**FLIGHT BOOKING SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

Chirasani Likkith Reddy – AP22110010179

Surya Mannava - AP22110010154

Amarnadh kalluri - AP22110010151

Nena - AP22110010163

Bhuvanesh Reddy - AP22110010168



Under the Guidance of

Aurobindo Behera

**Department of Computer Science and Engineering** **SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

# Table of Contents

1. Identification of Project related to DBMS project (project Title)

1. Project Background

1. Description of the Project

1. ER Diagram Creation (use any online tools to draw ER diagram)

1. Description of ER diagram

1. Conversion of ER diagram into Tables

1. Description of Tables

1. Normalization of tables up to 3-NF

1. Creation of Data in the tables (at least 5 tables)

1. Few sql queries on the created tables (your choice)

1. Creation of 5 views using the tables

# Project background:

Flight booking systems have become indispensable tools in the modern travel landscape, revolutionizing the way individuals plan and book their journeys. This abstract offers a succinct overview of flight booking systems, highlighting their significance, functionalities, challenges, and future prospects.

Flight booking systems serve as digital platforms that enable users to search, compare, and reserve flights from various airlines. These systems leverage advanced technologies, including user-friendly interfaces, real-time data integration, and secure payment gateways, to streamline the booking process.

Challenges such as cybersecurity threats, pricing complexities, and the dynamic nature of the travel industry require continuous innovation and adaptation. However, advancements in artificial intelligence, machine learning, and blockchain technology offer opportunities to overcome these challenges and enhance the efficiency and reliability of flight booking systems.

In conclusion, flight booking systems play a pivotal role in simplifying travel arrangements, providing travellers with convenience, flexibility, and choice. As technology continues to evolve, these systems are poised to deliver even more personalized and seamless booking experiences, further enriching the journey of travellers worldwide.

# Description of project:

The flight booking system described in the SQL code serves as a foundational structure for managing airline operations and passenger bookings. It acts as a centralized hub where crucial information about airlines, airports, flights, and passengers is stored and organized. This system is vital for ensuring smooth airline operations and efficient passenger management.

**Airlines List:**

The Airlines List table contains details about different airlines, such as their names. This table helps in identifying the airlines operating within the system and allows for easy reference when associating flights with specific airlines.

**Airport List:**

The Airport List table stores information about various airports, including their names and locations. This table is essential for identifying departure and arrival airports for flights and helps in providing accurate airport information to passengers.

**Flight List:**

The Flight List table manages detailed flight schedules, including information like the airline associated with the flight, the plane number, departure and arrival airports, departure and arrival times, available seats, and ticket prices. This table serves as a central repository for flight-related information, enabling airlines to efficiently manage their flight schedules and bookings.

**Booked Flight:**

The Booked Flight table records passenger bookings for flights. It includes details such as the flight ID, passenger name, address, and contact information. This table helps in managing passenger bookings and facilitates communication with passengers regarding their travel details.

**Users:**

The Users table stores information about users who can access the system. This includes their usernames, passwords, email addresses, and phone numbers. This table ensures secure access to the system and allows for user-specific functionalities, such as booking flights and managing reservations.

In conclusion, the flight booking system outlined by the SQL code is a comprehensive database schema that plays a crucial role in managing airline operations and passenger bookings.

# Methodology

* **Database Design:** 
  + Describe the database schema, including tables for airlines, airports, flights, booked flights, and users.
  + Explain the rationale behind the design decisions, such as the use of primary keys, foreign keys, and data types.

* **Data Population:** 
  + Discuss the process of populating the database with sample data, including airlines, airports, flights, and passenger bookings.
  + Highlight any challenges or considerations in data population.

