LIVE FLIGHT DATA ANALYSIS

Submitted by
JOES AROCKIAM X - 71762133020

Submitted to
Dr.V.Radhamani
Dr.A.Aruna
Department of Decision and Computing Sciences

COIMBATORE INSTITUTE OF TECHNOLOGY
(Government Aided Autonomous Institution)
Coimbatore-641014

April– 2025

Table of Contents

1. Introduction

This project involves analyzing live flight data from various countries in real-time, using data from the OpenSky Network API. The goal is to monitor and process flight data, store it for future analysis, and provide real-time visualizations using Power BI. By leveraging AWS services like Lambda, S3, and Athena, the system fetches and stores data efficiently, while the custom-built Power BI dashboards offer insights into global flight operations. Additionally, the project includes an alert engine to monitor and flag important flight events.

Technologies Used:

Python: Used for fetching and processing live flight data from the OpenSky API, interacting with AWS services, and integrating the alert engine for monitoring specific flight conditions.

AWS:

Lambda: Automates the process of fetching flight data from the OpenSky API.

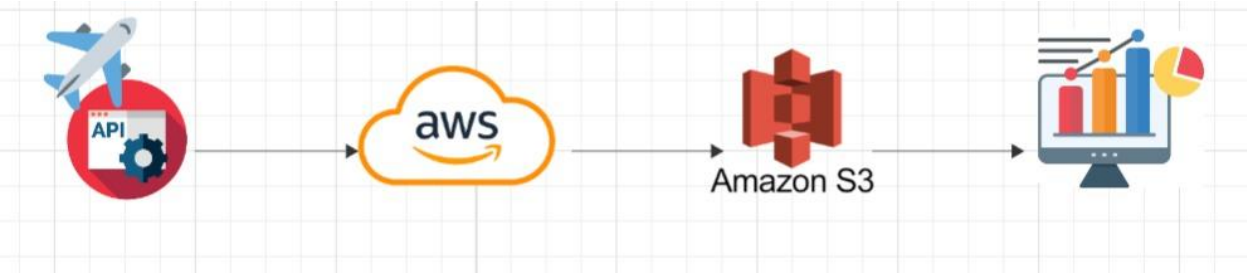
S3 Bucket: Stores the fetched data in JSON format for easy access and analysis.

Athena: Queries the data stored in S3 and synchronizes it with Power BI for real-time reporting.

Power BI: Used for visualizing the live flight data through custom-built dashboards, allowing users to monitor real-time flight status and trends.

2. System Design and Architecture

High-Level Architecture Overview:



- Data Fetching: An AWS Lambda function fetches real-time flight data from the OpenSky API at regular intervals.

- Data Storage: The fetched data is then stored in an AWS S3 Bucket in a structured JSON format.

- Data Synchronization: AWS Athena is used to query and synchronize data from the S3 bucket to Power BI for real-time updates.

- Visualization: The live data is visualized in custom dashboards in Power BI to provide insights into the real-time flight status across various countries.

3. Implementation Details

Data Collection:

Source Data : https://opensky-network.org/api/states/all

- A Python script fetches live flight data from the OpenSky Network API using the requests library.
- The data includes aircraft details such as location, altitude, and velocity.

Data Storage:

- The data is processed and stored in JSON format in an AWS S3 Bucket for structured storage.
- Each file is named based on the current timestamp to ensure uniqueness.

Data Synchronization and Visualization:

- AWS Athena queries the data in the S3 bucket for synchronization with Power BI.
- Power BI visualizes the data through interactive dashboards and dynamic metrics.

4.Code Documentation

Fetching Data using AWS Lambda:

```
import json
import requests
import datetime
def lambda_handler(event, context):
    # OpenSky API URL (example for all states)
    url = "https://opensky-network.org/api/states/all"

    # Fetch the data from OpenSky API
    response = requests.get(url)
    data = response.json()

    # Create a unique file name based on current time
    current_time = datetime.datetime.utcnow()
    file_name = f"opensky_data_{current_time.strftime('%Y-%m-%d_%H-%M-%S')}.json"

    # Upload data to S3
    s3 = boto3.client('s3')
    bucket_name = 'opensky-api-data-bucket'  # Replace with your actual S3 bucket name
    s3.put_object(
        Bucket=bucket_name,
        Key=f"opensky-data/{file_name}",
        Body=json.dumps(data),
        ContentType='application/json'
```

```
    return {
        'statusCode': 200,
        'body': json.dumps('Data successfully saved to S3!')
    }
```

