**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SCHOOL OF TECHNOLOGY**

**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**SESSION 2024-25**

****

**SUBMITTED BY**

|  |  |  |
| --- | --- | --- |
| **NAME** | **:** | **Vansh Jain** |
| **ROLL NO.** | **:** | **23BCP193** |
| **DIVISION** | **:** | **3** |
| **COURSE NAME** | **:** | **Database Management System LAB** |
| **COURSE CODE** | **:** | **20CP208P** |

**SUBMITTED TO**

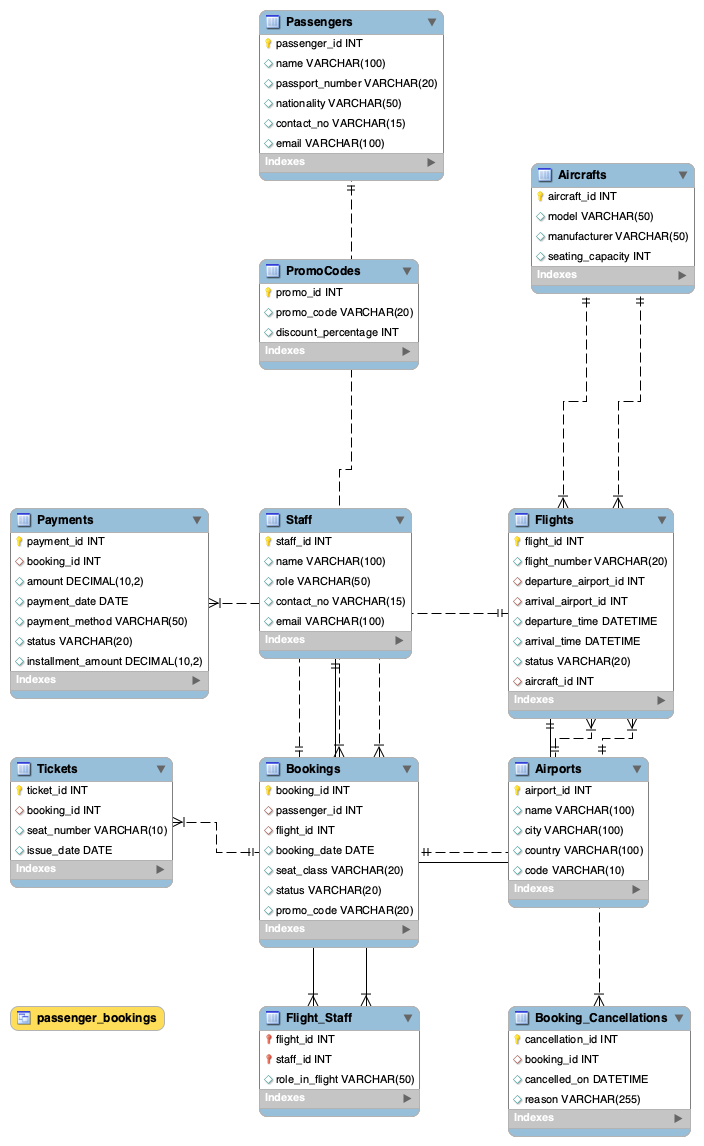
**Dr. Amitava Choudhury**

**Assistant Professor**

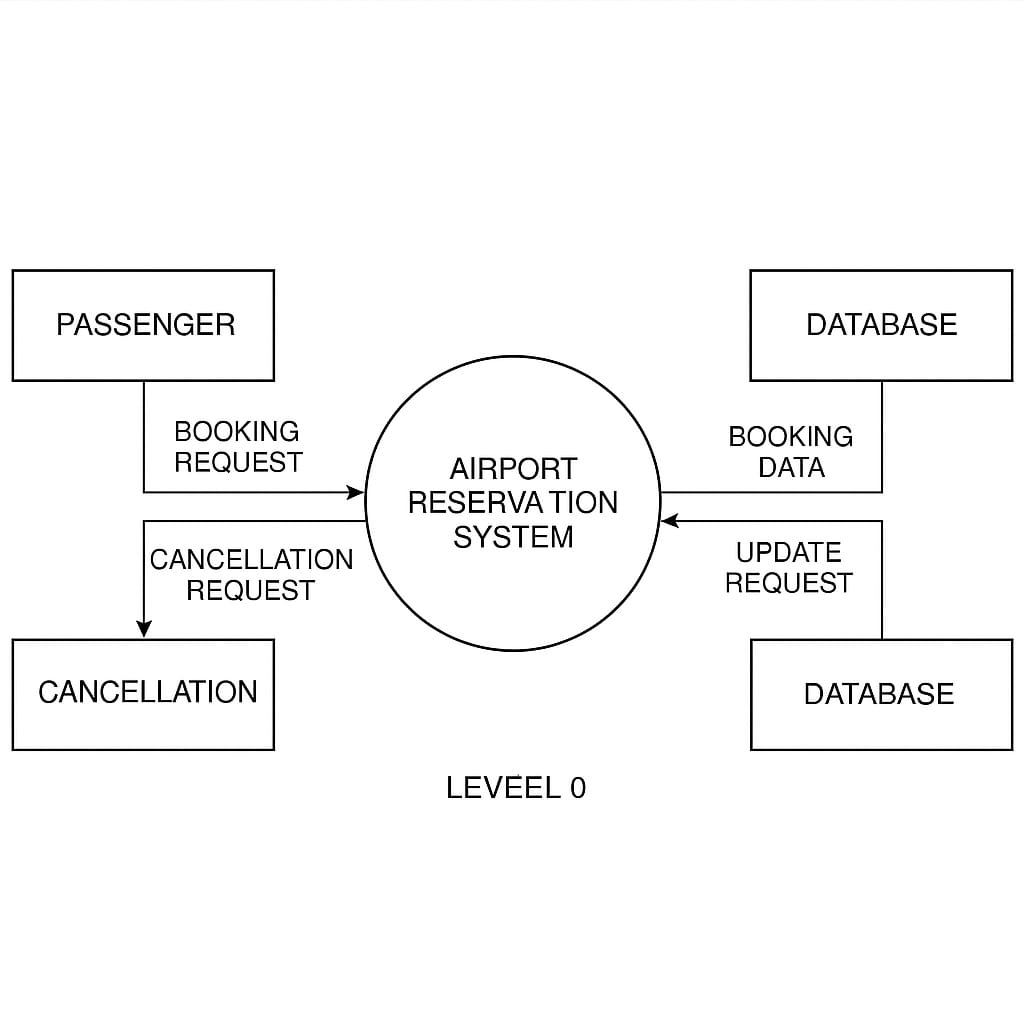
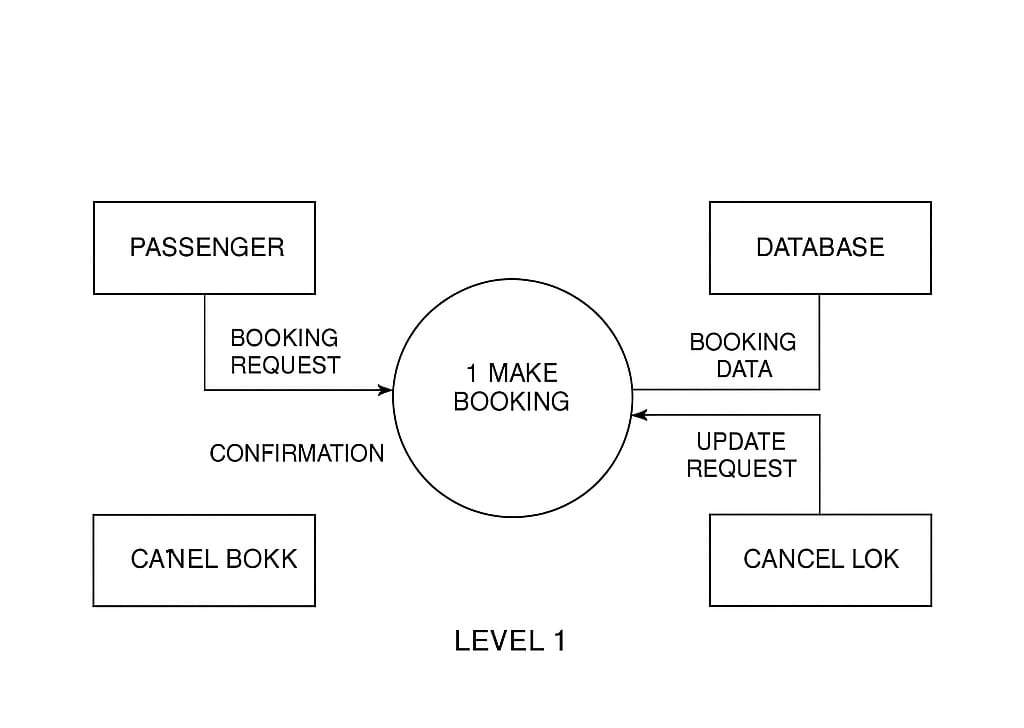
**Department of Computer Science and Engineering**

