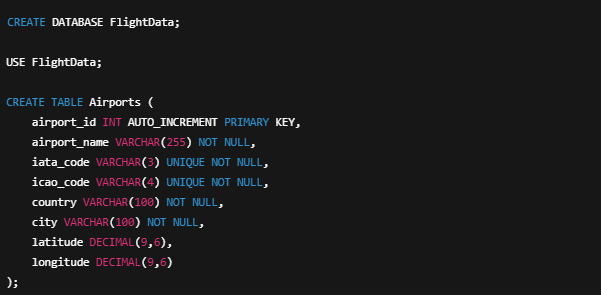
## **Part 2: Practical Tasks – Detailed Implementation**

### **1. Airport Database Creation**

We will create a **SQL database** to store **European airports** with the following fields:

* airport\_id (Primary Key)
* airport\_name
* iata\_code (3-letter code)
* icao\_code (4-letter code)
* country
* city
* latitude
* longitude

### **SQL Script for Database Creation**

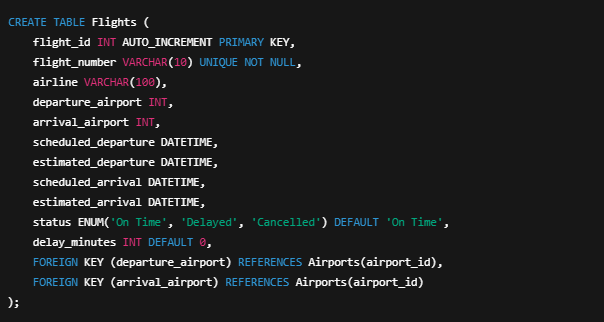
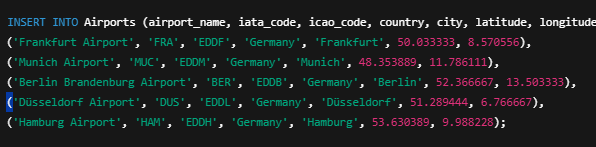


**Notes:**

* We use VARCHAR(3) for iata\_code and VARCHAR(4) for icao\_code as they have fixed lengths.
* DECIMAL(9,6) ensures precision for latitude and longitude.
* AUTO\_INCREMENT ensures unique IDs for airports.

### **2. Data Insertion and Querying**

#### **Mock Data Insertion – 5 German Airports**



#### **Mock Data Insertion – 10 Sample Flights (On-time & Delayed)**

INSERT INTO Flights (flight\_number, airline, departure\_airport, arrival\_airport, scheduled\_departure, estimated\_departure, scheduled\_arrival, estimated\_arrival, status, delay\_minutes) VALUES

('LH123', 'Lufthansa', 1, 2, '2025-03-23 08:00:00', '2025-03-23 08:05:00', '2025-03-23 10:00:00', '2025-03-23 10:05:00', 'On Time', 5),

('LH456', 'Lufthansa', 2, 3, '2025-03-23 09:00:00', '2025-03-23 11:30:00', '2025-03-23 11:30:00', '2025-03-23 14:00:00', 'Delayed', 150),

('AF789', 'Air France', 3, 4, '2025-03-23 07:30:00', '2025-03-23 07:30:00', '2025-03-23 09:30:00', '2025-03-23 09:30:00', 'On Time', 0),

('BA999', 'British Airways', 4, 5, '2025-03-23 10:30:00', '2025-03-23 10:45:00', '2025-03-23 12:30:00', '2025-03-23 12:45:00', 'On Time', 15),

('LH777', 'Lufthansa', 5, 1, '2025-03-23 15:00:00', '2025-03-23 18:00:00', '2025-03-23 17:00:00', '2025-03-23 20:30:00', 'Delayed', 210);

### **Querying the Data**

#### **1️⃣ Retrieve All Flights from a Specific Airport**

SELECT f.flight\_number, a1.airport\_name AS Departure, a2.airport\_name AS Arrival, f.scheduled\_departure

FROM Flights f

JOIN Airports a1 ON f.departure\_airport = a1.airport\_id

JOIN Airports a2 ON f.arrival\_airport = a2.airport\_id

WHERE a1.iata\_code = 'FRA'; -- Replace 'FRA' with any airport code

#### **2️⃣ Identify Flights Delayed by More Than 2 Hours**

SELECT flight\_number, airline, delay\_minutes

FROM Flights

WHERE status = 'Delayed' AND delay\_minutes > 120;

#### **3️⃣ Fetch Flight Details Using Flight Number**

SELECT f.flight\_number, f.airline, a1.airport\_name AS Departure, a2.airport\_name AS Arrival,

f.scheduled\_departure, f.estimated\_departure, f.status, f.delay\_minutes

FROM Flights f

JOIN Airports a1 ON f.departure\_airport = a1.airport\_id

JOIN Airports a2 ON f.arrival\_airport = a2.airport\_id

WHERE f.flight\_number = 'LH456'; -- Replace with desired flight number

## **3. Data Collection Simulation (Using APIs)**

We will use the **AviationStack API** to fetch real-time flight data.