Alert Engine Code:

The alert engine monitors flight states and triggers alerts for various conditions (e.g., sudden drops in altitude, suspicious callsigns, ground congestion near JFK airport).

```
def run_alert_engine():
    response = requests.get("https://opensky-network.org/api/states/all")
    if response.status_code != 200:
        print("Failed to fetch data")
        return

    states = response.json().get("states", [])
    if not states:
        print("No state vectors found.")
        return

    for item in states:
        callsign = item[1]
        country = item[2]
        lon = item[5]
        lat = item[6]
```

```
        geo_alt = item[13]
        vertical_rate = item[11]
        velocity = item[9]
        on_ground = item[9]
        # 1. Sudden Drop in Altitude
        if vertical_rate is not None and geo_alt is not None:
            if vertical_rate < -4000 and geo_alt > 1000:
                alerts.append("⚠️ Sudden Drop in Altitude")
                log_to_csv("sudden_drops.csv", {
                    "callsign": callsign, "vertical_rate": vertical_rate, "geo_altitude": geo_alt, "country": country
                })
        # 2. Suspicious Callsign
        if not callsign or any(char.isdigit() for char in callsign.strip()):
            alerts.append("⚠️ Suspicious Callsign")
            log_to_csv("suspicious_flights.csv", {
                "callsign": callsign, "country": country, "latitude": lat, "longitude": lon
            }  # -- If any alert triggered, send email
        if alerts:
            subject = f"🚨 Flight Alert: {callsign or 'Unknown'}"
            body = f"Alerts: {', '.join(alerts)}\nLocation: {lat}, {lon}\nCountry: {country}"
            send_email(subject, body)
            print(f"Sent alert for {callsign}: {alerts}")

run_alert_engine()
```

5. Testing and Evaluation Results

Validation Strategies:

1. API Testing: Storing Data into S3

- The Lambda function successfully fetched data from the OpenSky Network API and stored it in the S3 bucket. Each API call was followed by a confirmation message stating that the data was successfully saved to S3.

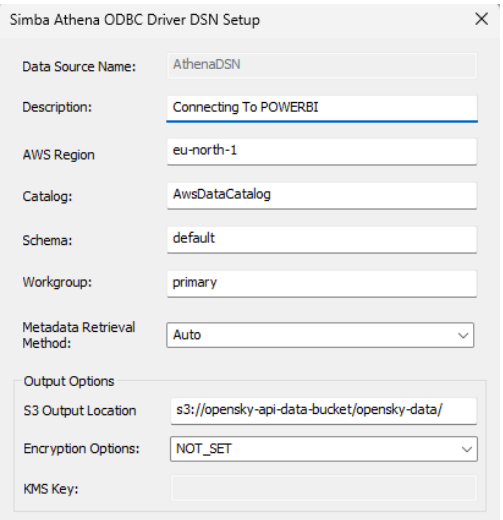
- **Test result:** Success

```
Status: Succeeded
Test Event Name: firstevent

Response:
{
  "statusCode": 200,
  "body": "\"Data successfully saved to S3!\""
}

Function Logs:
START RequestId: 7d555aac-4f7f-441d-bcf7-a6529e4da78e Version: $LATEST
END RequestId: 7d555aac-4f7f-441d-bcf7-a6529e4da78e
REPORT RequestId: 7d555aac-4f7f-441d-bcf7-a6529e4da78e  Duration: 1845.75 ms    Billed Duration: 1846 ms    Memory Size: 128 MB Max Memory Used: 100 MB

Request ID: 7d555aac-4f7f-441d-bcf7-a6529e4da78e
```

PowerBI Testing: **Data Loading**

- Data fetched and stored in S3 was successfully queried using **AWS Athena** and synchronized with **Power BI**. The visualization provided real-time insights into flight data, and the custom dashboards updated regularly as new data was fetched.