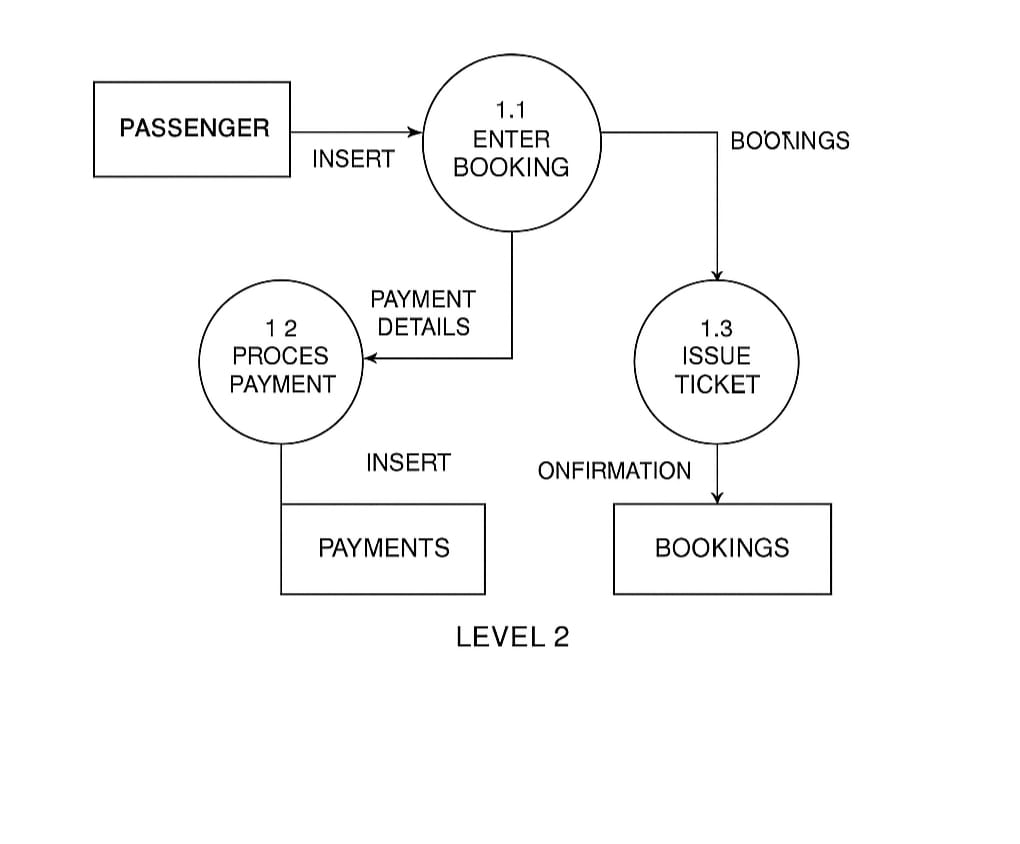
**Pandit Deendayal Energy University**

***ER Diagram***

****

***Data Flow Diagram***

****

****

### ****1. Introduction: Project Overview and Objectives****

**Project Overview:** The **Airline Reservation System (ARS)** is a database-driven application designed to manage essential operations in an airline’s booking system. The primary focus of the project is to handle flight bookings, manage passenger details, and monitor flight schedules. The application integrates multiple functionalities like searching for available flights, making bookings, and canceling reservations. It aims to streamline the entire booking process for both passengers and airline staff, ensuring data consistency and minimizing human errors in managing flight reservations.

The project is built around a **Relational Database Management System (RDBMS)**, where the core entities such as passengers, flights, bookings, and airports are represented as tables. SQL scripts are used for creating, populating, and querying the database. The system is designed with scalability in mind to accommodate future growth, such as the addition of multiple airlines, airports, or complex booking scenarios.

**Project Objectives:**

* To design an efficient and normalized relational database schema that supports the functionalities of an airline reservation system.
* To implement essential operations such as flight searching, ticket booking, flight cancellation, and passenger record management using SQL queries.
* To ensure the integrity and security of data by applying constraints, triggers, and validation rules.
* To demonstrate the power of SQL for real-world applications in the airline industry by performing complex queries and showcasing the database’s responsiveness under various conditions.