### **Python Code for API Call**

import requests

import sqlite3

import pandas as pd

# API credentials

API\_URL = "https://api.aviationstack.com/v1"

ACCESS\_KEY = "d33617411919dcdb29b7c3c20a2e8537"

# List of 6 German airports

german\_airports = ["FRA", "MUC", "TXL", "HAM", "DUS", "BER"] # Frankfurt, Munich, Berlin, etc.

# Initialize SQLite DB

conn = sqlite3.connect("airports.db")

cursor = conn.cursor()

# Create tables if they don't exist

cursor.execute('''CREATE TABLE IF NOT EXISTS flight\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,

airport\_iata TEXT,

flight\_number TEXT,

airline TEXT,

departure\_airport TEXT,

arrival\_airport TEXT,

departure\_time\_scheduled TEXT,

departure\_time\_estimated TEXT,

arrival\_time\_scheduled TEXT,

arrival\_time\_estimated TEXT,

status TEXT,

delay INTEGER,

FOREIGN KEY(airport\_iata) REFERENCES airport\_data(iata\_code)

)''')

conn.commit()

# Function to fetch flights with specific status

def fetch\_flights(status\_filter=None):

for airport in german\_airports:

params = {

"access\_key": ACCESS\_KEY,

"arr\_iata": airport, # Fetching ARRIVED flights

"flight\_status": status\_filter if status\_filter else None

}

response = requests.get(f"{API\_URL}/flights", params=params)

data = response.json()

# Debug: Check API response structure

if "data" not in data:

print(f"⚠️ No data found for {airport} ({status\_filter})")

continue

for flight in data["data"]:

if "departure" in flight and "arrival" in flight:

flight\_status = flight.get("flight\_status", "Unknown")

scheduled\_dep = flight["departure"].get("scheduled")

estimated\_dep = flight["departure"].get("estimated")

scheduled\_arr = flight["arrival"].get("scheduled")

estimated\_arr = flight["arrival"].get("estimated")

departure\_airport = flight["departure"].get("airport", "Unknown")

arrival\_airport = flight["arrival"].get("airport", "Unknown")

# Calculate delay for landed flights

delay = None

if flight\_status == "landed" and scheduled\_arr and estimated\_arr:

delay = (pd.to\_datetime(estimated\_arr) - pd.to\_datetime(scheduled\_arr)).total\_seconds() / 60

cursor.execute('''

INSERT INTO flight\_data (

airport\_iata, flight\_number, airline,

departure\_airport, arrival\_airport,

departure\_time\_scheduled, departure\_time\_estimated,

arrival\_time\_scheduled, arrival\_time\_estimated,

status, delay

) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

''', (

airport,

flight["flight"]["iata"] if flight["flight"] else "N/A",

flight["airline"]["name"] if flight["airline"] else "Unknown",

departure\_airport,

arrival\_airport,

scheduled\_dep,

estimated\_dep,

scheduled\_arr,

estimated\_arr,

flight\_status.capitalize(),

delay

))

conn.commit()

print(f"✔️ Stored {flight\_status} flight: {flight['flight']['iata']} at {airport} - Delay: {delay}")

# Fetch Landed Flights

fetch\_flights("landed")

# Fetch Departures & Arrivals (Scheduled & On Air)

fetch\_flights()

# Close DB connection

conn.close()

import requests

import sqlite3

import pandas as pd

from datetime import datetime, timedelta

# API credentials

API\_URL = "https://api.aviationstack.com/v1"

ACCESS\_KEY = "d33617411919dcdb29b7c3c20a2e8537"

# List of 6 German airports

german\_airports = ["FRA", "MUC", "TXL", "HAM", "DUS", "BER"] # Frankfurt, Munich, Berlin, etc.