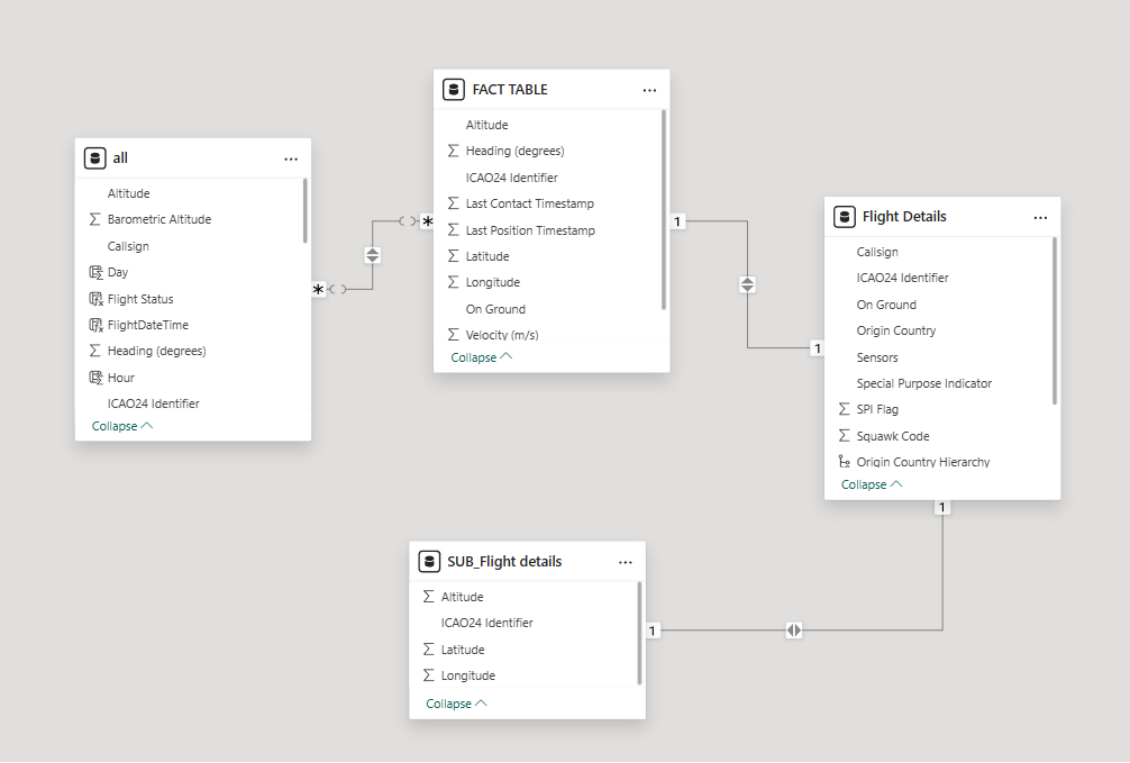
- **Test result:** Success



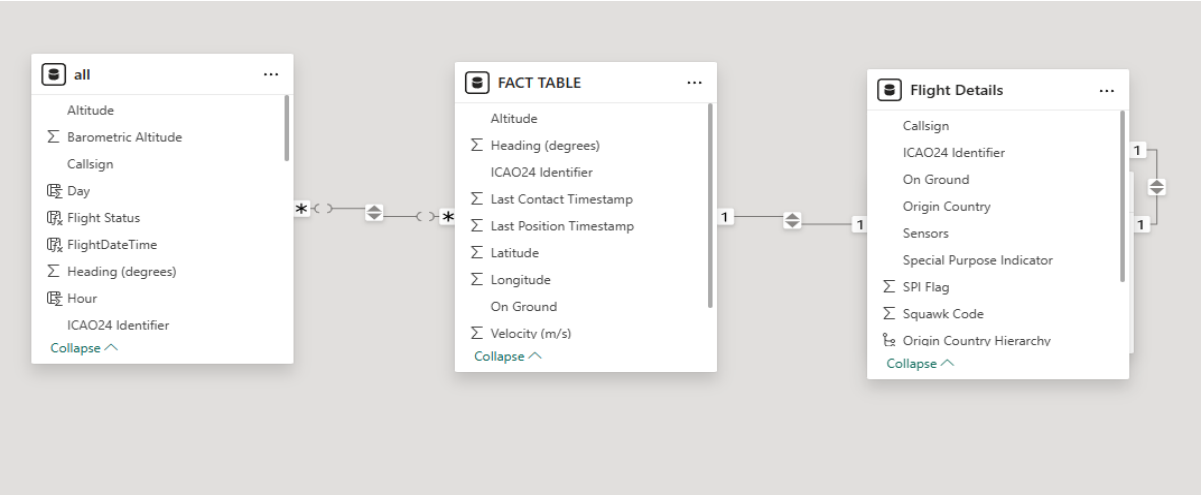
6.Output screenshots

Schema Design

SNOWFLAKE Schema



STAR Schema



DAX QUERY

```
EVALUATE
VAR ResultTable =
    ADDCOLUMNS (
        'all',
        "IN AIR",
        IF (
            NOT ISBLANK('all'[Altitude]) &&
            'all'[Altitude] > 0,
            "🛬 Aircraft In Air",
            BLANK()
        )
    )
RETURN
    SELECTCOLUMNS (
