# AIRLINE SERVICE SYSTEM

## A MINI PROJECT REPORT

**Submitted by**

| | |
|---|---|
| **KRITHIKA B** | **220701137** |
| **OVIYA G** | **220701192** |

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023 - 24

# BONAFIDE CERTIFICATE

Certified that this project report " **AIRLINE SERVICE SYSTEM** " is the bonafide work of **" KRITHIKA B (220701137) , OVIYA G (220701192) "** who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE**

**Dr.R.SABITHA**<br>
**Professor and II Year Academic Head**<br>
**Computer Science and Engineering,**<br>
**Rajalakshmi Engineering College**<br>
**(Autonomus)**<br>
**Thandalam, Chennai - 602 105**

**SIGNATURE**

**Mrs.D.KALPANA**<br>
**Assistant Professor ,**<br>
**Computer Science and Engineering,**<br>
**Rajalakshmi Engineering College,**<br>
**(Autonomous),**<br>
**Thandalam, Chennai - 602 105**

# ABSTRACT

The Airline Service System is a comprehensive solution designed to streamline and enhance the operations of airline services. This system integrates various functionalities such as flight availability display, ticket booking, and ticket status checking into a single, user-friendly interface.

Built with Python, MySQL, and Tkinter, the system offers robust backend database management and a visually appealing frontend. The system connects seamlessly to a MySQL database, storing and retrieving data efficiently to provide real-time information to users.

The user interface, developed with Tkinter, ensures a smooth user experience, allowing customers to navigate through the system effortlessly. Users can view available flights, book tickets, and check their ticket status with just a few clicks.

# TABLE OF CONTENTS

# CHAPTER 1
# INTRODUCTION

## 1.1 INTRODUCTION

In the Airline Service System, users can perform fundamental airline management operations such as booking flights, checking flight status, and viewing their booking details. Each flight in the system has a unique identification number. The user books a flight by entering the flight ID and their personal details. Each user can book multiple flights, but each booking is treated individually. When a user books a flight, the system updates the flight’s availability status. The record of the booked flight with user details can also be viewed at any time.

This system is designed to streamline the process of flight booking and management, making it easier for both the airline and the passengers. It ensures a smooth and efficient operation, enhancing the overall user experience.

## 1.2 EXISISTING SYSTEM

The existing system for airline services is typically managed through a software suite known as a Passenger Service System (PSS). The PSS supports all transactions between carriers and their customers, from ticket reservations to boarding. It’s a complex structure, combining dozens of tools and applications that automate a wide range of passenger-related activities.

At its core, an airline or central reservation system serves as a database for flight schedules, available seats, fares and rules for each booking class, and passenger profiles. Apart from storing flight-related information, its major functions include reservations, ticketing, check-ins, online and mobile bookings, punctuality management, administration of loyalty programs, and passenger assistance.

The key components of a standard PSS include an airline reservations system, an airline inventory system, and a departure control system (DCS). These systems have evolved from basic Computerized Reservation Systems (CRS) into sophisticated software that seamlessly integrates with Global Distribution Systems (GDSs), enhancing their functionality.

However, these systems are not without their challenges. Failures in the PSS can cost airlines tens of millions of dollars in lost revenue. Moreover, the current state of aviation IT is often fragile, caused by different factors, from aging technologies to poor communication between different components to the introduction of immature solutions.

In conclusion, while the existing systems have served the airline industry well for many years, there is a need for more modern, efficient, and reliable solutions. This is where the proposed Airline Service System comes in, aiming to revolutionize the way airlines operate, offering improved operational efficiency, enhanced customer experience, and increased profitability.

## 1.3 PROPOSED SYSTEM

The proposed Airline Service System is a comprehensive solution designed to streamline and enhance the operations of airline services. It integrates various functionalities such as flight availability display, ticket booking, and ticket status checking into a single, user-friendly interface.

Built with Python, MySQL, and Tkinter, the system offers robust backend database management and a visually appealing frontend. The system connects seamlessly to a MySQL database, storing and retrieving data efficiently to provide real-time information to users.

The user interface, developed with Tkinter, ensures a smooth user experience, allowing customers to navigate through the system effortlessly. Users can view available flights, book tickets, and check their ticket status with just a few clicks.

The proposed system aims to revolutionize the way airlines operate, offering improved operational efficiency, enhanced customer experience, and increased profitability. It is a step towards the future of airline service management, where technology and convenience go hand in hand.

This project is a testament to the potential of integrating various technologies to create a solution that is greater than the sum of its parts. It showcases the power of Python, MySQL, and Tkinter in developing practical, real-world applications.

## 1.4 OBJECTIVES

1. **Streamline Operations:** The system aims to streamline the operations of airline services, making it easier for both the airline and the passengers to manage bookings.

2. **Enhance User Experience:** By integrating various functionalities into a single platform, the system seeks to enhance the user experience. Users can view flight availability, book tickets, and check ticket status, all from a user-friendly interface.

3. **Improve Efficiency:** With a robust MySQL database at its core, the system is designed to improve efficiency by providing real-time updates to users and managing data effectively.

4. **Increase Accessibility:** By making air travel booking a hassle-free process, the system aims to increase the accessibility of air travel to a wider audience.

5. **Showcase Technological Integration:** The project also serves as a testament to the potential of integrating various technologies (Python, MySQL, Tkinter) to create a solution that is greater than the sum of its parts.

## 1.5 MODULES

1. **User Registration and Authentication Module:** This module handles user registration and login. It ensures that only registered and authenticated users can book flights.

2. **Flight Management Module:** This module manages all the flight-related information. It includes functionalities such as adding new flights, updating flight schedules, and managing flight availability.

3. **Booking Module:** This module allows users to book flights. It includes selecting a flight, providing passenger details, and confirming the booking.

4. **Ticket Management Module:** This module manages the tickets booked by the users. It includes functionalities such as viewing ticket details, canceling bookings, and checking ticket status.

5. **User Interface Module:** Developed using Tkinter, this module handles the presentation layer of the system. It ensures a smooth and intuitive user experience.

6. **Database Management Module:** This module, powered by MySQL, handles all the data storage and retrieval operations. It ensures that data is stored efficiently and can be retrieved in real-time.

# CHAPTER 2
# SURVEY OF TECHNOLOGIES

## 2.1 SOFTWARE DESCRIPTION

1. **Functionality:** The system integrates various functionalities into a single platform. It allows users to view flight availability, book tickets, and check ticket status. Each flight in the system has a unique identification number, and each user can book multiple flights, with each booking treated individually.

2. **Compatibility:** The system is designed to be compatible with various operating systems. However, the actual compatibility may depend on the specific Python, MySQL, and Tkinter versions used in development.

3. **Platform Compatibility**: The system should be compatible with various operating systems. However, the actual compatibility may depend on the specific Python, MySQL, and Tkinter versions used in development.

4. **User-Friendly Interface**: The system should have an intuitive and easy-to-use interface, ensuring a smooth user experience. The interface should be designed keeping in mind the end-users and their tech-savviness.

5. **Performance:** The system should be able to handle a large number of users simultaneously without any degradation in performance. It should provide quick response times for all functionalities.

6. **Security:** The system should ensure the security of user data. It should implement appropriate security measures to prevent unauthorized access and data breaches.

7. **Reliability:** The system should be reliable and should function correctly and consistently under the defined conditions.

8. **Scalability:** The system should be scalable. It should be able to handle an increase in users and data without a significant impact on performance.

9. **Maintainability:** The system should be easy to maintain. It should be designed in a way that allows for easy updates and bug fixes.

10. **Data Integrity:** The system should ensure the accuracy and consistency of data. It should implement checks to prevent and correct any inconsistencies in the data.

11. **Documentation:** Adequate documentation should be provided for the system. This includes user manuals, system documentation, and developer guides.

## 2.2 LANGUAGE

### 2.2.1 MYSQL

The system uses MySQL for database management. MySQL is a popular open-source relational database management system (RDBMS) that is highly reliable and efficient. It stores and manages all the necessary data, including flight schedules, ticket bookings, and customer information.

### 2.2.2 PYTHON

The system is developed using Python, a high-level, interpreted programming language known for its simplicity and versatility. Python's extensive library support makes it an excellent choice for developing complex applications like the Airline Service System.

The user interface of the system is developed using Tkinter, a standard Python interface to the Tk GUI toolkit. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit, allowing the creation of visually appealing and user-friendly interfaces.

# CHAPTER 3
# REQUIREMENTS AND ANALYSIS

## 3.1 REQUIREMENT SPECIFICATION

1. **Platform Compatibility:** The system should be compatible with various operating systems. However, the actual compatibility may depend on the specific Python, MySQL, and Tkinter versions used in development.

2. **User-Friendly Interface:** The system should have an intuitive and easy-to-use interface, ensuring a smooth user experience. The interface should be designed keeping in mind the end-users and their tech-savviness.

3. **Performance:** The system should be able to handle a large number of users simultaneously without any degradation in performance. It should provide quick response times for all functionalities.

4. **Security:** The system should ensure the security of user data. It should implement appropriate security measures to prevent unauthorized access and data breaches.

5. **Reliability:** The system should be reliable and should function correctly and consistently under the defined conditions.

6. **Scalability:** The system should be scalable. It should be able to handle an increase in users and data without a significant impact on performance.

7. **Maintainability:** The system should be easy to maintain. It should be designed in a way that allows for easy updates and bug fixes.