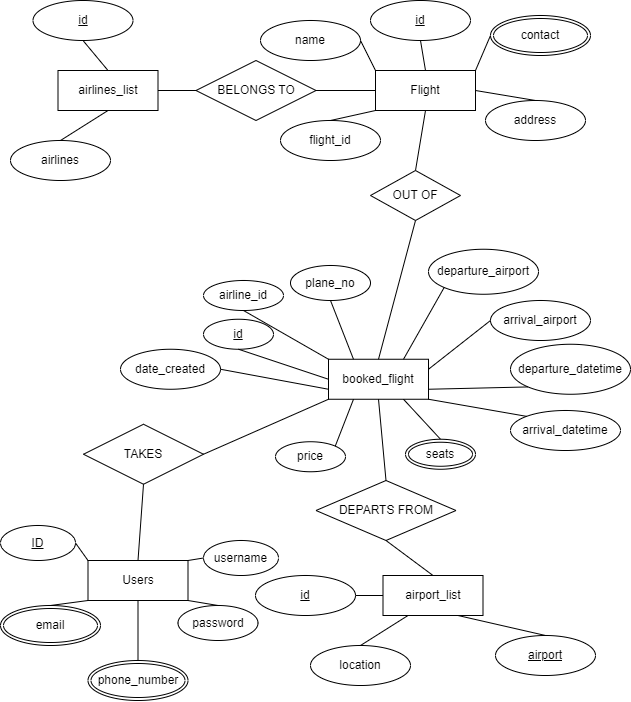
* **System Functionality:** 
  + Explain the functionality of the flight booking system, including how it manages flight schedules, seat availability, and passenger bookings.
  + Describe how users interact with the system, such as searching for flights and making reservations.

* **SQL Queries:** 
  + Provide examples of SQL queries used in the system, such as retrieving flight information, booking flights, and updating passenger details.
  + Explain the purpose and functionality of each query.

* **Normalization and Optimization:**
  + Applied normalization techniques to reduce data redundancy and improve database performance.
  + Optimized SQL queries and database operations to improve overall system efficiency and responsiveness.

# Entity Relation Diagram (ER Diagram)

# 



Description of ER diagram:

1. **Entities and Attributes**:
   * **Airlines**: Represents different airlines. Attributes may include an auto-incremented ID and the name of the airline.
   * **Airports**: Represents various airports. Attributes may include an auto-incremented ID, the name of the airport, and its location.
   * **Flights**: Represents individual flights. Attributes may include an auto-incremented ID, the ID of the airline operating the flight, the plane number, departure and arrival airport IDs, departure and arrival datetimes, available seats, price, and creation timestamp.
   * **Booked Flights**: Represents flights that have been booked by users. Attributes may include an auto-incremented ID, the ID of the booked flight, the name, address, and contact information of the person booking the flight.
   * **Users**: Represents users of the system. Attributes may include an auto-incremented ID, username, password, email, and phone number.
2. **Relationships**:
   * **Airlines and Flights**: Each flight is operated by one airline, establishing a one-to-many relationship between Airlines and Flights.
   * **Airports and Flights**: Each flight has a departure and arrival airport, establishing two one-to-many relationships between Airports and Flights (departure and arrival).
   * **Flights and Booked Flights**: Each booked flight corresponds to one flight, establishing a one-to-many relationship between Flights and Booked Flights.
   * **Users and Booked Flights**: Each booked flight is associated with one user who made the booking, establishing a one-to-many relationship between Users and Booked Flights.
3. **Attributes with Keys**:
   * Primary keys are marked for each entity (ID fields).
   * Foreign keys are used to establish relationships between entities (e.g., airline\_id, departure\_airport\_id, arrival\_airport\_id, flight\_id).

Conversion of ER diagram into Tables:

**Airlines Table**:

CREATE TABLE airlines\_list (

id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

airlines text NOT NULL

);

**Airports Table**:

CREATE TABLE airports (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

location VARCHAR(255) NOT NULL

);

**Flights Table**:

CREATE TABLE flights (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

airline\_id INT NOT NULL,

plane\_no VARCHAR(50) NOT NULL,

departure\_airport\_id INT NOT NULL,

arrival\_airport\_id INT NOT NULL,

departure\_datetime DATETIME NOT NULL,

arrival\_datetime DATETIME NOT NULL,

seats INT NOT NULL,

price DOUBLE NOT NULL,

date\_created DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

FOREIGN KEY (airline\_id) REFERENCES airlines(id),

FOREIGN KEY (departure\_airport\_id) REFERENCES airports(id),

FOREIGN KEY (arrival\_airport\_id) REFERENCES airports(id)

);

**Booked Flights Table**:

CREATE TABLE booked\_flights (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

flight\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

contact VARCHAR(20) NOT NULL,

FOREIGN KEY (flight\_id) REFERENCES flights(id)

