



PES UNIVERSITY, Bengaluru  
Department of Computer Science and Engineering  
B. Tech (CSE) – 5th Semester – Aug-Dec 2024

## DBMS

### MINIPROJECT REPORT

#### TITLE: AIRLINE MANAGEMENT SYSTEM

PES1UG22CS363	MUJASEEM D
PES1UG22CS368	N SWETHA

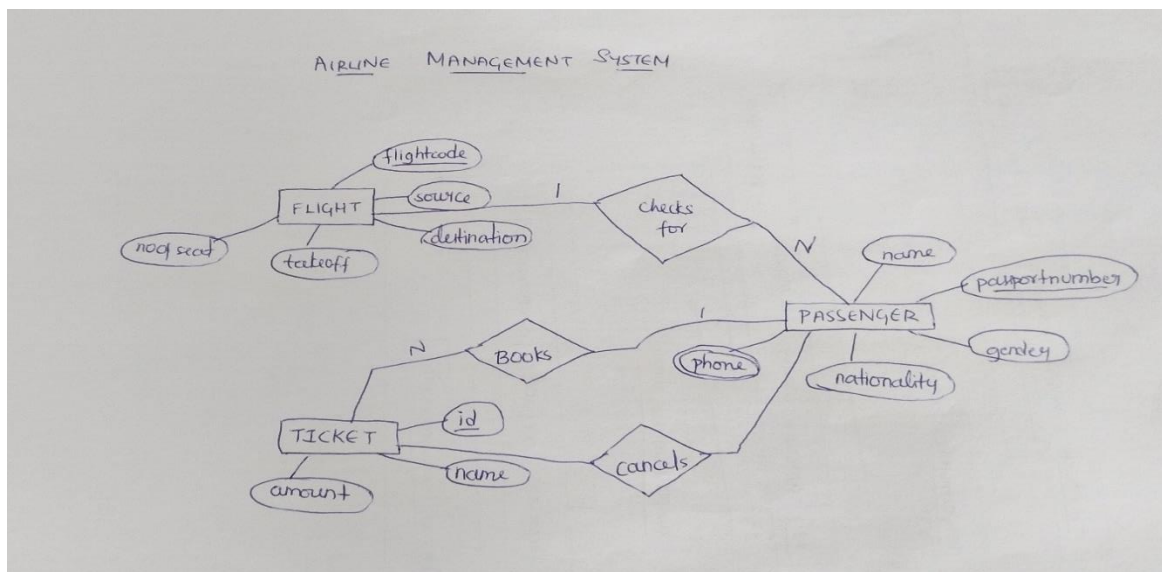
#### Description:

The Airline Management System is a robust database-driven application designed to manage key operations of an airline efficiently. It incorporates the management of users, flights, ticket booking and ticket cancellation ensuring a seamless experience for passengers and admin.