8. **Data Integrity:** The system should ensure the accuracy and consistency of data. It should implement checks to prevent and correct any inconsistencies in the data.

9. **Documentation:** Adequate documentation should be provided for the system. This includes user manuals, system documentation, and developer guides.

## 3.2 HARDWARE AND SOFTWARE REQUIREMENTS

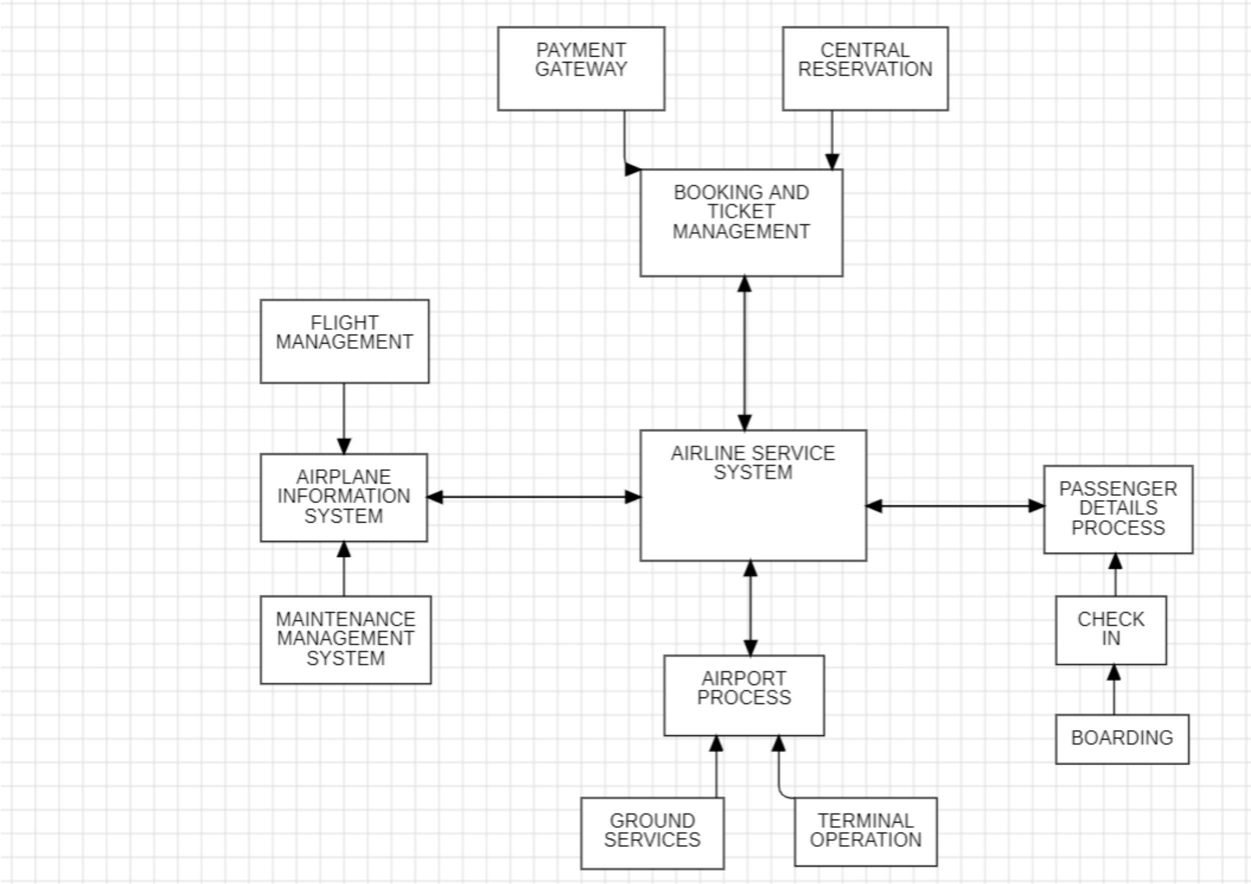
1. Hardware requirements

    1. Processor: A modern multi-core processor for efficient execution of the system.
    2. Memory: Sufficient RAM (at least 4GB) to ensure smooth operation of the system.
    3. Storage: Adequate hard disk space (at least 1GB) for storing the system files, database, and other data.
    4. Network: A stable internet connection for real-time updates and communication with the database.

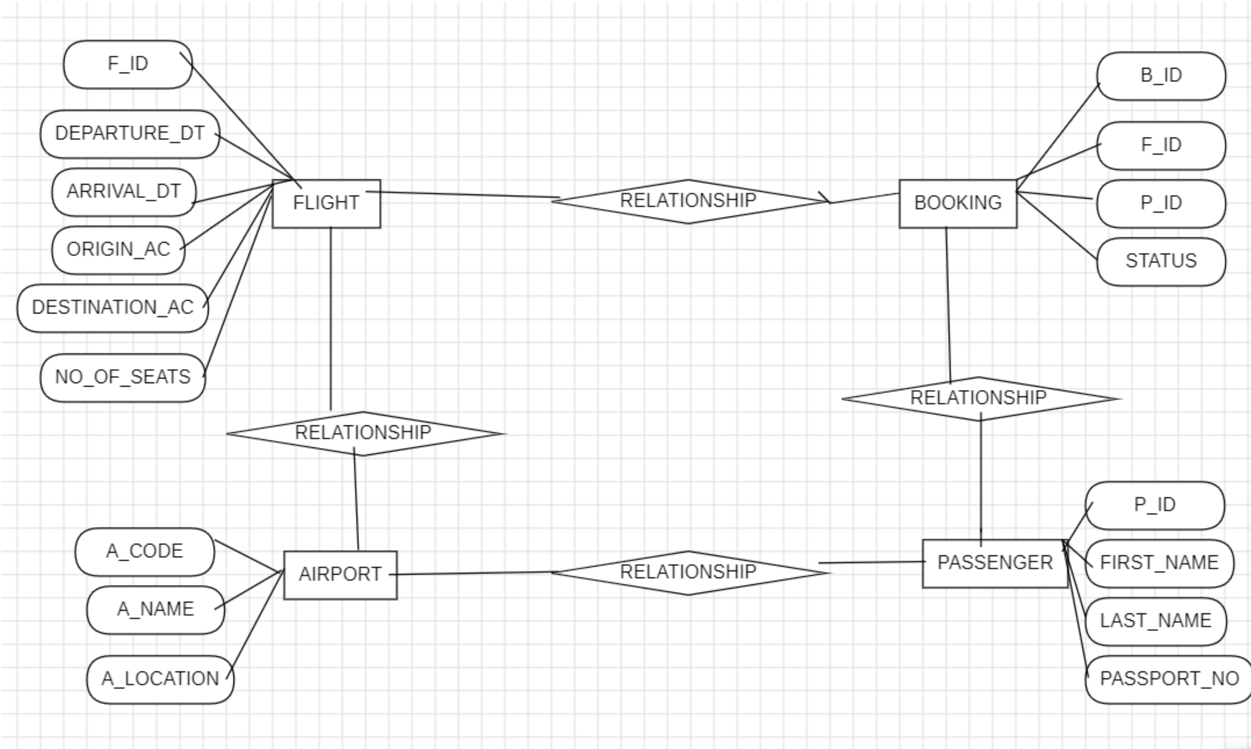
2. Software requirements

    1. Operating System: The system should be compatible with various operating systems. However, the actual compatibility may depend on the specific Python, MySQL, and Tkinter versions used in development.
    2. Python: The system is developed using Python, so a compatible version of Python (preferably the latest stable release) should be installed.
    3. MySQL: A compatible version of MySQL should be installed for database management.
    4. Tkinter: Tkinter, a standard Python interface to the Tk GUI toolkit, is used for developing the user interface. It comes pre-installed with Python, so no separate installation is required.
    5. Text Editor/IDE: A text editor or Integrated Development Environment (IDE) like Visual Studio Code, PyCharm, or Sublime Text for writing and managing the code.