);

**Users Table**:

CREATE TABLE users (

id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone\_number VARCHAR(20) NOT NULL

);

Description of Tables:

1. **Airlines Table**:
   * **id**: An auto-incremented unique identifier for each airline.
   * **name**: The name of the airline.
2. **Airports Table**:
   * **id**: An auto-incremented unique identifier for each airport.
   * **name**: The name of the airport.
   * **location**: The location or city where the airport is located.
3. **Flights Table**:
   * **id**: An auto-incremented unique identifier for each flight.
   * **airline\_id**: A foreign key referencing the id of the airline operating the flight.
   * **plane\_no**: The plane number or identifier for the flight.
   * **departure\_airport\_id**: A foreign key referencing the id of the departure airport.
   * **arrival\_airport\_id**: A foreign key referencing the id of the arrival airport.
   * **departure\_datetime**: The date and time of departure for the flight.
   * **arrival\_datetime**: The date and time of arrival for the flight.
   * **seats**: The number of available seats on the flight.
   * **price**: The price of the flight.
   * **date\_created**: The date and time when the flight record was created.
4. **Booked Flights Table**:
   * **id**: An auto-incremented unique identifier for each booked flight.
   * **flight\_id**: A foreign key referencing the id of the flight that has been booked.
   * **name**: The name of the person who booked the flight.
   * **address**: The address of the person who booked the flight.
   * **contact**: The contact information of the person who booked the flight.
5. **Users Table**:
   * **id**: An auto-incremented unique identifier for each user.
   * **username**: The username chosen by the user for login.
   * **password**: The password associated with the user's account.
   * **email**: The email address of the user.
   * **phone\_number**: The phone number of the user.

Each table serves a specific purpose within the database schema for managing airline-related data, including information about airlines, airports, flights, booked flights, and users. The tables are designed to capture relevant attributes and establish relationships between entities where necessary to maintain data integrity and support the functionality of the airline system.

Normalization of tables up to 3-NF:

**Step 1: Identify Functional Dependencies:**

* Analyse the tables to identify functional dependencies between attributes.

**Step 2: Normalize to 1NF:**

1. **Airlines Table**:
   * Already in 1NF as it has a primary key (**id**) and each column contains atomic values.
2. **Airports Table**:
   * Already in 1NF as it has a primary key (**id**) and each column contains atomic values.
3. **Flights Table**:
   * Already in 1NF as it has a primary key (**id**) and each column contains atomic values.
4. **Booked Flights Table**:
   * Already in 1NF as it has a primary key (**id**) and each column contains atomic values.
5. **Users Table**:
   * Already in 1NF as it has a primary key (**id**) and each column contains atomic values.

**Step 3: Normalize to 2NF:**

* There are no partial dependencies in the provided tables, so they are already in 2NF.

**Step 4: Normalize to 3NF:**

* Ensure there are no transitive dependencies.

In the provided schema, all tables are already in 3NF as there are no transitive dependencies. Each non-key attribute is directly dependent on the primary key, and there are no dependencies between non-key attributes.

Therefore, the tables are already normalized up to the Third Normal Form (3NF), and no further normalization steps are required.

Code:

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

create database airlineee;

use airlineee;

drop database airlineee;

CREATE TABLE airlines\_list (

id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

airlines text NOT NULL

);

CREATE TABLE airport\_list (

id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

airport text NOT NULL,

location text NOT NULL

);

CREATE TABLE flight\_list (

id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

airline\_id int NOT NULL,

plane\_no text NOT NULL,

departure\_airport\_id int NOT NULL,

arrival\_airport\_id int NOT NULL,

departure\_datetime datetime NOT NULL,

arrival\_datetime datetime NOT NULL,

seats int NOT NULL DEFAULT 0,

price double NOT NULL,

date\_created datetime NOT NULL DEFAULT current\_timestamp(),

FOREIGN KEY (airline\_id) REFERENCES airlines\_list(id),

FOREIGN KEY (departure\_airport\_id) REFERENCES airport\_list(id),

FOREIGN KEY (arrival\_airport\_id) REFERENCES airport\_list(id)

);