# Initialize SQLite DB

conn = sqlite3.connect("airports.db")

cursor = conn.cursor()

# Create tables if they don't exist

cursor.execute('''CREATE TABLE IF NOT EXISTS flight\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,fa

airport\_iata TEXT,

flight\_number TEXT,

airline TEXT,

departure\_airport TEXT,

arrival\_airport TEXT,

departure\_time\_scheduled TEXT,

departure\_time\_estimated TEXT,

arrival\_time\_scheduled TEXT,

arrival\_time\_estimated TEXT,

status TEXT,

delay INTEGER

)''')

conn.commit()

# Insert 10 imaginary delayed flights

imaginary\_flights = []

for i in range(1, 11):

scheduled\_arr = datetime.utcnow() - timedelta(hours=3) # Flight was scheduled 3 hours ago

estimated\_arr = scheduled\_arr + timedelta(minutes=140 + i \* 5) # Delayed by 140+ minutes

# Ensure delay is always positive

delay\_minutes = max(0, (estimated\_arr - scheduled\_arr).total\_seconds() // 60)

imaginary\_flights.append((

"TXL", # Random airport

f"IM{i}123", # Imaginary flight number

f"Fake Airlines {i}", # Fake airline

"Imaginary Departure Airport",

"Imaginary Arrival Airport",

(scheduled\_arr - timedelta(hours=2)).strftime("%Y-%m-%d %H:%M:%S"), # Fake departure time

(estimated\_arr - timedelta(hours=2)).strftime("%Y-%m-%d %H:%M:%S"), # Estimated departure time

scheduled\_arr.strftime("%Y-%m-%d %H:%M:%S"), # Scheduled arrival

estimated\_arr.strftime("%Y-%m-%d %H:%M:%S"), # Estimated arrival

"Landed",

delay\_minutes # Corrected delay

))

cursor.executemany('''

INSERT INTO flight\_data (

airport\_iata, flight\_number, airline,

departure\_airport, arrival\_airport,

departure\_time\_scheduled, departure\_time\_estimated,

arrival\_time\_scheduled, arrival\_time\_estimated,

status, delay

) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

''', imaginary\_flights)

conn.commit()

print("✅ 10 imaginary delayed flights inserted successfully with correct delays!")