## 3.3ARCHITECTURE DIAGRAM



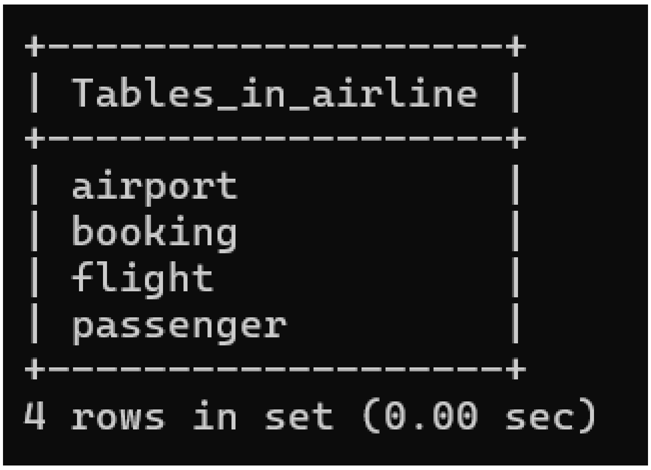
## 3.4ER DIAGRAM



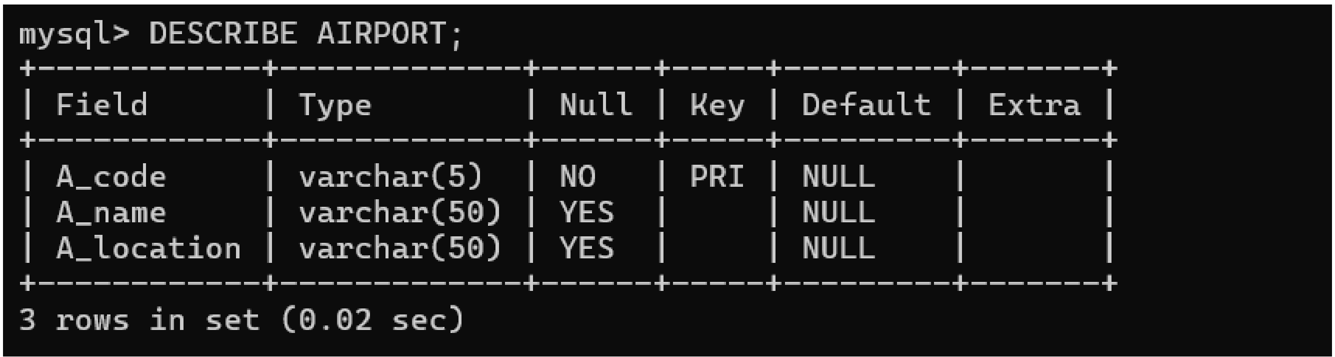
# CHAPTER 4

# PROGRAM CODE

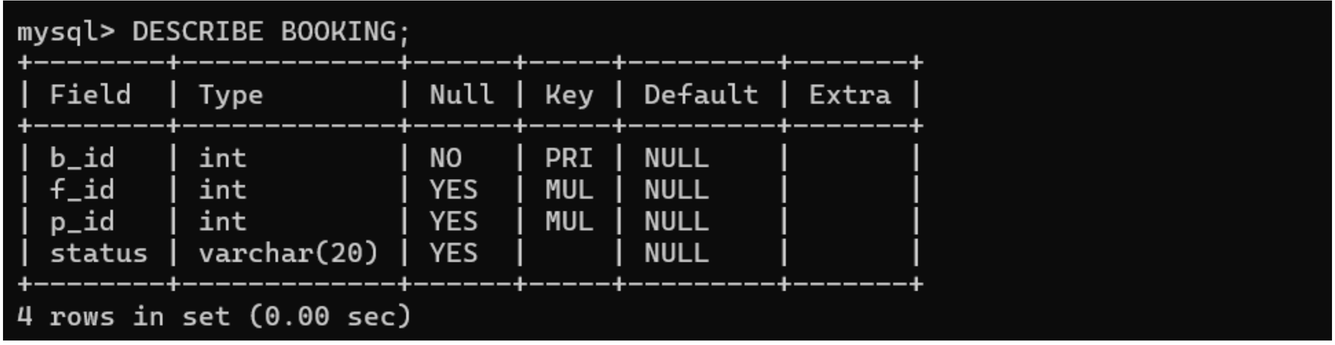
BACKEND



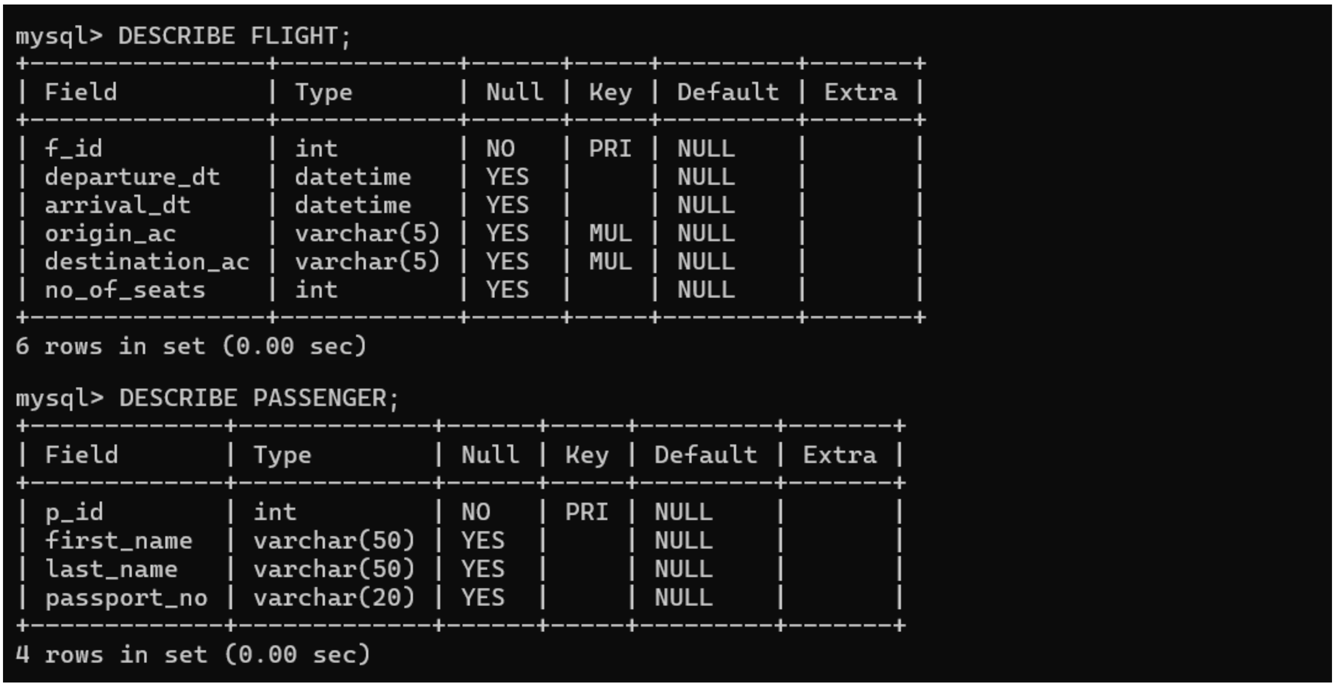
```
+-------------------+
| Tables_in_airline |
+-------------------+
| airport           |
| booking           |
| flight            |
| passenger         |
+-------------------+
4 rows in set (0.00 sec)
```