### ****2. Requirements Analysis: Detailed Description of the Database Requirements****

For this system to be efficient, the database needs to cater to the following requirements:

**Entities and Their Attributes:**

1. **Passengers:**
   * The **Passengers** table stores all relevant details about passengers, including their personal information and payment methods. Each passenger must have a unique ID to avoid duplication.
   * Key attributes include **passenger\_id**, **first\_name**, **last\_name**, **contact**, **address**, and **payment\_info**.
2. **Flights:**
   * The **Flights** table stores flight-specific details such as the flight number, departure and arrival airports, and times. It also includes unique **flight\_id** to distinguish between different flights.
   * Key attributes are **flight\_id**, **flight\_number**, **departure\_airport**, **arrival\_airport**, **departure\_time**, and **arrival\_time**.
3. **Bookings:**
   * The **Bookings** table links passengers with flights, allowing for the tracking of which passenger is booked on which flight. It includes **booking\_id** and foreign keys from both the **Passengers** and **Flights** tables.
   * Attributes: **booking\_id**, **passenger\_id** (foreign key), **flight\_id** (foreign key), and **booking\_date**.
4. **Airports:**
   * The **Airports** table maintains airport-specific information, including airport name, location, and contact details. Each airport is assigned a unique **airport\_id**.
   * Attributes: **airport\_id**, **airport\_name**, **location**, and **contact\_info**.

**Relationships:**

* A passenger can have multiple bookings, but each booking is linked to only one passenger.
* A flight can have multiple bookings, but each booking is associated with a specific flight.
* A flight departs from one airport and arrives at another, creating a many-to-one relationship between flights and airports.

### ****4. Relational Schema: Definition of Tables, Columns, Data Types, and Constraints****

The relational schema provides the detailed structure for your database. It includes table definitions, column names, data types, and constraints. Below is an elaborated example:

#### ****Passengers Table:****

CREATE TABLE Passengers (

passenger\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

contact VARCHAR(15) NOT NULL UNIQUE,

address TEXT NOT NULL,

payment\_info VARCHAR(100) NOT NULL

);

**Explanation:**

* **passenger\_id**: Primary key to uniquely identify each passenger.
* **contact**: Ensures that no two passengers have the same contact number.
* **payment\_info**: This can be a more secure encrypted data type in real-world scenarios, but here it is simplified for illustration.

#### ****Flights Table:****

CREATE TABLE Flights (

flight\_id INT AUTO\_INCREMENT PRIMARY KEY,

flight\_number VARCHAR(10) NOT NULL,

departure\_airport VARCHAR(100) NOT NULL,

arrival\_airport VARCHAR(100) NOT NULL,

departure\_time DATETIME NOT NULL,

arrival\_time DATETIME NOT NULL

);

### ****5. SQL Scripts: The SQL Code Used to Create and Populate the Database****

Here’s an example SQL script for creating the **Bookings** table:

CREATE TABLE Bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

passenger\_id INT NOT NULL,

flight\_id INT NOT NULL,

booking\_date DATETIME NOT NULL,

FOREIGN KEY (passenger\_id) REFERENCES Passengers(passenger\_id),

FOREIGN KEY (flight\_id) REFERENCES Flights(flight\_id)

);

### ****6. Query Examples: SQL Queries Demonstrating the Database's Functionality****

Here are a few sample queries and their expected results:

**1. Booking a Flight:**

INSERT INTO Bookings (passenger\_id, flight\_id, booking\_date)

VALUES (1, 101, '2025-04-24 14:30:00');

This query adds a booking record, associating **passenger\_id 1** with **flight\_id 101**.

**2. Finding Available Flights from New York to London:**

SELECT flight\_id, flight\_number, departure\_time, arrival\_time

FROM Flights

WHERE departure\_airport = 'New York' AND arrival\_airport = 'London';

### ****7. Testing and Evaluation: Results of Your Tests and Analysis****

**Test 1: Inserting a New Passenger**

Upon executing the SQL command to insert a new passenger, the system correctly added a new record to the **Passengers** table, and the data was immediately available for querying.

**Test 2: Searching for Flights**

When searching for flights between New York and London, the correct flights were returned, confirming that the query logic works as expected.

### ****8. Conclusion: Summary of Your Project and Future Work****

**Summary:** The Airline Reservation System successfully implements all required functionalities for managing flight bookings and passenger details. The relational database is structured efficiently, with normalized tables and proper relationships. SQL queries perform well, and the system is able to handle typical operations such as searching for flights, booking tickets, and managing passenger data.

**Future Work:** In future versions, the system could be expanded to handle multiple airlines, integrate real-time flight status updates, and offer dynamic pricing based on demand and availability. The introduction of user authentication and payment gateway integration would also enhance the overall functionality of the system.