        FILTER (ResultTable, NOT ISBLANK([IN AIR])),
        "Callsign", 'all'[Callsign],
        "Altitude", 'all'[Altitude],
        "Vertical Rate", 'all'[Vertical Rate (m/s)],
        "IN AIR", [IN AIR]
    )
```

Results | Result 1 of 1 | Copy

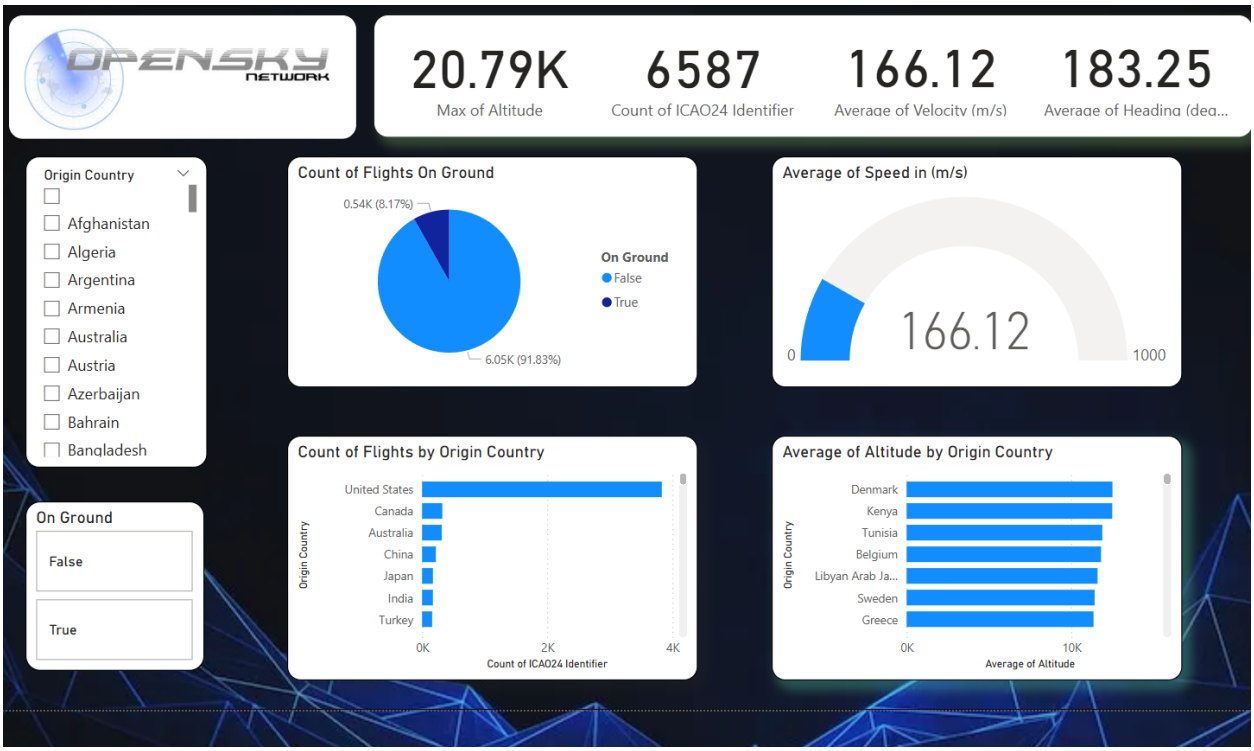
	[Callsign]	[Altitude]	[Vertical Rate]	[IN AIR]
308	VXP136	4838.7	-0.33	Aircraft In Air
309	NKS694	9753.6	-0.33	Aircraft In Air
310	SKW3958	11277.6	-0.33	Aircraft In Air
311	AAL2131	10058.4	-0.33	Aircraft In Air