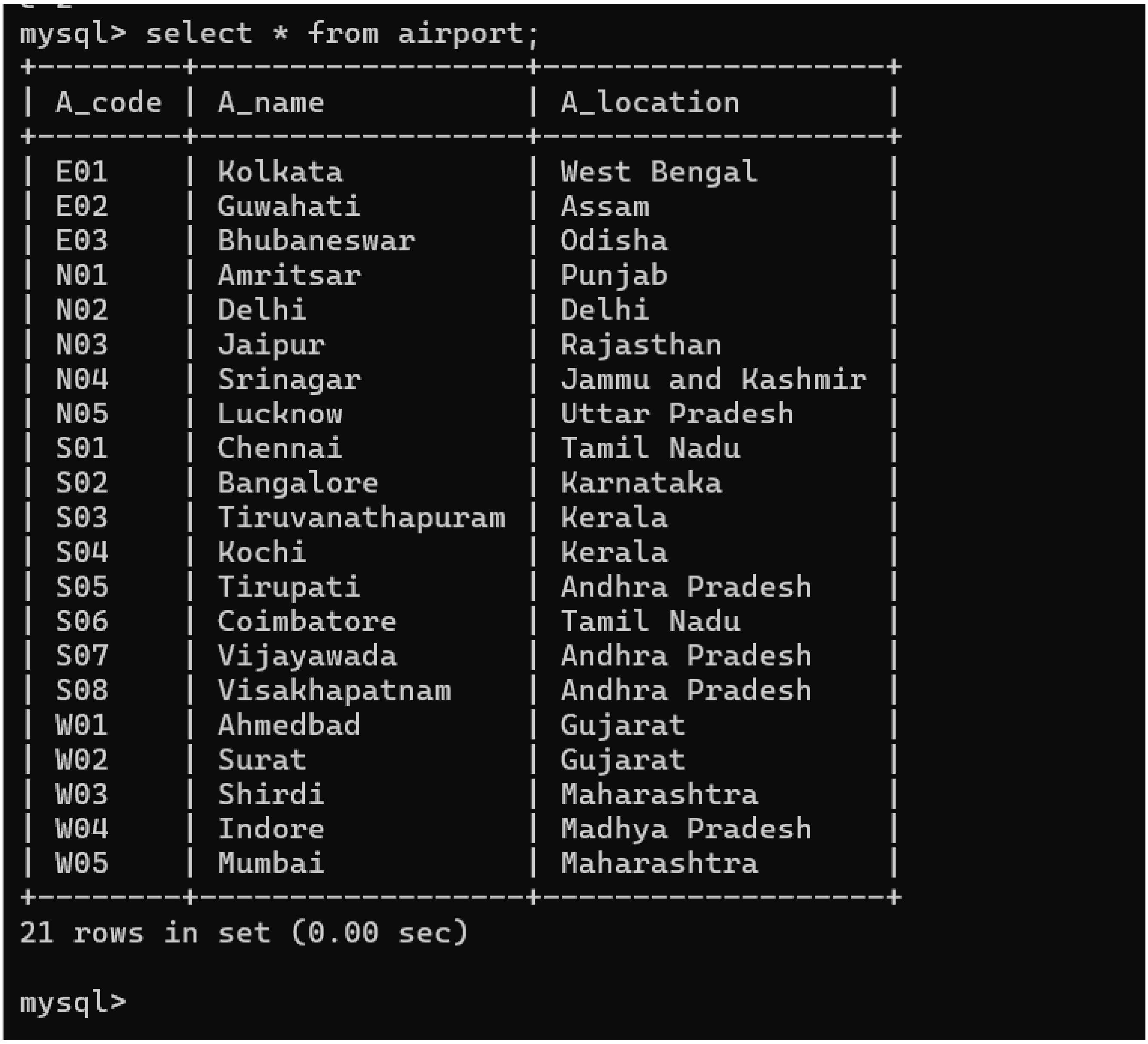
```
mysql> DESCRIBE AIRPORT;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| A_code     | varchar(5)  | NO   | PRI | NULL    |       |
| A_name     | varchar(50) | YES  |     | NULL    |       |
| A_location | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```



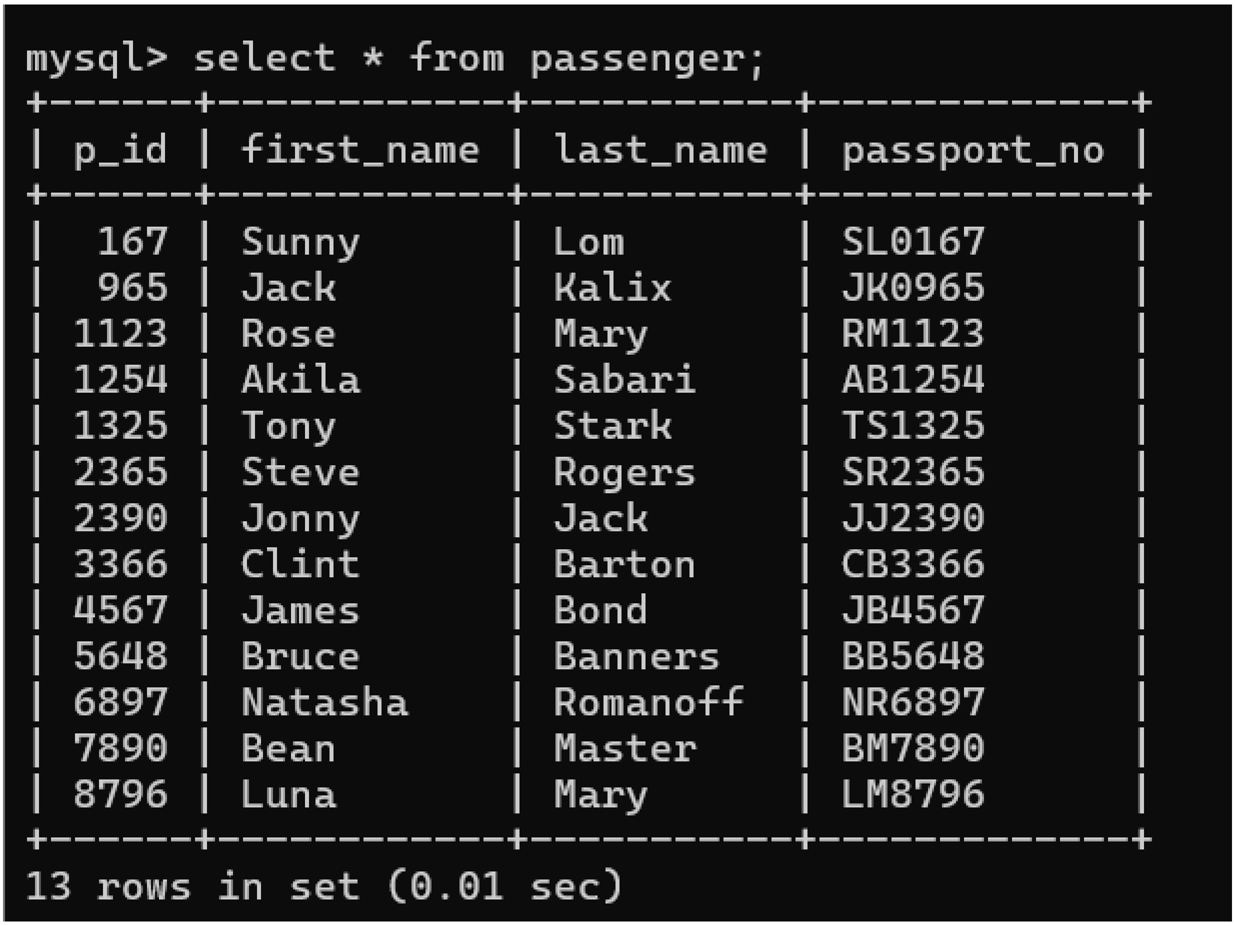
```
mysql> DESCRIBE BOOKING;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| b_id   | int         | NO   | PRI | NULL    |       |
| f_id   | int         | YES  | MUL | NULL    |       |
| p_id   | int         | YES  | MUL | NULL    |       |
| status | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