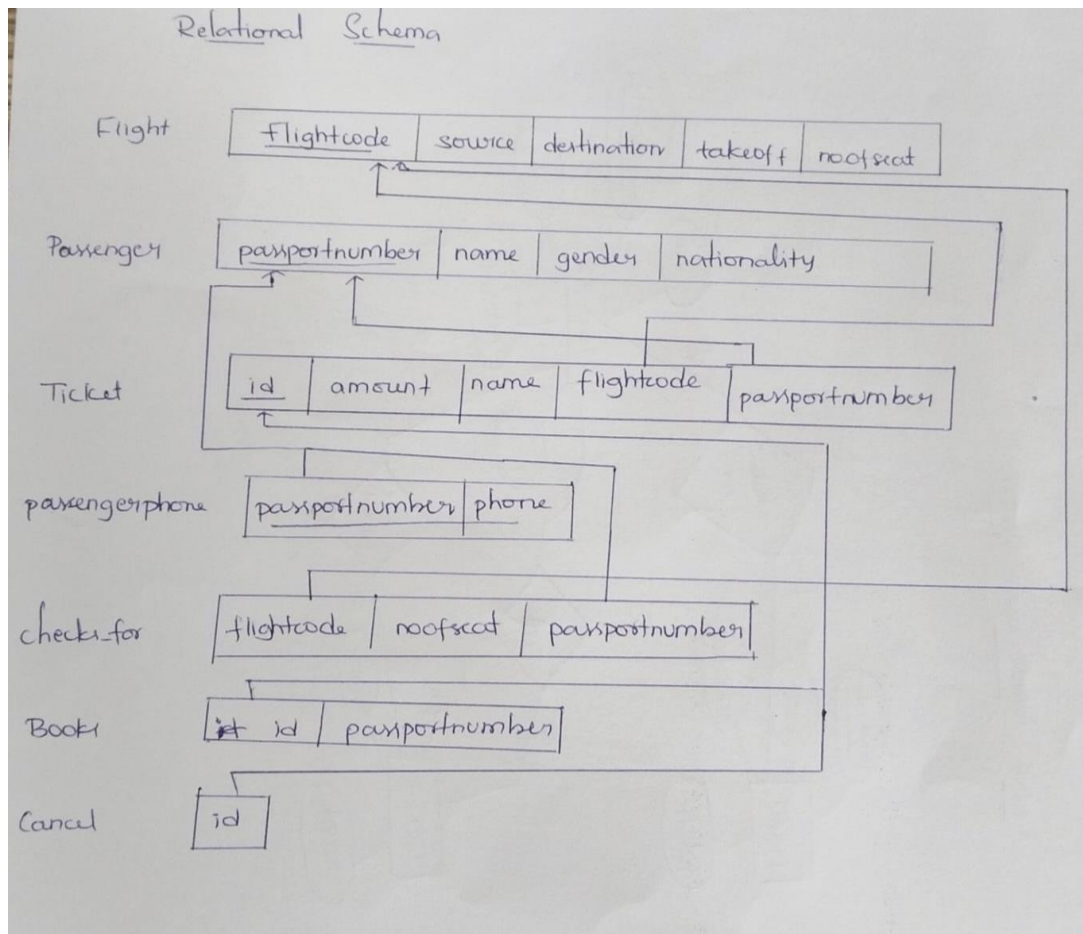
#### Core Features:

- User Management
- Flight Management
- Ticket Management

#### ER DIAGRAM:



## RELATIONAL SCHEMA:



## JOIN:

1. It retrieves the passenger details (name, gender, nationality, and passport number) for all passengers booked on a specified flight.

### Query:

```
SELECT p.name, p.gender, p.nationality, p.passportnumber
FROM ticket_booking t
JOIN managepassenger p ON t.passportnumber = p.passportnumber

WHERE t.flightcode = flightCode;
```

### Output:

Retrieve Passengers for a Flight

Flight Code:

f1-10

Retrieve Passengers

Name: Michael Brown  
Gender:  
Nationality: British  
Passport Number: Y66778899

Name: Shwetha  
Gender: female  
Nationality: Indian  
Passport Number: 11111111

BACK

### NESTED Query:

**Query:** This query retrieves all records from the ticket\_booking table for flights that have 'Bengaluru' as their source city in the manageflight table.

```
SELECT *
FROM
    ticket_booking
WHERE
    flightcode IN (
        SELECT flightcode
        FROM manageflight
        WHERE source = 'Bengaluru'
    );
```

### Output:

id	name	flightcode	gender	passportnumber	amount	nationality
220	Surya	f1-202	Male	Y66778899	5000	Indian

## AGGREGATE FUNCTIONS:

### 1. Highest Ticket Price

This query retrieves the highest ticket price (amount) for each flight by selecting the flightcode and the maximum amount from the ticket\_booking table, grouped by flightcode

**Query:**

```
SELECT flightcode, MAX(amount) AS highest_ticket_price
FROM ticket_booking
GROUP BY flightcode;
```

**Output:**

flightcode	highest_ticket_price
f1-10	5000

### 2. Total Revenue

This query calculates the total revenue from the ticket\_booking table for flight code "f1-10" by summing the amount values, casting them as decimal to ensure precise calculation.

