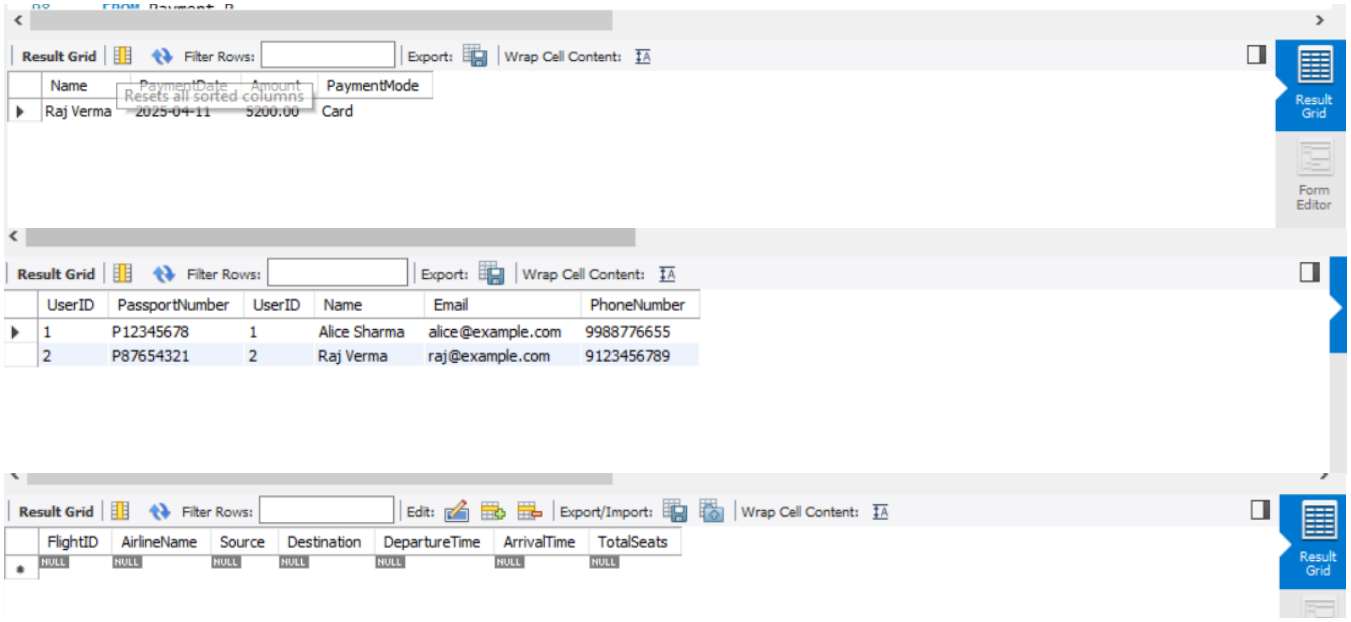
| UserID | PassportNumber | UserID | Name | Email | PhoneNumber |
|---|---|---|---|---|---|
| 1 | P12345678 | 1 | Alice Sharma | alice@example.com | 9988776655 |
| 2 | P87654321 | 2 | Raj Verma | raj@example.com | 9123456789 |



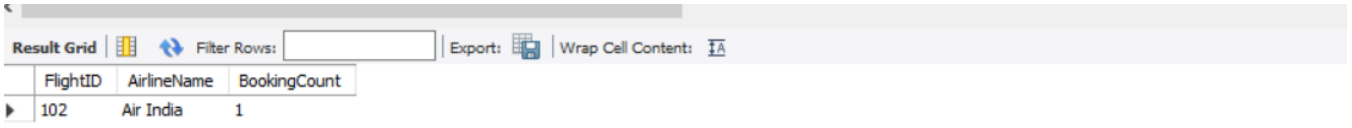
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Name | AirlineName | Source | Destination | SeatNumber | Status |
|---|---|---|---|---|---|
| Alice Sharma | IndiGo | Delhi | Mumbai | 12A | Cancelled |
| Raj Verma | Air India | Chennai | Kolkata | 16B | Confirmed |