CREATE TABLE booked\_flight (

id int NOT NULL PRIMARY KEY AUTO\_INCREMENT,

flight\_id int NOT NULL,

name text NOT NULL,

address text NOT NULL,

contact text NOT NULL,

FOREIGN KEY (flight\_id) REFERENCES flight\_list(id)

);

CREATE TABLE Users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone\_number VARCHAR(20) NOT NULL

);

Few SQL queries on the created tables:

ALTER TABLE airlines\_list

MODIFY id int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE airport\_list

MODIFY id int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

ALTER TABLE booked\_flight

MODIFY id int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

ALTER TABLE flight\_list

MODIFY id int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

ALTER TABLE Users

MODIFY id int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

COMMIT;

Creation of Data in the tables:

INSERT INTO airlines\_list (id, airlines) VALUES

(1, 'Air India'),

(2, 'IndiGo'),

(3, 'SpiceJet');

INSERT INTO airport\_list (id, airport, location) VALUES

(1, 'Indira Gandhi International Airport', 'New Delhi'),

(2, 'Chhatrapati Shivaji Maharaj International Airport', 'Mumbai'),

(3, 'Kempegowda International Airport', 'Bengaluru'),

(4, 'Netaji Subhas Chandra Bose International Airport', 'Kolkata'),

(5, 'Rajiv Gandhi International Airport', 'Hyderabad');

INSERT INTO flight\_list (id, airline\_id, plane\_no, departure\_airport\_id, arrival\_airport\_id, departure\_datetime, arrival\_datetime, seats, price) VALUES

(1, 1, 'AI101', 1, 2, '2024-04-21 08:00:00', '2024-04-21 11:00:00', 200, 8000),

(2, 2, '6E205', 2, 3, '2024-04-22 10:00:00', '2024-04-22 12:30:00', 150, 6000),

(3, 3, 'SG321', 3, 4, '2024-04-23 12:00:00', '2024-04-23 15:00:00', 180, 7000),

(4, 1, 'AI102', 1, 2, '2024-04-21 14:00:00', '2024-04-21 17:00:00', 180, 8500),

(5, 2, '6E206', 2, 3, '2024-04-22 08:30:00', '2024-04-22 10:30:00', 160, 6200),

(6, 3, 'SG322', 4, 5, '2024-04-23 09:00:00', '2024-04-23 12:00:00', 200, 7200);

INSERT INTO booked\_flight (id, flight\_id, name, address, contact) VALUES

(1, 1, 'Rahul Sharma', '123, New Delhi', '+91 9876543210'),

(2, 2, 'Priya Singh', '456, Mumbai', '+91 9876543211'),

(3, 3, 'Amit Patel', '789, Bengaluru', '+91 9876543212');

INSERT INTO Users (id ,username, password, email, phone\_number) VALUES

(1,'rahul\_sharma', 'rs@123', 'rahul.sharma@example.com', '+91 98765 43210'),

(2,'priya\_singh', 'ps@456', 'priya.singh@example.com', '+91 98765 43211'),

(3,'amit\_patel', 'ap@789', 'amit.patel@example.com', '+91 98765 43212');

DELIMITER //

CREATE TRIGGER reduce\_seats\_after\_booking

AFTER INSERT ON booked\_flight

FOR EACH ROW

BEGIN

UPDATE flight\_list

SET seats = seats - 1

WHERE id = NEW.flight\_id;

END;

//

DELIMITER ;

INSERT INTO booked\_flight (flight\_id, name, address, contact) VALUES

(4, 'Sneha Patel', '456, Ahmedabad', '+91 9876543213');

select \* from booked\_flight;

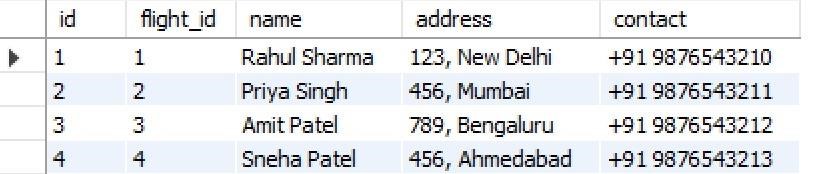
select \* from flight\_list;