**Query:**

```
SELECT SUM(CAST(amount AS DECIMAL(10,2))) as totalRevenue
FROM ticket_booking
WHERE flightcode = "f1-10";
```

**Output:**

totalRevenue
6000.00

## PROCEDURES:

1. **Add New Flight:** This procedure inserts a new flight record into the manageflight table using the provided flight details and returns a success message.

**SQL Query:**

```
DELIMITER $$
CREATE DEFINER=`` PROCEDURE `AddNewFlight`(
    IN flightcode VARCHAR(6),
    IN source VARCHAR(30),
    IN destination VARCHAR(30),
    IN takeoff VARCHAR(30),
    IN noofseat BIGINT
)
BEGIN
    -- Insert the new flight into the manageflight table
```

```

INSERT INTO manageflight (flightcode, source, destination, takeoff, noofseat)
VALUES (flightcode, source, destination, takeoff, noofseat);

-- Return a success message
SELECT 'Flight added successfully!' AS message;
END$$
DELIMITER ;

```

**Output:**

2. **Calculate Total Revenue:** This procedure calculates and returns the total revenue for a specified flight based on ticket bookings, or a message if no bookings are found.

**SQL Query:**

```
DELIMITER $$
```

```

CREATE DEFINER=`` PROCEDURE `CalculateTotalRevenue`(
  IN flightCode VARCHAR(6)
)

```

```
BEGIN
```

```
  DECLARE totalRevenue DECIMAL(10,2);
```

```
  -- Calculate the total revenue by summing up the amount from the ticket_booking table
```

```
  SELECT SUM(CAST(amount AS DECIMAL(10,2))) INTO totalRevenue
```

```
  FROM ticket_booking
```

```
  WHERE flightcode = flightCode;
```

```

-- Return the total revenue
IF totalRevenue IS NOT NULL THEN
    SELECT CONCAT('Total Revenue for Flight ', flightCode, ' is: ', totalRevenue) AS
message;
ELSE
    SELECT 'No bookings found for this flight.' AS message;
END IF;
END$$
DELIMITER ;

```

**Output:**

The screenshot shows a web application window with the title "Calculate Total Revenue". Inside the window, there is a label "Flight Code:" followed by a text input field containing the value "f1-10". Below the input field is a button labeled "Calculate Revenue". Underneath the button, the text "Total Revenue for Flight f1-10 is: 8000.00" is displayed. At the bottom of the window, there is a "BACK" button.

3. **Get all passengers for a particular flight:** This procedure retrieves the passenger details (name, gender, nationality, and passport number) for all passengers booked on a specified flight.

**SQL Query:**

```

DELIMITER $$
CREATE DEFINER='' PROCEDURE `GetAllPassengersForFlight`(
    IN flightCode VARCHAR(6)
)
BEGIN
    -- Retrieve the passenger details for the given flight code
    SELECT p.name, p.gender, p.nationality, p.passportnumber
    FROM ticket_booking t
    JOIN managepassenger p ON t.passportnumber = p.passportnumber
    WHERE t.flightcode = flightCode;
END$$
DELIMITER ;

```

**Output:**

**Retrieve Passengers for a Flight**

**Flight Code:**

f1-10

**Retrieve Passengers**

Name: Michael Brown  
Gender:  
Nationality: British  
Passport Number: Y66778899

Name: Shwetha  
Gender: female  
Nationality: Indian  
Passport Number: 11111111

**BACK**

4. **Retrieve Flight info between cities:** This procedure retrieves flight information (flight code, source, destination, takeoff time, and number of seats) for flights between the specified source and destination cities.