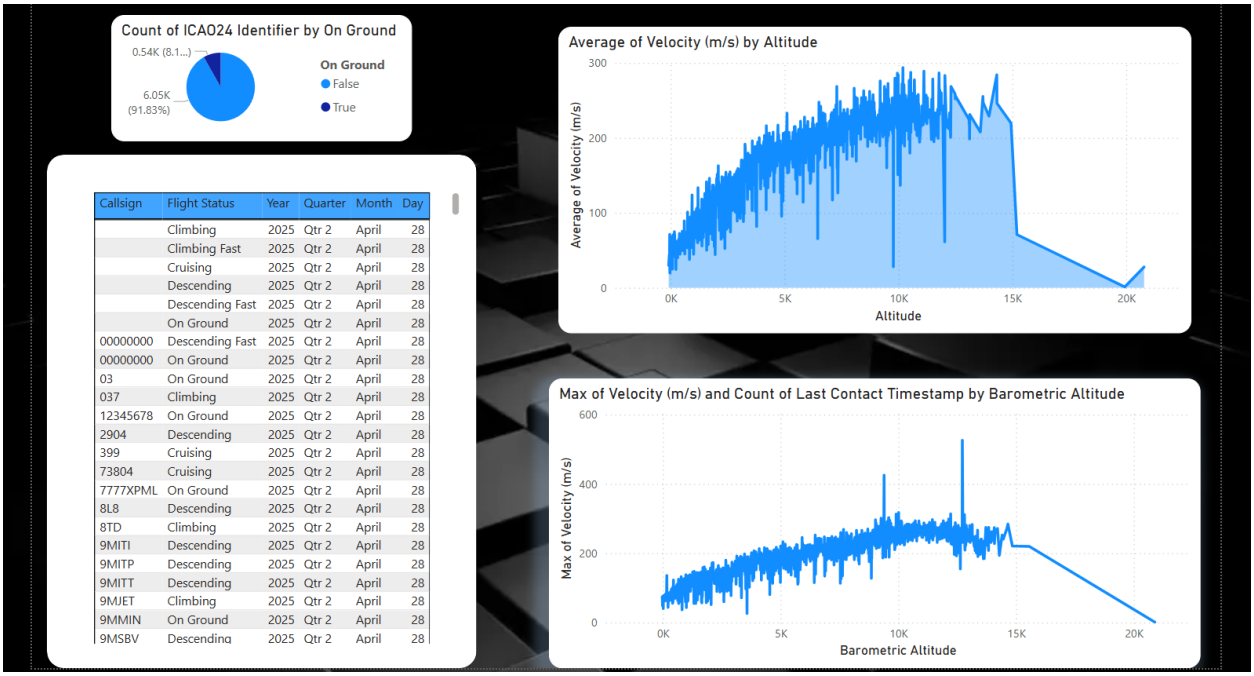
```
EVALUATE
    SELECTCOLUMNS(
        ADDCOLUMNS(
            'all',
            "Speed Band",
            SWITCH(
                TRUE(),
                'all'[Velocity (m/s)] < 100, "Slow (<100 km/h)",
                'all'[Velocity (m/s)] < 300, "Medium (100-300 km/h)",
                'all'[Velocity (m/s)] < 600, "Fast (300-600 km/h)",
                "Very Fast (>600 km/h)"
            )
        ),
        "Origin Country", 'all'[Origin Country],
        "Callsign", 'all'[callsign],
        "Speed Band", [Speed Band]
    )
```