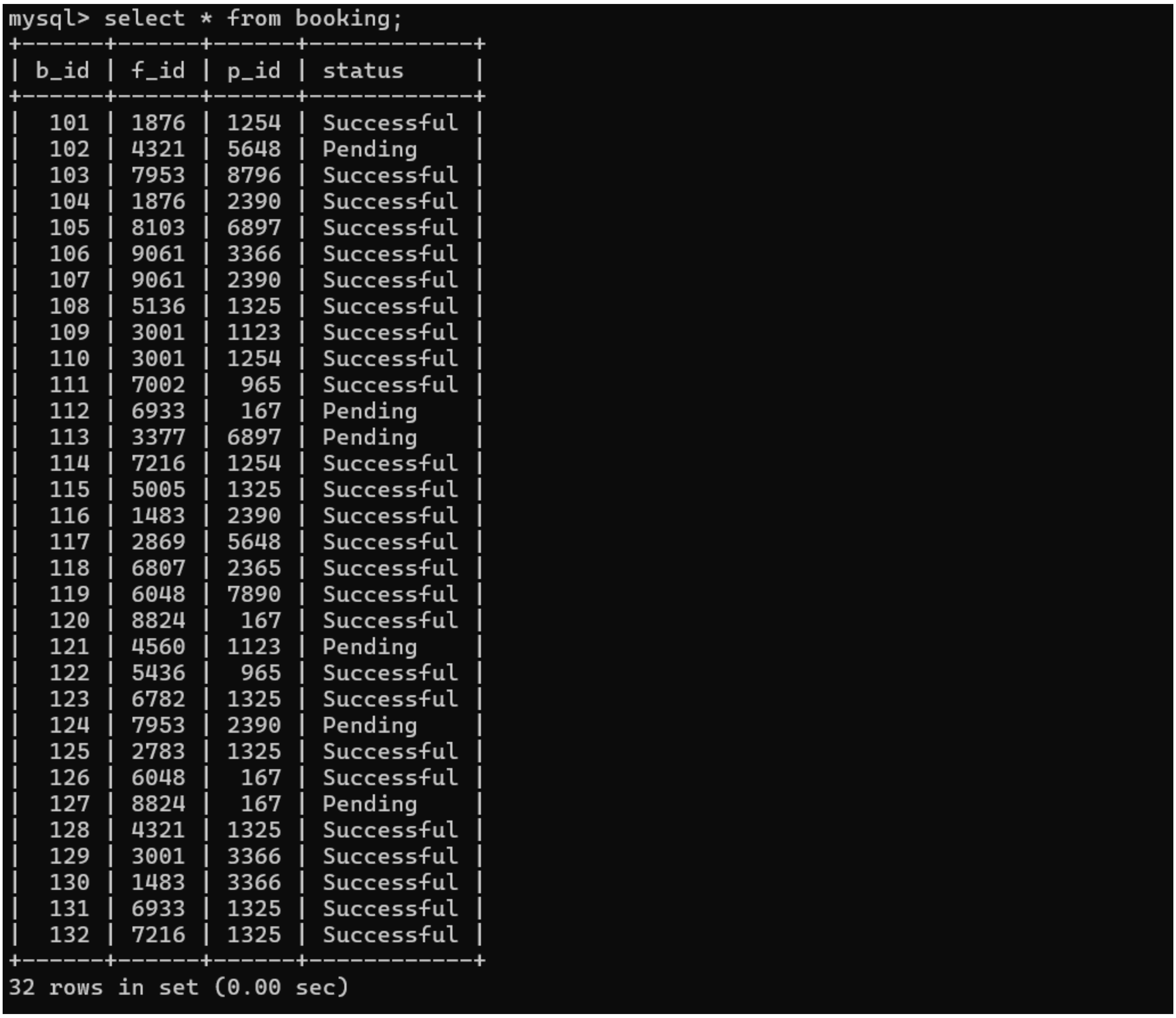
```
mysql> DESCRIBE FLIGHT;
+-----------------+------------+------+-----+---------+-------+
| Field           | Type       | Null | Key | Default | Extra |
+-----------------+------------+------+-----+---------+-------+
| f_id            | int        | NO   | PRI | NULL    |       |
| departure_dt    | datetime   | YES  |     | NULL    |       |
| arrival_dt      | datetime   | YES  |     | NULL    |       |
| origin_ac       | varchar(5) | YES  | MUL | NULL    |       |
| destination_ac  | varchar(5) | YES  | MUL | NULL    |       |
| no_of_seats     | int        | YES  |     | NULL    |       |
+-----------------+------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> DESCRIBE PASSENGER;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| p_id        | int         | NO   | PRI | NULL    |       |
| first_name  | varchar(50) | YES  |     | NULL    |       |
| last_name   | varchar(50) | YES  |     | NULL    |       |
| passport_no | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



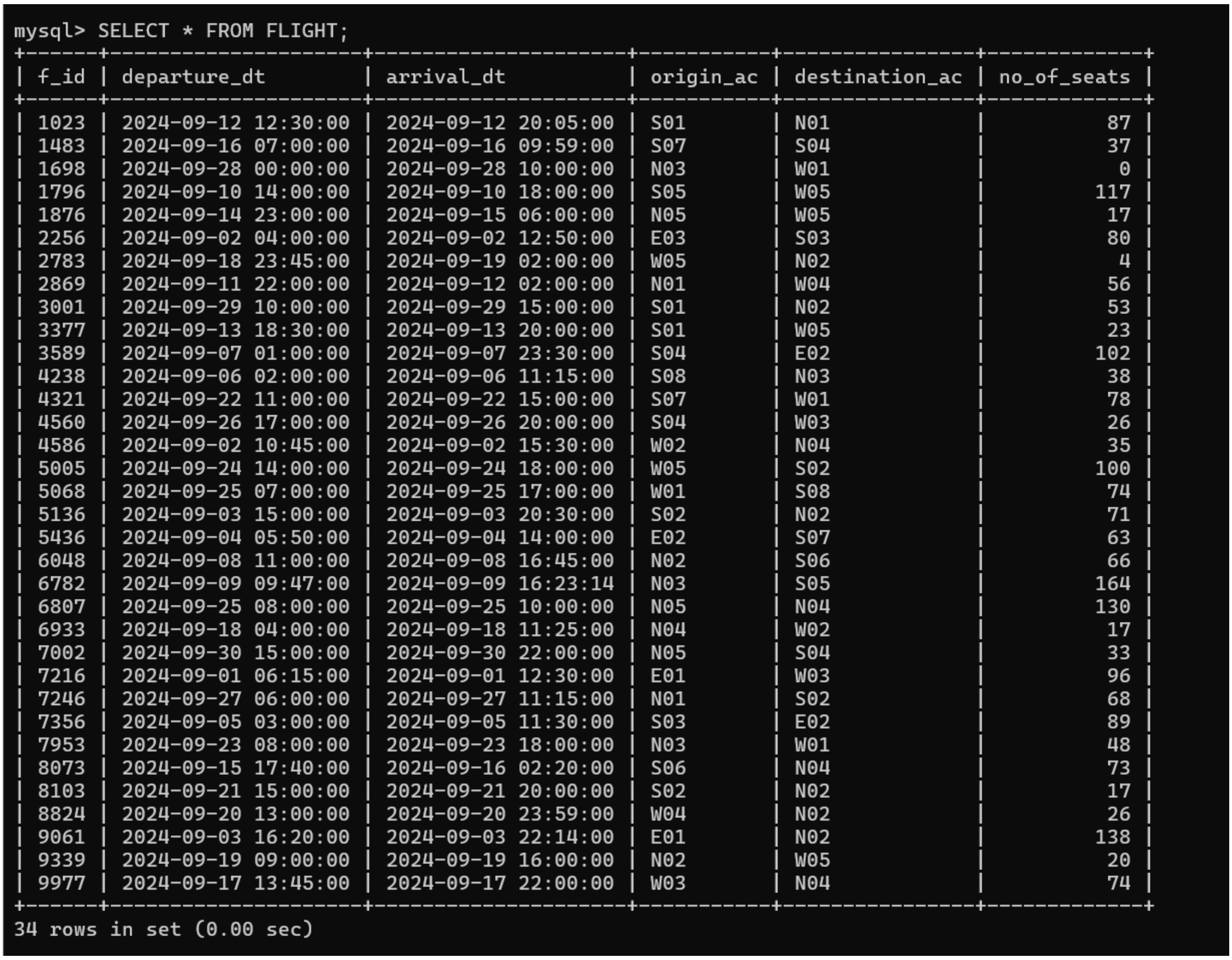
```
mysql> select * from airport;
+--------+------------------+-------------------+
| A_code | A_name           | A_location        |
+--------+------------------+-------------------+
| E01    | Kolkata          | West Bengal       |
| E02    | Guwahati         | Assam             |
| E03    | Bhubaneswar      | Odisha            |
| N01    | Amritsar         | Punjab            |
| N02    | Delhi            | Delhi             |
| N03    | Jaipur           | Rajasthan         |
| N04    | Srinagar         | Jammu and Kashmir |
| N05    | Lucknow          | Uttar Pradesh     |
| S01    | Chennai          | Tamil Nadu        |
| S02    | Bangalore        | Karnataka         |
| S03    | Tiruvanathapuram | Kerala            |
| S04    | Kochi            | Kerala            |
| S05    | Tirupati         | Andhra Pradesh    |
| S06    | Coimbatore       | Tamil Nadu        |
| S07    | Vijayawada       | Andhra Pradesh    |
| S08    | Visakhapatnam    | Andhra Pradesh    |
| W01    | Ahmedbad         | Gujarat           |
| W02    | Surat            | Gujarat           |
| W03    | Shirdi           | Maharashtra       |
| W04    | Indore           | Madhya Pradesh    |
| W05    | Mumbai           | Maharashtra       |
+--------+------------------+-------------------+
21 rows in set (0.00 sec)

mysql>
```



```
mysql> select * from passenger;
+------+------------+-----------+-------------+
| p_id | first_name | last_name | passport_no |
+------+------------+-----------+-------------+
|  167 | Sunny      | Lom       | SL0167      |
|  965 | Jack       | Kalix     | JK0965      |
| 1123 | Rose       | Mary      | RM1123      |
| 1254 | Akila      | Sabari    | AB1254      |
| 1325 | Tony       | Stark     | TS1325      |
| 2365 | Steve      | Rogers    | SR2365      |
| 2390 | Jonny      | Jack      | JJ2390      |
| 3366 | Clint      | Barton    | CB3366      |
| 4567 | James      | Bond      | JB4567      |
| 5648 | Bruce      | Banners   | BB5648      |
| 6897 | Natasha    | Romanoff  | NR6897      |
| 7890 | Bean       | Master    | BM7890      |
| 8796 | Luna       | Mary      | LM8796      |
+------+------------+-----------+-------------+
13 rows in set (0.01 sec)
```



```
mysql> select * from booking;
+------+------+------+------------+
| b_id | f_id | p_id | status     |
+------+------+------+------------+
|  101 | 1876 | 1254 | Successful |
|  102 | 4321 | 5648 | Pending    |
|  103 | 7953 | 8796 | Successful |
|  104 | 1876 | 2390 | Successful |
|  105 | 8103 | 6897 | Successful |
|  106 | 9061 | 3366 | Successful |
|  107 | 9061 | 2390 | Successful |
|  108 | 5136 | 1325 | Successful |
|  109 | 3001 | 1123 | Successful |
|  110 | 3001 | 1254 | Successful |
|  111 | 7002 |  965 | Successful |
|  112 | 6933 |  167 | Pending    |
|  113 | 3377 | 6897 | Pending    |
|  114 | 7216 | 1254 | Successful |
|  115 | 5005 | 1325 | Successful |
|  116 | 1483 | 2390 | Successful |
|  117 | 2869 | 5648 | Successful |
|  118 | 6807 | 2365 | Successful |
|  119 | 6048 | 7890 | Successful |
|  120 | 8824 |  167 | Successful |
|  121 | 4560 | 1123 | Pending    |
|  122 | 5436 |  965 | Successful |
|  123 | 6782 | 1325 | Successful |
|  124 | 7953 | 2390 | Pending    |
|  125 | 2783 | 1325 | Successful |
|  126 | 6048 |  167 | Successful |
|  127 | 8824 |  167 | Pending    |
|  128 | 4321 | 1325 | Successful |
|  129 | 3001 | 3366 | Successful |
|  130 | 1483 | 3366 | Successful |
|  131 | 6933 | 1325 | Successful |
|  132 | 7216 | 1325 | Successful |
+------+------+------+------------+
32 rows in set (0.00 sec)
```