# --DELETE Queries



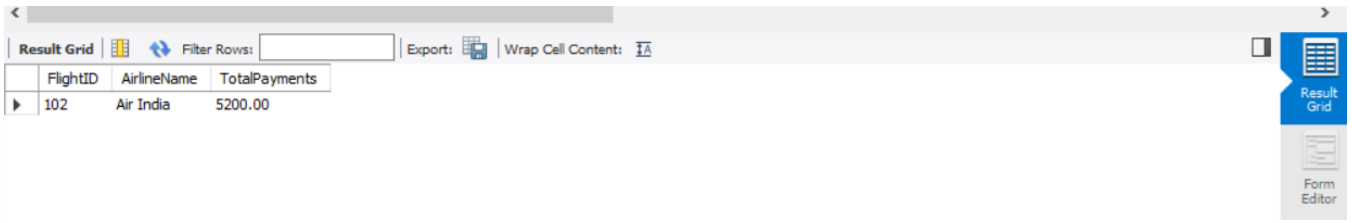
## --Count Bookings for Each Flight



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| FlightID | AirlineName | BookingCount |
|---|---|---|
| 102 | Air India | 1 |

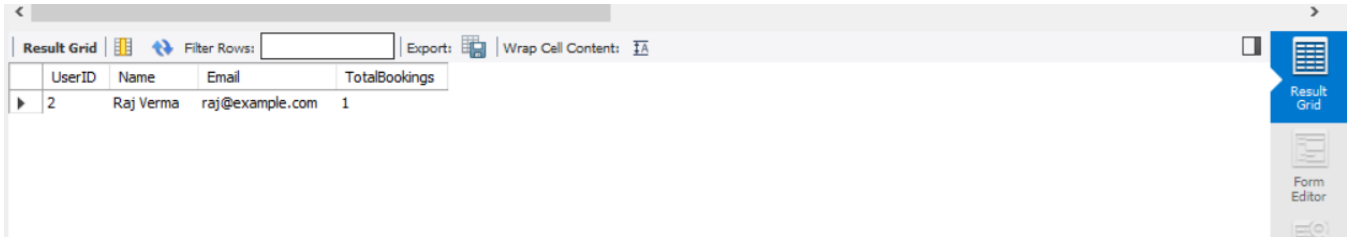
## --Total Payment Amount per Flight



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| FlightID | AirlineName | TotalPayments |
|---|---|---|
| 102 | Air India | 5200.00 |

## -- passenger Details with Booking Count Using a Subquery



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| UserID | Name | Email | TotalBookings |
|---|---|---|---|
| 2 | Raj Verma | raj@example.com | 1 |

# READ ME

# Airline Reservation System – DBMS Case Study

## Overview

The **Airline Reservation System** is a database management system (DBMS) project designed to simulate core functionalities of an airline ticketing platform. The project covers the entire lifecycle of airline operations including user management (admin and passengers), flight details, bookings, and payment processing. It aims to demonstrate key DBMS concepts such as:

- Entity-Relationship (ER) modeling and Enhanced ER (EER) with generalization.
- Relational schema design and database normalization (up to 3NF).
- SQL operations (CREATE, INSERT, SELECT, UPDATE, DELETE) for data manipulation.
- Handling complex queries involving JOINs, aggregate functions, subqueries, and unions.

This project is intended for educational purposes as part of a college-level DBMS case study.

## Project Structure

The case study documentation and SQL scripts are structured as follows:

1. **Title Page and Acknowledgement:**
   Covers the project title, author details, and acknowledgements.
2. **Abstract:**
   Summarizes the project objectives, scope, and key features.
3. **Introduction, Objectives, and Scope:**
   Explains the rationale behind the project, system objectives, and overall project scope.
4. **Requirements Gathering:**
   Details both functional and non-functional requirements of the system.
5. **ER/EER Diagrams:**
   Presents the Entity-Relationship model, including enhancements through generalization (User → Admin & Passenger).
   *The EER diagram is provided in Mermaid syntax for visualization using Mermaid Live Editor.*
6. **Relational Schema Design:**
   Converts the ER/EER diagrams into normalized relational tables with primary and foreign keys.
7. **SQL Code:**
   Contains complete SQL scripts for creating tables, inserting sample data, and performing data operations (CRUD).

- **CREATE TABLE** scripts for `User`, `Passenger`, `Admin`, `Flight`, `Booking`, and `Payment`.
- **Sample INSERT, SELECT, UPDATE, and DELETE queries** to demonstrate database functionalities.
- Additional queries for aggregating data and handling subqueries.

8. **Case Study Analysis & Documentation:**
   Discussion on normalization, schema design decisions, and the significance of each module in solving real-world problems.