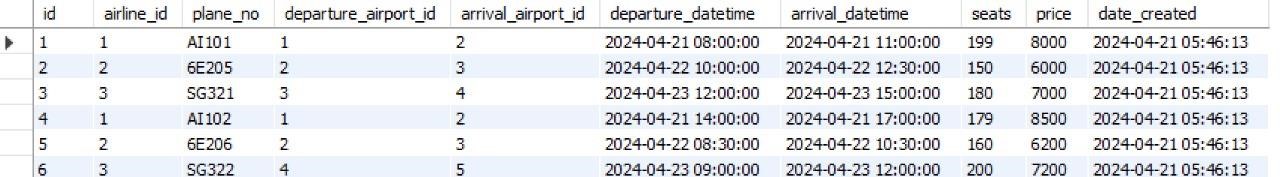
select \* from airport\_list;

select \* from airlines\_list;

select \* from Users;

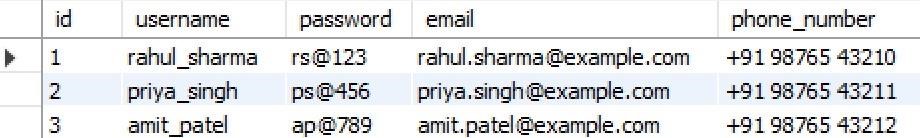
# Output:











Creation of 5 views using the tables:

**View 1: Available Flights:**

CREATE VIEW available\_flights AS

SELECT

f.id AS flight\_id,

a1.name AS departure\_airport,

a2.name AS arrival\_airport,

f.departure\_datetime,

f.arrival\_datetime,

f.seats

FROM

flights f

JOIN

airports a1 ON f.departure\_airport\_id = a1.id

JOIN

airports a2 ON f.arrival\_airport\_id = a2.id;

**View 2: Booked Flights Details:**

CREATE VIEW booked\_flights\_details AS

SELECT

bf.id AS booked\_flight\_id,

bf.name AS passenger\_name,

bf.address AS passenger\_address,

bf.contact AS passenger\_contact,

f.id AS flight\_id,

a1.name AS departure\_airport,

a2.name AS arrival\_airport,

f.departure\_datetime,

f.arrival\_datetime

FROM

booked\_flights bf

JOIN

flights f ON bf.flight\_id = f.id

JOIN

airports a1 ON f.departure\_airport\_id = a1.id

JOIN

airports a2 ON f.arrival\_airport\_id = a2.id;

**View 3: Flights by Airline:**

CREATE VIEW flights\_by\_airline AS

SELECT

f.id AS flight\_id,

a.name AS airline\_name,

f.plane\_no,

a1.name AS departure\_airport,

a2.name AS arrival\_airport,

f.departure\_datetime,

f.arrival\_datetime,

f.seats,

f.price

FROM

flights f

JOIN

airlines a ON f.airline\_id = a.id

JOIN

airports a1 ON f.departure\_airport\_id = a1.id

JOIN

airports a2 ON f.arrival\_airport\_id = a2.id;

**View 4: Users with Booked Flights:**

CREATE VIEW users\_with\_booked\_flights AS

SELECT

u.id AS user\_id,

u.username,

u.email,

u.phone\_number,

bf.name AS passenger\_name,

bf.address AS passenger\_address,

bf.contact AS passenger\_contact,

f.id AS flight\_id,

a1.name AS departure\_airport,

a2.name AS arrival\_airport,

f.departure\_datetime,

f.arrival\_datetime

FROM

users u

JOIN

booked\_flights bf ON u.id = bf.id

JOIN

flights f ON bf.flight\_id = f.id

JOIN

airports a1 ON f.departure\_airport\_id = a1.id

JOIN

airports a2 ON f.arrival\_airport\_id = a2.id;

**View 5: Flights with Low Availability:**

CREATE VIEW flights\_with\_low\_availability AS

SELECT

f.id AS flight\_id,

a.name AS airline\_name,

f.plane\_no,

a1.name AS departure\_airport,

a2.name AS arrival\_airport,

f.departure\_datetime,

f.arrival\_datetime,

f.seats,

f.price

FROM

flights f

JOIN

airlines a ON f.airline\_id = a.id

JOIN

airports a1 ON f.departure\_airport\_id = a1.id

JOIN

airports a2 ON f.arrival\_airport\_id = a2.id

WHERE

f.seats < 10;