```
mysql> SELECT * FROM FLIGHT;
+------+---------------------+---------------------+-----------+----------------+-------------+
| f_id | departure_dt        | arrival_dt          | origin_ac | destination_ac | no_of_seats |
+------+---------------------+---------------------+-----------+----------------+-------------+
| 1023 | 2024-09-12 12:30:00 | 2024-09-12 20:05:00 | S01       | N01            |          87 |
| 1483 | 2024-09-16 07:00:00 | 2024-09-16 09:59:00 | S07       | S04            |          37 |
| 1698 | 2024-09-28 00:00:00 | 2024-09-28 10:00:00 | N03       | W01            |           0 |
| 1796 | 2024-09-10 14:00:00 | 2024-09-10 18:00:00 | S05       | W05            |         117 |
| 1876 | 2024-09-14 23:00:00 | 2024-09-15 06:00:00 | N05       | W05            |          17 |
| 2256 | 2024-09-02 04:00:00 | 2024-09-02 12:50:00 | E03       | S03            |          80 |
| 2783 | 2024-09-18 23:45:00 | 2024-09-19 02:00:00 | W05       | N02            |           4 |
| 2869 | 2024-09-11 22:00:00 | 2024-09-12 02:00:00 | N01       | W04            |          56 |
| 3001 | 2024-09-29 10:00:00 | 2024-09-29 15:00:00 | S01       | N02            |          53 |
| 3377 | 2024-09-13 18:30:00 | 2024-09-13 20:00:00 | S01       | W05            |          23 |
| 3589 | 2024-09-07 01:00:00 | 2024-09-07 23:30:00 | S04       | E02            |         102 |
| 4238 | 2024-09-06 02:00:00 | 2024-09-06 11:15:00 | S08       | N03            |          38 |
| 4321 | 2024-09-22 11:00:00 | 2024-09-22 15:00:00 | S07       | W01            |          78 |
| 4560 | 2024-09-26 17:00:00 | 2024-09-26 20:00:00 | S04       | W03            |          26 |
| 4586 | 2024-09-02 10:45:00 | 2024-09-02 15:30:00 | W02       | N04            |          35 |
| 5005 | 2024-09-24 14:00:00 | 2024-09-24 18:00:00 | W05       | S02            |         100 |
| 5068 | 2024-09-25 07:00:00 | 2024-09-25 17:00:00 | W01       | S08            |          74 |
| 5136 | 2024-09-03 15:00:00 | 2024-09-03 20:30:00 | S02       | N02            |          71 |
| 5436 | 2024-09-04 05:50:00 | 2024-09-04 14:00:00 | E02       | S07            |          63 |
| 6048 | 2024-09-08 11:00:00 | 2024-09-08 16:45:00 | N02       | S06            |          66 |
| 6782 | 2024-09-09 09:47:00 | 2024-09-09 16:23:14 | N03       | S05            |         164 |
| 6807 | 2024-09-25 08:00:00 | 2024-09-25 10:00:00 | N05       | N04            |         130 |
| 6933 | 2024-09-18 04:00:00 | 2024-09-18 11:25:00 | N04       | W02            |          17 |
| 7002 | 2024-09-30 15:00:00 | 2024-09-30 22:00:00 | N05       | S04            |          33 |
| 7216 | 2024-09-01 06:15:00 | 2024-09-01 12:30:00 | E01       | W03            |          96 |
| 7246 | 2024-09-27 06:00:00 | 2024-09-27 11:15:00 | N01       | S02            |          68 |
| 7356 | 2024-09-05 03:00:00 | 2024-09-05 11:30:00 | S03       | E02            |          89 |
| 7953 | 2024-09-23 08:00:00 | 2024-09-23 18:00:00 | N03       | W01            |          48 |
| 8073 | 2024-09-15 17:40:00 | 2024-09-16 02:20:00 | S06       | N04            |          73 |
| 8103 | 2024-09-21 15:00:00 | 2024-09-21 20:00:00 | S02       | N02            |          17 |
| 8824 | 2024-09-20 13:00:00 | 2024-09-20 23:59:00 | W04       | N02            |          26 |
| 9061 | 2024-09-03 16:20:00 | 2024-09-03 22:14:00 | E01       | N02            |         138 |
| 9339 | 2024-09-19 09:00:00 | 2024-09-19 16:00:00 | N02       | W05            |          20 |
| 9977 | 2024-09-17 13:45:00 | 2024-09-17 22:00:00 | W03       | N04            |          74 |
+------+---------------------+---------------------+-----------+----------------+-------------+
34 rows in set (0.00 sec)
```

## FRONTEND AND CONNECTIVITY

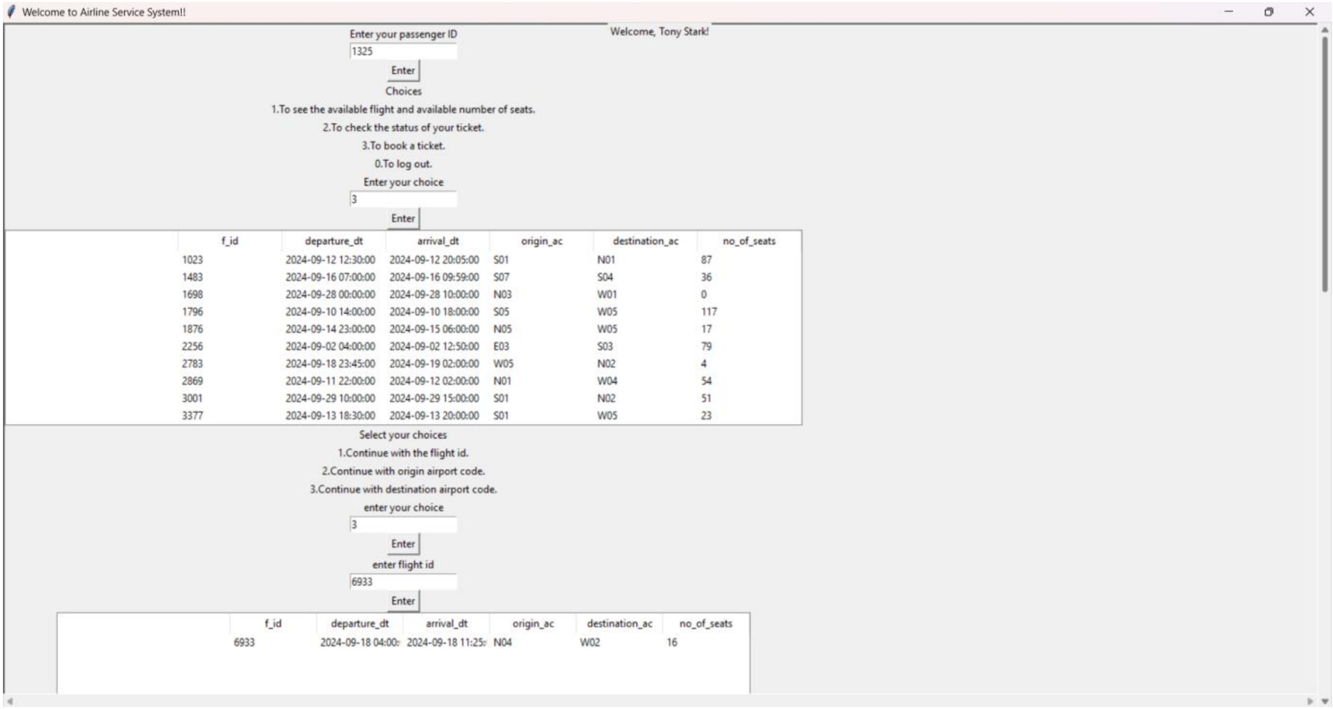
```
from tkinter import *

from tkinter import ttk

import mysql.connector as sql

import sys


conn=sql.connect(host='localhost',password='Krithika@23',user='root',database='airline')

cur=conn.cursor()


r=Tk()


canvas = Canvas(r)

canvas.place(relx=0.5, rely=0.5, anchor='center', relwidth=1.0, relheight=1.0)


y_scroll = Scrollbar(r, orient=VERTICAL, command=canvas.yview)

y_scroll.pack(side=RIGHT, fill=Y)

x_scroll =Scrollbar(r, orient=HORIZONTAL, command=canvas.xview)

x_scroll.pack(side=BOTTOM, fill=X)

canvas.configure(yscrollcommand=y_scroll.set, xscrollcommand=x_scroll.set)

frame = Frame(canvas)

canvas.create_window((0, 0), window=frame, anchor='nw')

r.update

canvas.config(scrollregion=canvas.bbox('all'))


def on_frame_configure(event):

    canvas.configure(scrollregion=canvas.bbox("all"))
```

```
frame.bind("<Configure>", on_frame_configure)

r.title("Welcome to Airline Service System!!")
r.geometry('700x700')

id=None
n=None
n1=None
n2=None
n3=None
bi=132

def get_passenger_id():
    global id
    id = e1.get()
    check_and_display()
    l5.pack()
    l6.pack()
    l7.pack()
    l8.pack()
    l9.pack()
    l10.pack()
    e2.pack()
    b2.pack()

def check_and_display():
    if check_id_exists(id):
        cur.execute("SELECT first_name, last_name FROM passenger WHERE p_id=%s", (id,))
        r2 = cur.fetchone()
        s = ' '.join(r2)
        l2 = Label(r, text=f"Welcome, {s}!")
        l2.pack()
        print("Welcome ",s,"!")
    else:
        l3 = Label(frame, text="Please enter your ID correctly to proceed.")
        l3.pack()
        l4 = Label(frame, text="TRY AGAIN!!")
        l4.pack()
        print("Please enter your ID correctly to proceed.")
        print("TRY AGAIN")
        r.destroy()
        sys.exit()



def check_id_exists(id):
    cur.execute("SELECT * FROM passenger WHERE p_id=%s",(id,))
    r1=cur.fetchone()
    return r1 is not None



def get_choice():
    global n
    n = int(e2.get())
    if(check_choice(n)):
        if(n==1):
            available()
        elif(n==2):
            status(id)
        elif(n==3):
            booking(id)
        elif(n==0):
            l11 = Label(frame, text="Have a great day")
            l11.pack()
            r.destroy()
            print("Have a great day!!")
            sys.exit()
    else:
        l12=Label(frame,text="Error!! Invalid choice. Please enter a number between 0 and 3.")
        l12.pack()
        r.destroy()
        print('"Error"\n"Invalid choice. Please enter a number between 0 and 3."')
        sys.exit()


def check_choice(n):
        if(n==0 or n==1 or n==2 or n==3):
                return True
        else:
                return False


