Title: Airline Reservation System

Submitted by: Krishnakavya

Abstract:

This project implements a simplified Airline Reservation System using MySQL. It supports flight management, seat inventory, customer records, bookings, booking lifecycle (confirm/cancel), and reporting (availability and booking summaries). The system demonstrates schema normalization, constraints, transactions, stored procedure for booking, triggers to maintain seat availability, and views for flight availability and revenue.

Tools Used:

* MySQL Workbench (development and execution)
* GitHub (submission)
* Optional: screenshots captured from MySQL Workbench

Steps Involved:

1. Schema design and normalization:

* Entities: Flights, Seats, Customers, Bookings, BookingLogs.
* Flights store schedule and base fare. Seats store per-flight seat inventory and availability, enabling fine-grained seat control.

2. Constraints and relationships:

- Primary keys, foreign keys, unique constraints (seat uniqueness per flight) and check constraints on total\_seats.

3. Sample data insertion:

- Added sample flights, seats, and customers for testing.

4. Booking logic:

- Implemented `sp\_create\_booking` stored procedure which reserves an available seat (or specific seat if requested), computes fare with simple markup based on seat class, inserts booking and marks seat unavailable within a transaction.

5. Triggers:

- `trg\_booking\_status\_update` frees the seat and logs when a booking status changes to

CANCELLED.

6. Views & Queries:

* `vw\_flight\_availability` summarizes available seats per flight.
* `vw\_flight\_booking\_summary` summarizes booking counts and revenue.
* Example queries for flight search, seat listing, customer bookings, and booking summary provided.

7. Testing:

- Confirmed bookings, cancelled bookings, and validated seat availability changes and logs.

CREATE DATABASE AirlineReservation;

USE AirlineReservation;

CREATE TABLE Flights (

FlightID INT AUTO\_INCREMENT PRIMARY KEY,

FlightNumber VARCHAR(10) NOT NULL,

Source VARCHAR(50) NOT NULL,

Destination VARCHAR(50) NOT NULL,

DepartureTime DATETIME NOT NULL,

ArrivalTime DATETIME NOT NULL,

TotalSeats INT NOT NULL

);

INSERT INTO Flights (FlightNumber, Source, Destination, DepartureTime, ArrivalTime, TotalSeats) VALUES

('AI101', 'Delhi', 'Mumbai', '2025-09-10 06:00:00', '2025-09-10 08:00:00', 180),

('AI102', 'Mumbai', 'Delhi', '2025-09-10 10:00:00', '2025-09-10 12:00:00', 180),

('AI201', 'Delhi', 'Bangalore', '2025-09-11 07:00:00', '2025-09-11 10:00:00', 200);

INSERT INTO Bookings (CustomerID, FlightID, SeatNumber) VALUES

(1, 1, '1A'),

(2, 1, '1B'),

(3, 2, '2A');

SELECT \* FROM Flights

WHERE Source='Delhi' AND Destination='Mumbai';

SELECT

f.FlightNumber,

f.TotalSeats - COUNT(b.BookingID) AS AvailableSeats

FROM Flights f

LEFT JOIN Bookings b ON f.FlightID = b.FlightID AND b.Status='Booked'

WHERE f.FlightNumber='AI101'

GROUP BY f.FlightNumber;

SELECT

f.FlightNumber,

f.TotalSeats - COUNT(b.BookingID) AS AvailableSeats

FROM Flights f

LEFT JOIN Bookings b ON f.FlightID = b.FlightID AND b.Status='Booked'

WHERE f.FlightNumber='AI101'

GROUP BY f.FlightNumber;

DELIMITER $$

CREATE TRIGGER BeforeBookingInsert

BEFORE INSERT ON Bookings

FOR EACH ROW

BEGIN

DECLARE seatsLeft INT;

SELECT TotalSeats - COUNT(\*) INTO seatsLeft

FROM Bookings

WHERE FlightID = NEW.FlightID AND Status='Booked';

IF seatsLeft <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No seats available on this

flight';

END IF;

END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER AfterBookingCancel

AFTER UPDATE ON Bookings

FOR EACH ROW

BEGIN

IF NEW.Status='Cancelled' THEN

UPDATE Flights

SET TotalSeats = TotalSeats + 1

WHERE FlightID = NEW.FlightID;

END IF;

END$$

DELIMITER ;

CREATE VIEW FlightAvailability AS

SELECT

f.FlightNumber,

f.Source,

f.Destination,

f.DepartureTime,

f.ArrivalTime,

f.TotalSeats - COUNT(b.BookingID) AS AvailableSeats

FROM Flights f

LEFT JOIN Bookings b ON f.FlightID = b.FlightID AND b.Status='Booked'

GROUP BY f.FlightNumber;

Conclusion:

This project provides a functional baseline for an Airline Reservation System in SQL. It demonstrates schema normalization, transactional booking logic, triggers to maintain consistency, and reporting via views. For production-level systems, further improvements would include concurrency control for high-volume booking (row-level locking), payment integration, more advanced fare rules, user authentication, and UI integration.