**SQL Query:**

```
DELIMITER $$
CREATE DEFINER='' PROCEDURE `RetrieveFlightInfoBetweenCities`(
  IN source_city VARCHAR(30),
  IN destination_city VARCHAR(30)
)
BEGIN
  -- Retrieve flight information between the specified source and destination
  SELECT flightcode, source, destination, takeoff, noofseat
  FROM manageflight
  WHERE source = source_city AND destination = destination_city;
END$$
DELIMITER ;
```

**Output:**

Retrieve Flight Info Between Cities

Source City: Bengaluru

Destination City: Chennai

Retrieve Flights Back to Dashboard

Flight Code: F1-202  
 Source: Bengaluru  
 Destination: Chennai  
 Takeoff Time: 12/09/2024  
 Seats Available: 100

## TRIGGERS:

1.**Seat Count:** This trigger keeps track of the number of seats available per flight.

**Before Booking :**

flightcode	source	destination	takeoff	noofseat
f1-10	Mumbai	Hyderabad	08/11/2022	99
f1-101	Chennai	Bhopal	09/22/2000	501
f1-200	Singapore	Bengaluru	08/22/2005	499
F1-202	Bengaluru	Chennai	12/09/2024	100
F1-500	Delhi	Kolkata	09/12/2024	200
f1-900	CHintamani	Bengaluru	09/27/2000	50

## Query:

```
CREATE TRIGGER `after_ticket_insert`
AFTER INSERT ON `ticket_booking`
FOR EACH ROW BEGIN
  UPDATE manageflight
  SET noofseat = noofseat - 1
```



WHERE flightcode = NEW.flightcode;

END

After Booking:

The screenshot shows a web application titled "TICKET BOOKING" with a "<--BACK" link. The form contains the following fields: Passengerid (221), Passenger name (Krishna), Flight Code (f1-900), Gender (Male), Passportnumber (Y66778899), Amount (5000), and nationality (Indian). Below the form are buttons for "BOOK", "RESET", and "SEARCH". A modal message box is displayed in the center, stating "Data inserted successfully!" with an "OK" button. The background shows a table with columns: Passenger ID, PassengerN..., Amount, and Nationality.

flightcode	source	destination	takeoff	noofseat
f1-10	Mumbai	Hyderabad	08/11/2022	99
f1-101	Chennai	Bhopal	09/22/2000	501
f1-200	Singapore	Bengaluru	08/22/2005	499
F1-202	Bengaluru	Chennai	12/09/2024	99
F1-500	Delhi	Kolkata	09/12/2024	200
f1-900	CHintamani	Bengaluru	09/27/2000	49

2.Preventing Duplicate Flight entries:

Before:

flightcode	source	destination	takeoff	noofseat
f1-10	Mumbai	Hyderabad	08/11/2022	99
f1-101	Chennai	Bhopal	09/22/2000	501
f1-200	Singapore	Bengaluru	08/22/2005	499
F1-202	Bengaluru	Chennai	12/09/2024	99
F1-500	Delhi	Kolkata	09/12/2024	200
f1-900	CHintamani	Bengaluru	09/27/2000	49

#### Query:

```
CREATE TRIGGER `prevent_duplicate_flight_code`
BEFORE INSERT ON `manageflight`
FOR EACH ROW BEGIN
    -- Check if the flight code already exists
    IF EXISTS (SELECT 1 FROM manageflight WHERE flightcode = NEW.flightcode) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Flight code already exists!';
    END IF;
END
```

#### After :

The screenshot shows a web application titled "Manage Flight" with a "< - BACK" link. The form contains five input fields: "Flight Code" (f1-10), "Source" (Hubli), "Destination" (Bengaluru), "Take off" (05/09/2024), and "No of Seats" (100). Below the form are three buttons: "INSERT", "UPDATE", and "VIEW". A modal message box is displayed in the foreground with a blue information icon and the text "Flight code already exists!". The message box has an "OK" button at the bottom right.