**Airline Reservation System: SQL Project Report**

**Abstract**

This project established a robust, relational database for a hypothetical Airline Reservation System using Structured Query Language (SQL). The core objective was to create a normalized, transaction-safe schema capable of managing flight inventory, customer data, and real-time booking status. The solution leverages relational constraints, advanced views for simplified data access, and triggers to enforce business logic automatically, ensuring data integrity across complex operations like booking confirmation and cancellation. The resulting database structure is scalable and provides instantaneous access to critical information, such as available seats and flight search results.

**Introduction**

The primary challenge in an airline reservation system is maintaining an accurate, realtime count of available seats while handling concurrent booking and cancellation transactions. This project focused on building the foundational database layer for such a system. Key tables were designed to separate static data (Airports, Aircrafts, Seats) from dynamic transactional data (Flights, Customers, Bookings, BookingDetails). The design prioritizes third normal form (3NF) to minimize redundancy and enhance transactional performance.

**Tools Used**

* **Database System:** MySQL (Relational Database Management System).
* **Development Environment:** MySQL Workbench (for design, execution, and debugging).
* **Language:** Structured Query Language (SQL), utilizing Data Definition Language (DDL), Data Manipulation Language (DML), and Transaction Control Language (TCL) features (specifically triggers and views).

**Steps Involved in Building the Project**

The project was executed through six primary phases, detailed below:

1. Schema Design and Normalization

The initial step involved defining the core tables and their relationships. The schema was normalized to separate booking details from seat pricing logic. Key tables and relationships include:

◦ **Aircrafts & Seats:** A one-to-many relationship where Seats defines the fixed layout, class, and price multiplier for each aircraft model.

◦ **Flights:** Links an aircraft to departure/arrival airports and establishes the base\_price. Includes an optimized available\_seats column.

◦ **Bookings & BookingDetails:** A many-to-many relationship between Bookings and the specific seat\_id selected, ensuring accurate calculation and linking.

2. Data Insertion (DML)

Sample data for customers, airports (e.g., JFK, LAX), aircrafts, and initial flights were inserted. A complex DML statement was used to generate seat entries for a Boeing 737-800, setting BUSINESS class seats with a ×2*.*00 price multiplier and ECONOMY seats with ×1*.*00.

3. Trigger Implementation

Three crucial triggers were developed to maintain data consistency in real-time:

◦ **TR\_Set\_Initial\_Seats\_Before\_Insert:** Automatically populates the available\_seats column on the Flights table with the aircraft’s total\_seats when a new flight is created.

◦ **TR\_Update\_Seats\_And\_Price\_After\_Detail\_Insert:** Updates the total\_price in the Bookings table and decrements available\_seats on the Flights table only if the booking status is CONFIRMED.

◦ **TR\_Update\_Seats\_After\_Booking\_Status\_Update:** Handles updates to the status column, incrementing or decrementing available\_seats based on transitions (e.g., CONFIRMED →CANCELLED increases the count).

4. Flight Availability Views

Two complex views were created to simplify application-level queries:

◦ **V\_FlightSearch:** Provides a summary of flights, including IATA codes, cities, dates, base price, and the current available\_seats count.

◦ **V\_AvailableSeats:** The core availability check. It returns a list of specific seat numbers and their calculated prices for a given flight by excluding all seat\_ids currently linked to CONFIRMED bookings.

5. Query Development

A final phase included essential operational queries, such as searching for specific routes/dates and generating a comprehensive booking summary report for a customer.

**Conclusion**

The project successfully delivered a robust and efficient SQL database structure for an airline reservation system. By prioritizing normalization and utilizing server-side business logic via triggers, the system ensures that the most critical metric—seat availability—is always accurate, even under heavy transactional load. The defined views abstract complex join logic, making it easy for external applications to query flight information and available seating. This foundation is fully prepared for integration into a front-end booking application.