def list_of_flights():
    cur.execute("SELECT * FROM flight")
    r3 = cur.fetchall()
    tree =ttk.Treeview(frame)
    tree["columns"]=("f_id", "departure_dt",
"arrival_dt","origin_ac","destination_ac","no_of_seats")
    for col in tree["columns"]:
        tree.column(col, width=120)
        tree.heading(col, text=col)
    for row in r3:
        tree.insert('', 'end', values=row)
    tree.pack()



def available():
    list_of_flights()
    l12=Label(frame,text="Select your choices")
    l13=Label(frame,text="1.Continue with the flight id.")
    l14=Label(frame,text="2.Continue with origin airport code.")
    l15=Label(frame,text="3.Continue with destination airport code.")
    l12.pack()
    l13.pack()
    l14.pack()
    l15.pack()
    l16.pack()
    e3.pack()
    b3.pack()


def get_choices():
    global n1
    n1 = int(e3.get())
    if(n1==1):
        l17.pack()
        e4.pack()
        b4.pack()

    elif(n1==2):
        l18.pack()
        e5.pack()
        b5.pack()

    elif(n1==3):
        l19.pack()
        e6.pack()
        b6.pack()

def get_flight_id():
    global n3
    n3 = int(e4.get())
    cur.execute("SELECT * FROM flight WHERE f_id=%s",(n3,))
    r4 = cur.fetchall()
    tree =ttk.Treeview(frame)
    tree["columns"]=("f_id", "departure_dt", 
"arrival_dt","origin_ac","destination_ac","no_of_seats")
    for col in tree["columns"]:
        tree.column(col, width=100)
        tree.heading(col, text=col)
    for row in r4:
        tree.insert('', 'end', values=row)
    tree.pack()



def get_oac():
    global a
    a=e5.get()
    cur.execute("SELECT * FROM flight WHERE origin_ac=%s",(a,))
    r5 = cur.fetchall()
    tree =ttk.Treeview(frame)
    tree["columns"]=("f_id", "departure_dt", "arrival_dt","origin_ac","destination_ac","no_of_seats")
    for col in tree["columns"]:
        tree.column(col, width=100)
        tree.heading(col, text=col)
    for row in r5:
        tree.insert('', 'end', values=row)
    tree.pack()


def get_dac():
    global b
    b=e6.get()
    cur.execute("SELECT * FROM flight WHERE destination_ac=%s",(b,))
    r6= cur.fetchall()
    tree =ttk.Treeview(frame)
    tree["columns"]=("f_id", "departure_dt", "arrival_dt","origin_ac","destination_ac","no_of_seats")
    for col in tree["columns"]:
        tree.column(col, width=100)

        tree.heading(col, text=col)

    for row in r6:

        tree.insert('', 'end', values=row)

    tree.pack()


def status(id):

    cur.execute("SELECT * FROM booking WHERE p_id=%s",(id,))

    r7=cur.fetchall()

    tree =ttk.Treeview(frame)

    tree["columns"]=("b_id", "f_id", "p_id","status")

    for col in tree["columns"]:

        tree.column(col, width=100)

        tree.heading(col, text=col)

    for row in r7:

        tree.insert('', 'end', values=row)

    tree.pack()


def get_f_id():

    global bi

    global n2

    n2=e7.get()

    cur.execute("UPDATE flight SET no_of_seats=no_of_seats-1 WHERE
f_id=%s",(n2,))

    conn.commit()

    bi+=1

    data=(bi,n2,id,"Successful")

    cur.execute("INSERT INTO booking VALUES(%s,%s,%s,%s)",data)
    conn.commit()
    l21.pack()



def booking(id):
    list_of_flights()
    l20.pack()
    e7.pack()
    b7.pack()

l1 = Label(frame, text="Enter your passenger ID")
l1.pack()
e1 = Entry(frame)
e1.pack()
b1 = Button(frame, text="Enter", command=get_passenger_id)
b1.pack()

l5=Label(frame,text="Choices")
l6=Label(frame,text="1.To see the available flight and available number of seats.")
l7=Label(frame,text="2.To check the status of your ticket.")
l8=Label(frame,text="3.To book a ticket.")
l9=Label(frame,text="0.To log out.")

l10 = Label(frame, text="Enter your choice")
e2 = Entry(frame)
b2 = Button(frame, text="Enter", command=get_choice)
```

```
l16 = Label(frame, text="enter your choice")

e3 = Entry(frame)

b3 = Button(frame, text="Enter", command=get_choices)


l17 = Label(frame, text="enter flight id")

e4 = Entry(frame)

b4 = Button(frame, text="Enter", command=get_flight_id)


l18 = Label(frame, text="enter the origin airport code")

e5 = Entry(frame)

b5 = Button(frame, text="Enter", command=get_oac)


l19 = Label(frame, text="enter the destination airport code")

e6= Entry(frame)

b6 = Button(frame, text="Enter", command=get_dac)


l20 = Label(frame, text="enter flight id")

e7 = Entry(frame)

b7 = Button(frame, text="Enter", command=get_f_id)


l21=Label(frame, text="Your ticket has been booked successfully you will recieve the e-ticket in your mail.")


r.mainloop()


