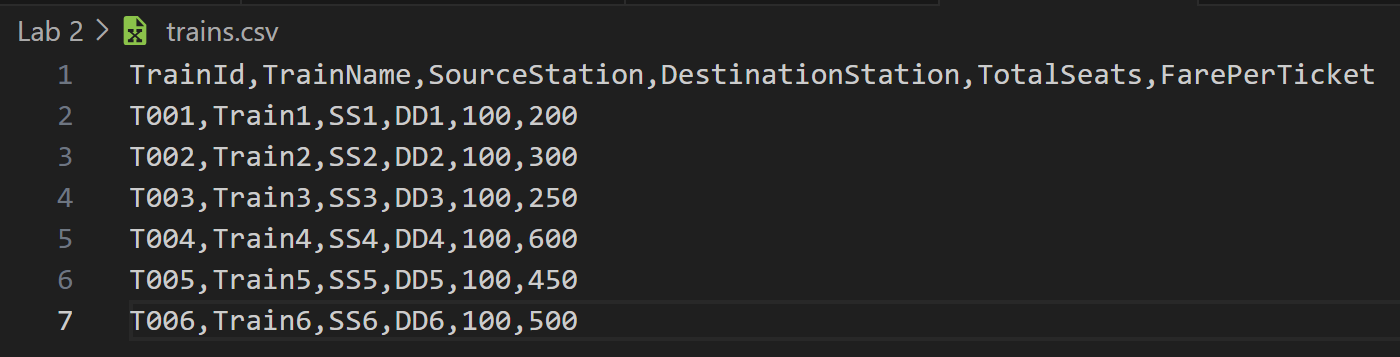
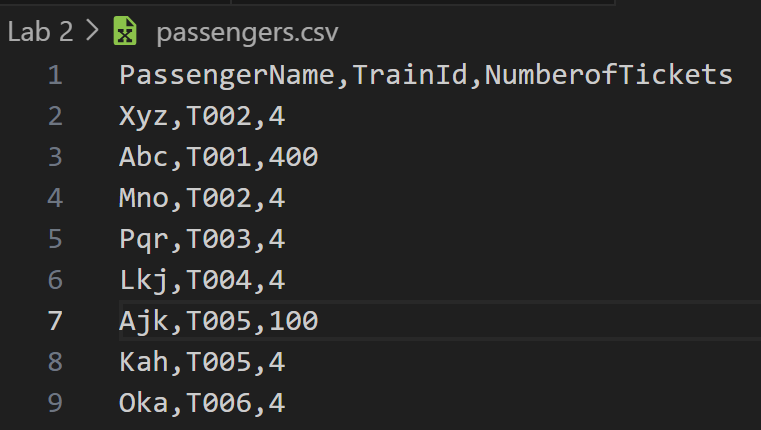
**Lab Assignment 2: Railway Ticket Reservation System**

You are tasked with developing a railway ticket reservation system for a busy rail network. The system should handle ticket booking, seat availability, and generate reports for the railway administration. Your task is to implement a Python program that provides the following functionalities:

* Load Train Data: The program should read the train data from a CSV file named "trains.csv." Each row in the CSV file represents a train with the following information:
  + Train ID (a unique alphanumeric code)
  + Train Name
  + Source Station
  + Destination Station
  + Total Seats (total number of seats available on the train)
* Load Passenger Data: The program should read the passenger data from a CSV file named "passengers.csv." Each row in the CSV file represents a passenger with the following information:
  + Passenger Name
  + Train ID (the ID of the train the passenger wants to book a ticket on)
  + Number of Tickets (the number of tickets the passenger wants to book)
* Check Seat Availability: Given the train ID and the number of tickets requested by a passenger, the program should check if there are enough seats available on the specified train for booking. If seats are available, the booking should be confirmed, and the total fare for the booking should be calculated as per the fare rules (you can define fare rules based on distance, class, etc.). Update Seat Availability: After confirming the booking, the program should update the seat availability for the corresponding train.
* Generate Reports:
  + Report 1: The program should generate a report showing the details of all the trains, including their names, source stations, destination stations, and the total number of seats available on each train.
  + Report 2: The program should generate a report showing the total revenue earned from each train based on the total number of confirmed bookings and their respective fares.
* Handle Errors: The program should handle various types of errors gracefully, such as invalid train IDs, invalid passenger names, insufficient seats, etc., and provide appropriate error messages.
* Text Files –





Code –

import csv

from collections import defaultdict

trains = {}

passenger\_bookings = defaultdict(list)

train\_revenue = defaultdict(int)

def load\_train\_data(filename):

    # Load train data from CSV file into the 'trains' dictionary.

    with open(filename, 'r') as file:

        reader = csv.DictReader(file)

        for row in reader:

            train\_id = row['TrainId']

            trains[train\_id] = {

                'Train Name': row['TrainName'],

                'Source Station': row['SourceStation'],

                'Destination Station': row['DestinationStation'],

                'Total Seats': int(row['TotalSeats']),

                'Available Seats': int(row['TotalSeats']),

                'Fare Per Ticket': float(row['FarePerTicket'])

            }

def load\_passenger\_data(filename):

    # Load passenger booking requests from a CSV file and attempt to book tickets.

    with open(filename, 'r') as file:

        reader = csv.DictReader(file)

        for row in reader:

            passenger\_name = row['PassengerName']

            train\_id = row['TrainId']

            num\_tickets = int(row['NumberofTickets'])

            book\_ticket(passenger\_name, train\_id, num\_tickets)

def check\_seat\_availability(train\_id, num\_tickets):

    # Check if there are enough seats available on the specified train.

    if train\_id not in trains:

        raise ValueError("Invalid train ID")

    if trains[train\_id]['Available Seats'] >= num\_tickets:

        return True

    else:

        return False

def book\_ticket(passenger\_name, train\_id, num\_tickets):

    # Book tickets for a passenger if seats are available.

    if check\_seat\_availability(train\_id, num\_tickets):

        trains[train\_id]['Available Seats'] -= num\_tickets

        passenger\_bookings[train\_id].append((passenger\_name, num\_tickets))

        fare\_per\_ticket = trains[train\_id]['Fare Per Ticket']

        total\_fare = num\_tickets \* fare\_per\_ticket

        train\_revenue[train\_id] += total\_fare

        print(f"Booking confirmed for {passenger\_name} on train {train\_id}. Total fare: Rupees {total\_fare}")

    else:

        print(f"Insufficient seats available for {passenger\_name} on train {train\_id}")

def generate\_train\_report():

    # Generate and print a report of all trains.

    print("Train Report:")

    for train\_id, details in trains.items():

        print(f"Train ID: {train\_id}, Name: {details['Train Name']}, Source: {details['Source Station']}, Destination: {details['Destination Station']}, Available Seats: {details['Available Seats']}, Fare Per Ticket: Rupees {details['Fare Per Ticket']}")

def generate\_revenue\_report():

    # Generate and print a report of total revenue for each train.

    print("Revenue Report:")

    for train\_id, revenue in train\_revenue.items():

        print(f"Train ID: {train\_id}, Total Revenue: Rupees {revenue}")

def calculate\_total\_revenue():

    # Calculate and return the total revenue generated.

    total\_revenue = sum(train\_revenue.values())

    return total\_revenue

load\_train\_data('D:/5th Lab/Python/Lab 2/trains.csv')

load\_passenger\_data('D:/5th Lab/Python/Lab 2/passengers.csv')

generate\_train\_report()

generate\_revenue\_report()

total\_revenue = calculate\_total\_revenue()

print(f"Total Revenue: Rupees {total\_revenue}")

* Output –