---

# Prerequisites

Before setting up the database, ensure you have the following installed on your system:

- **RDBMS Software:**
  MySQL, PostgreSQL, or any other SQL-compliant database server.
- **SQL Client Tools:**
  MySQL Workbench, pgAdmin, or any terminal/command-line tool to execute SQL scripts.
- **Mermaid Live Editor (Optional):**
  To visualize the EER diagram, visit Mermaid Live Editor.

---

# Setup Instructions

1. **Database Creation:**
   - Create a new database in your RDBMS environment (e.g., `airline_reservation`).
2. **Running the SQL Scripts:**
   - Copy the complete SQL code provided in the case study.
   - Execute the code block containing `CREATE TABLE` statements to create the database schema.
   - Optionally, run the sample `INSERT` statements to populate the tables with initial test data.
   - Finally, test the additional queries (SELECT, UPDATE, DELETE) to ensure the data operations are working as expected.
3. **Visualizing the EER Diagram:**
   - Open Mermaid Live Editor.
   - Paste the provided Mermaid syntax for the EER diagram to generate and export the diagram if needed.

---

# Running the SQL Queries

## Execution Order:

1. **Table Creation:**
   - Run the `CREATE TABLE` scripts.
2. **Data Insertion:**
   - Execute the `INSERT` scripts to populate your tables.
3. **Query Testing:**
   - Use the `SELECT` queries to retrieve and verify data.

4. **Update/Delete Operations:**
    - Test `UPDATE` and `DELETE` queries as needed.

## Example:

- After logging into your SQL client and selecting the `airline_reservation` database, run:

```
sql
CopyEdit
-- Create tables
-- [Paste the complete SQL code block here]

-- Insert sample data
-- [Paste sample INSERT statements here]

-- Run a sample query
SELECT * FROM Flight;
```

---

## Usage

- **Administration:**
  Admin users can manage flights and view bookings.
- **Passenger Services:**
  Registered passengers can search for flights, book tickets, and view their booking history.

- **Data Maintenance:**
  CRUD operations are implemented to ensure data integrity and manage the lifecycle of database records.

---

## Notes & Future Enhancements

- **Extensibility:**
  The current design serves as a foundational framework. Future enhancements can include additional modules such as user authentication, real-time flight updates, loyalty programs, etc.
- **Validation & Security:**
  While the current setup focuses on DBMS concepts, real-world applications would require added layers of security and input validation.
- **Performance:**
  With the given schema design, indexing, and optimized queries are recommended for production-level systems.

---

## References


- Database Management System textbooks and articles.
- Online resources, tutorials, and official documentation from RDBMS vendors.
- Mermaid Documentation for visualizing EER diagrams.


---

# Summary

The **Airline Reservation System** case study demonstrates the complete lifecycle of database system development, from analysis and design to implementation and testing. It highlights how real-world problems can be effectively addressed using a structured DBMS approach.

Key outcomes of this project include:

- **Entity-Relationship Modeling:**
  A comprehensive ER and EER diagram capturing the core entities such as User, Admin, Passenger, Flight, Booking, and Payment. It also showcases generalization (User → Admin/Passenger).
- **Normalized Database Design:**
  All database tables are normalized up to the Third Normal Form (3NF), ensuring data consistency, minimal redundancy, and relational integrity.
- **Full SQL Implementation:**
  The SQL scripts cover the entire schema creation, sample data insertion, and common data operations including `SELECT`, `UPDATE`, `DELETE`, and advanced queries using `JOIN`, `GROUP BY`, and subqueries.
- **Realistic Use Cases:**
  The system models real-world functionalities such as flight

scheduling, ticket booking, and payment, providing a practical understanding of relational databases.

- **Scalable Design:**
  The database structure is built in a modular way, enabling easy future enhancements such as integration with a web or mobile front-end, implementation of loyalty systems, or automation features.

## Conclusion

This project underscores the effective design, implementation, and management of a relational database tailored for an airline reservation system. It provides a practical demonstration of how fundamental DBMS principles can be utilized to solve real-world problems in the aviation domain.

---

By following this README, you can set up, execute, and further expand the Airline Reservation System as a functional DBMS project.

---

Feel free to customize this README further to align with your personal style or any additional requirements from your college. Let me know if you need any further assistance or additional sections!o3-mini