conn.close()
```

# CHAPTER 5

# RESULTS



Welcome to Airline Service System!!

Welcome, Tony Stark!

Enter your passenger ID

1325

Choices

1.To see the available flight and available number of seats.

2.To check the status of your ticket.

3.To book a ticket.

0.To log out.

Enter your choice

3

| f_id | departure_dt | arrival_dt | origin_ac | destination_ac | no_of_seats |
|---|---|---|---|---|---|
| 1023 | 2024-09-12 12:30:00 | 2024-09-12 20:05:00 | S01 | N01 | 87 |
| 1483 | 2024-09-16 07:00:00 | 2024-09-16 09:59:00 | S07 | S04 | 36 |
| 1698 | 2024-09-28 00:00:00 | 2024-09-28 10:00:00 | N03 | W01 | 0 |
| 1796 | 2024-09-10 14:00:00 | 2024-09-10 18:00:00 | S05 | W05 | 117 |
| 1876 | 2024-09-14 23:00:00 | 2024-09-15 06:00:00 | N05 | W05 | 17 |
| 2256 | 2024-09-02 04:00:00 | 2024-09-02 12:50:00 | E03 | S03 | 79 |
| 2783 | 2024-09-18 23:45:00 | 2024-09-19 02:00:00 | W05 | N02 | 4 |
| 2869 | 2024-09-11 22:00:00 | 2024-09-12 02:00:00 | N01 | W04 | 54 |
| 3001 | 2024-09-29 10:00:00 | 2024-09-29 15:00:00 | S01 | N02 | 51 |
| 3377 | 2024-09-13 18:30:00 | 2024-09-13 20:00:00 | S01 | W05 | 23 |

Select your choices

1.Continue with the flight id.

2.Continue with origin airport code.

3.Continue with destination airport code.

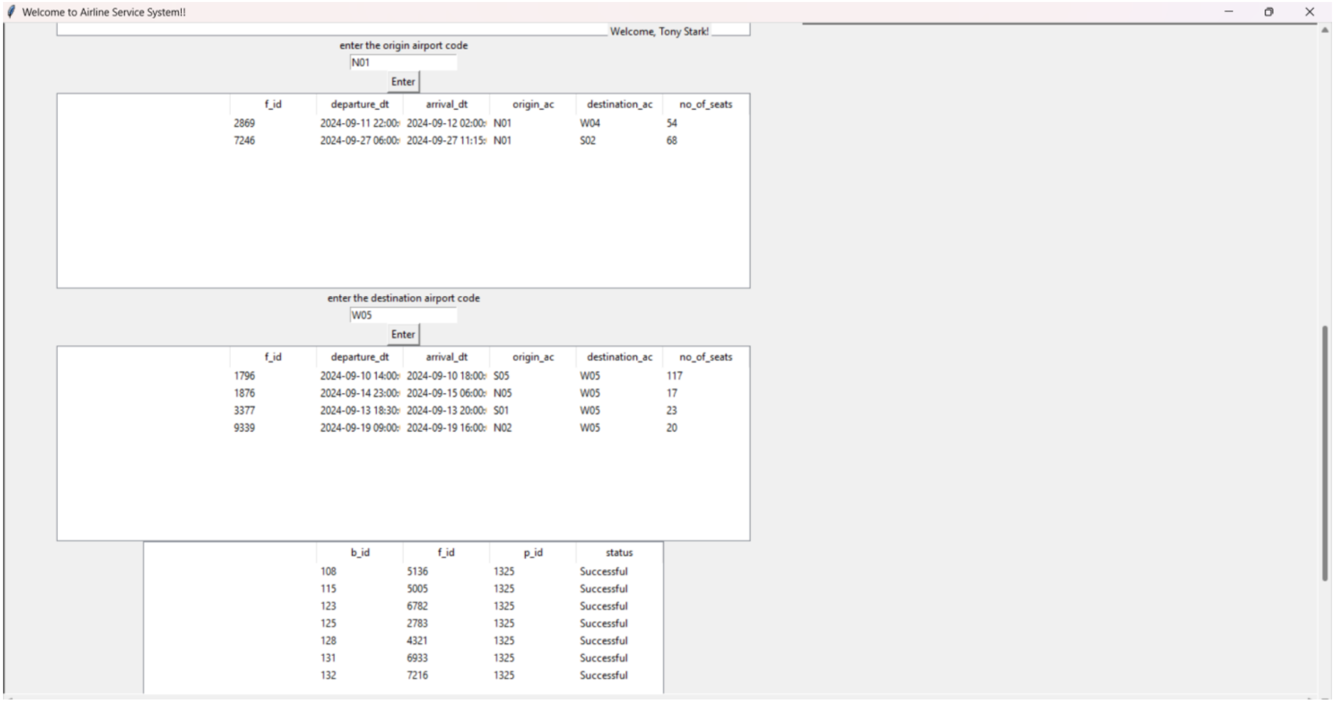
enter your choice

3

enter flight id

6933

| f_id | departure_dt | arrival_dt | origin_ac | destination_ac | no_of_seats |
|---|---|---|---|---|---|
| 6933 | 2024-09-18 04:00: | 2024-09-18 11:25: | N04 | W02 | 16 |



Welcome to Airline Service System!!

Welcome, Tony Stark!

enter the origin airport code

N01

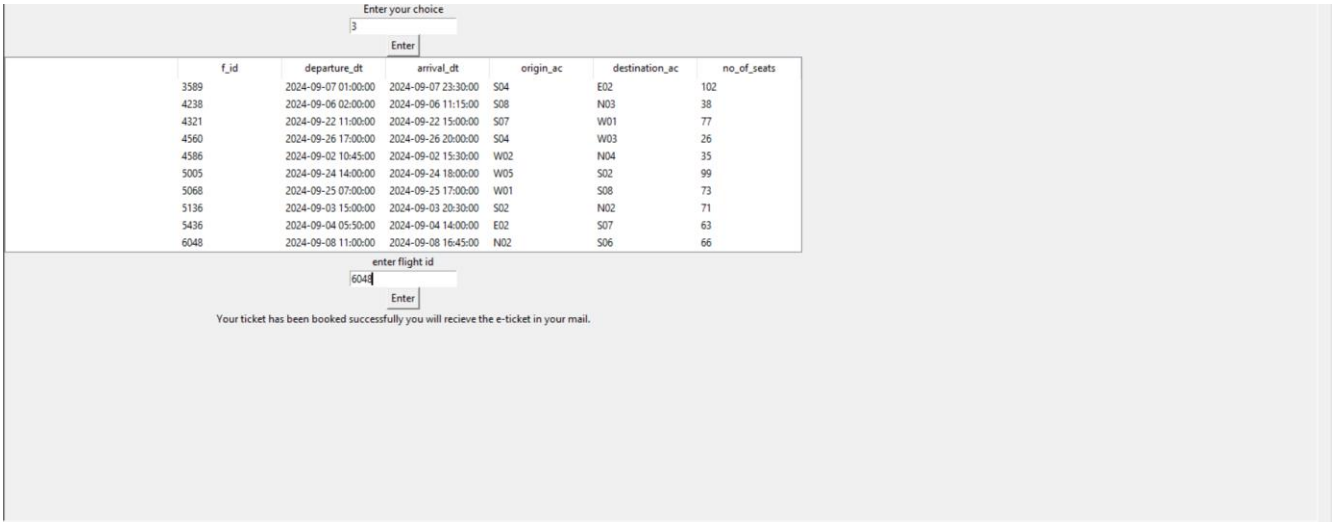
| f_id | departure_dt | arrival_dt | origin_ac | destination_ac | no_of_seats |
|---|---|---|---|---|---|
| 2869 | 2024-09-11 22:00: | 2024-09-12 02:00: | N01 | W04 | 54 |
| 7246 | 2024-09-27 06:00: | 2024-09-27 11:15: | N01 | S02 | 68 |

enter the destination airport code

W05

| f_id | departure_dt | arrival_dt | origin_ac | destination_ac | no_of_seats |
|---|---|---|---|---|---|
| 1796 | 2024-09-10 14:00: | 2024-09-10 18:00: | S05 | W05 | 117 |
| 1876 | 2024-09-14 23:00: | 2024-09-15 06:00: | N05 | W05 | 17 |
| 3377 | 2024-09-13 18:30: | 2024-09-13 20:00: | S01 | W05 | 23 |
| 9339 | 2024-09-19 09:00: | 2024-09-19 16:00: | N02 | W05 | 20 |

| b_id | f_id | p_id | status |
|---|---|---|---|
| 108 | 5136 | 1325 | Successful |
| 115 | 5005 | 1325 | Successful |
| 123 | 6782 | 1325 | Successful |
| 125 | 2783 | 1325 | Successful |
| 128 | 4321 | 1325 | Successful |
| 131 | 6933 | 1325 | Successful |
| 132 | 7216 | 1325 | Successful |



Enter your choice

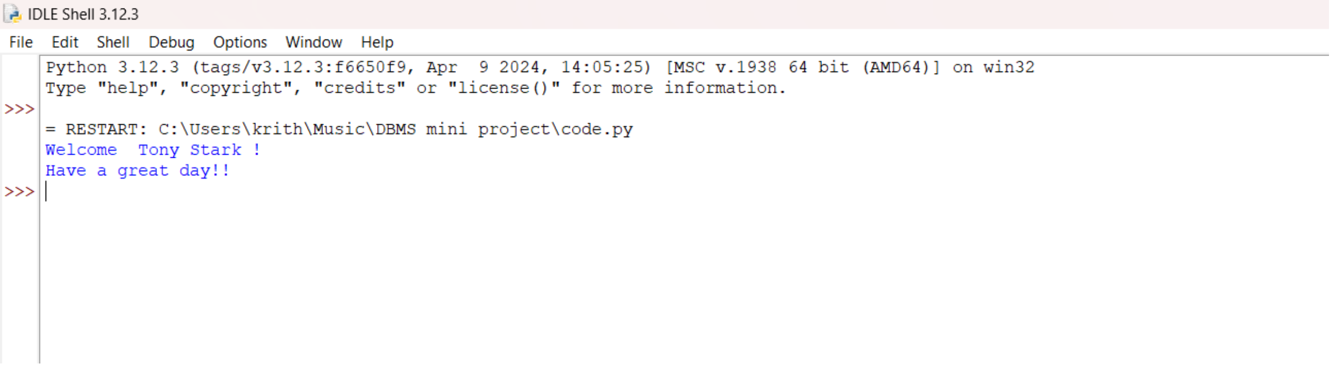
3

| f_id | departure_dt | arrival_dt | origin_ac | destination_ac | no_of_seats |
|---|---|---|---|---|---|
| 3589 | 2024-09-07 01:00:00 | 2024-09-07 23:30:00 | S04 | E02 | 102 |
| 4238 | 2024-09-06 02:00:00 | 2024-09-06 11:15:00 | S08 | N03 | 38 |
| 4321 | 2024-09-22 11:00:00 | 2024-09-22 15:00:00 | S07 | W01 | 77 |
| 4560 | 2024-09-26 17:00:00 | 2024-09-26 20:00:00 | S04 | W03 | 26 |
| 4586 | 2024-09-02 10:45:00 | 2024-09-02 15:30:00 | W02 | N04 | 35 |
| 5005 | 2024-09-24 14:00:00 | 2024-09-24 18:00:00 | W05 | S02 | 99 |
| 5068 | 2024-09-25 07:00:00 | 2024-09-25 17:00:00 | W01 | S08 | 73 |
| 5136 | 2024-09-03 15:00:00 | 2024-09-03 20:30:00 | S02 | N02 | 71 |
| 5436 | 2024-09-04 05:50:00 | 2024-09-04 14:00:00 | E02 | S07 | 63 |
| 6048 | 2024-09-08 11:00:00 | 2024-09-08 16:45:00 | N02 | S06 | 66 |

enter flight id

6048

Your ticket has been booked successfully you will recieve the e-ticket in your mail.



```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\krith\Music\DBMS mini project\code.py
Welcome  Tony Stark !
Have a great day!!
>>>
```

# CHAPTER 6
# CONCLUSION

To sum up, the Airline Service System is a powerful and intuitive platform that aims to transform airline services operations. It offers a smooth user experience by integrating essential features including flight availability display, ticket booking, and ticket status monitoring.

Even if the current system provides a thorough answer, the process is far from over. We see more features being added to the system in the future as it gets better. We are dedicated to innovation and ongoing progress. We intend to include functions like reward programs, personalized travel suggestions, real-time flight monitoring, and more. The purpose of these changes is to give users an even more convenient and personalized experience.

Additionally, we want to enhance the system's scalability and performance to maintain its dependability, efficiency, consistency as it grows and evolves.

# CHAPTER 7
# REFERENCES

https://www.geeksforgeeks.org/how-to-design-database-for-flight-reservation-system/

https://www.geeksforgeeks.org/python-gui-tkinter/

https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/1922-b.sc-cs-batchno-26.pdf.pdf.pdf

1. “Database System Concepts” by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan or “Fundamentals of Database Systems” by Ramez Elmasri and Shamkant B. Navathe provide in-depth knowledge about database design and management.
2. Python Crash Course” by Eric Matthes or “Learn Python the Hard Way” by Zed Shaw. These books provide a comprehensive introduction to Python programming.
3. “MySQL Explained: Your Step By Step Guide” by Mr Andrew Comeau or “Learning MySQL: Get a Handle on Your Data” by Seyed M.M. Tahaghoghi and Hugh E. Williams. These books provide a detailed understanding of MySQL.
4. Python GUI Programming with Tkinter” by Alan D. Moore or “Modern Tkinter for Busy Python Developers” by Mark Roseman. These books can help you understand how to create GUIs in Python using Tkinter.