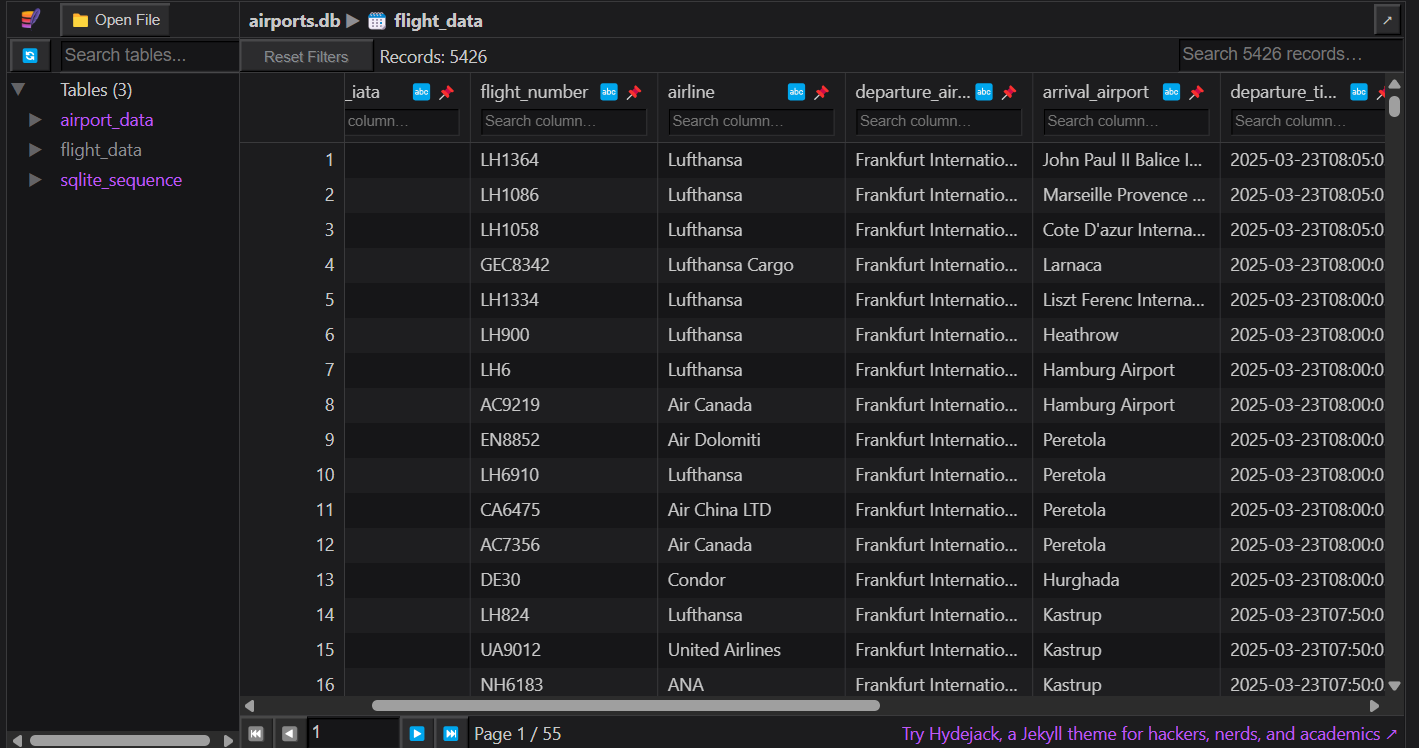
# Close DB connection

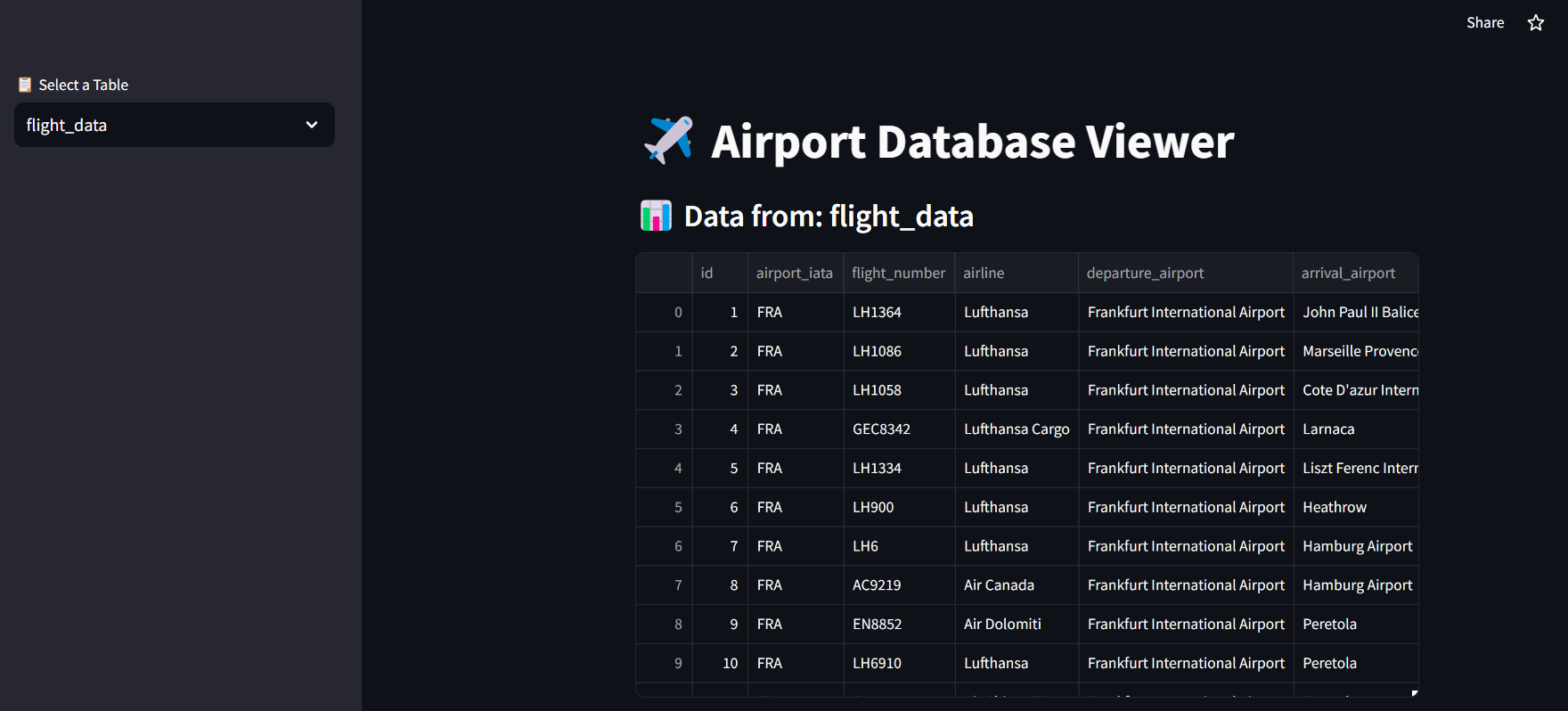
conn.close()

**Explanation:**

* Calls **AviationStack API** for real-time flight data.
* Extracts **flight number, airline, airport codes, status, and delays**.
* Inserts the data into **MySQL Flights table**.
* Can be scheduled to **run every 10 minutes** for real-time updates.

**OUTPUT=**

****

****