Results | Result 1 of 1 | Copy

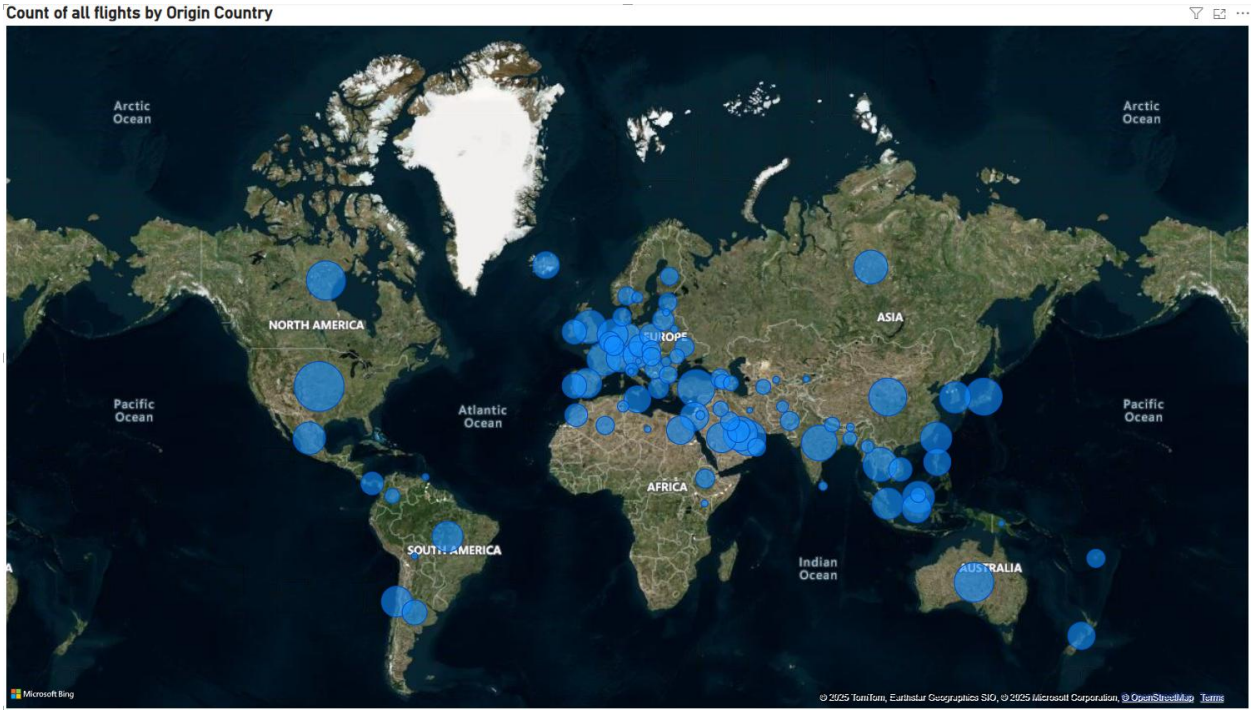
	[Origin Country]	[Callsign]	[Speed Band]
1	United States	UAL2306	Medium (100-300 km/h)
2	United States	N452TR	Medium (100-300 km/h)
3	United States	N555AL	Medium (100-300 km/h)
4	United States	RCH451	Medium (100-300 km/h)
5	United States	CXK1113	Slow (<100 km/h)
6	United States	N3132V	Medium (100-300 km/h)
7	United States	N28T	Medium (100-300 km/h)

DASHBOARD OUTPUT:

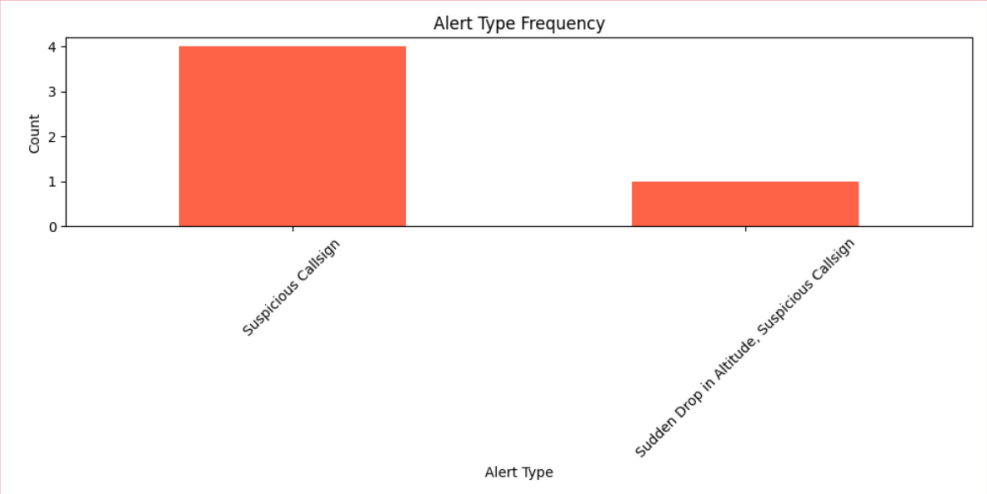




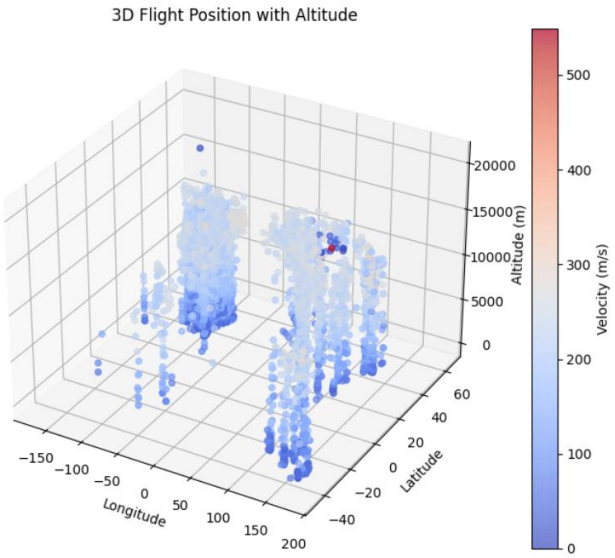
Callsign	Flight Status	Year	Quarter	Month	Day
	Climbing	2025	Qtr 2	April	28
	Climbing Fast	2025	Qtr 2	April	28
	Cruising	2025	Qtr 2	April	28
	Descending	2025	Qtr 2	April	28
	Descending Fast	2025	Qtr 2	April	28
	On Ground	2025	Qtr 2	April	28
00000000	Descending Fast	2025	Qtr 2	April	28
00000000	On Ground	2025	Qtr 2	April	28
03	On Ground	2025	Qtr 2	April	28
037	Climbing	2025	Qtr 2	April	28
12345678	On Ground	2025	Qtr 2	April	28
2904	Descending	2025	Qtr 2	April	28
399	Cruising	2025	Qtr 2	April	28
73804	Cruising	2025	Qtr 2	April	28
7777XPML	On Ground	2025	Qtr 2	April	28
8L8	Descending	2025	Qtr 2	April	28
8TD	Climbing	2025	Qtr 2	April	28
9MITI	Descending	2025	Qtr 2	April	28
9MITP	Descending	2025	Qtr 2	April	28
9MITT	Descending	2025	Qtr 2	April	28
9MJET	Climbing	2025	Qtr 2	April	28
9MMIN	On Ground	2025	Qtr 2	April	28
9MSBV	Descending	2025	Qtr 2	April	28



Latitude, Longitude, Callsign, Altitude, Vertical Rate (m/s) and On Ground

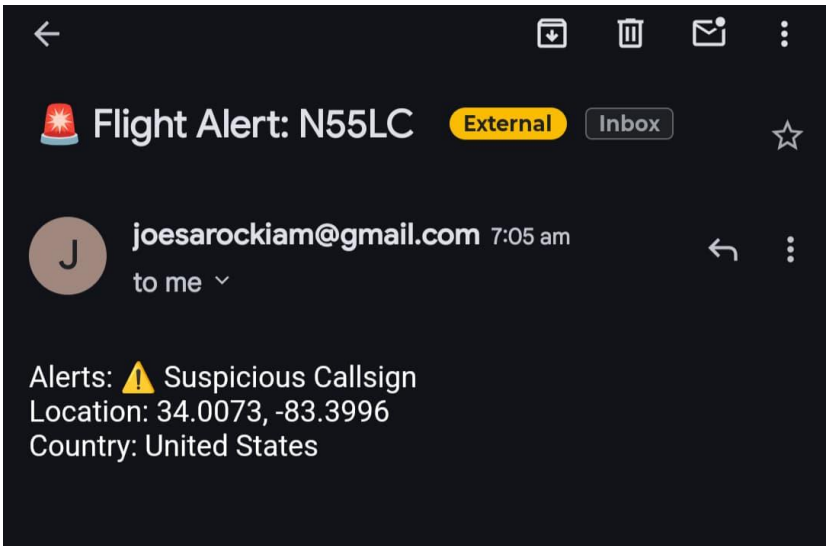


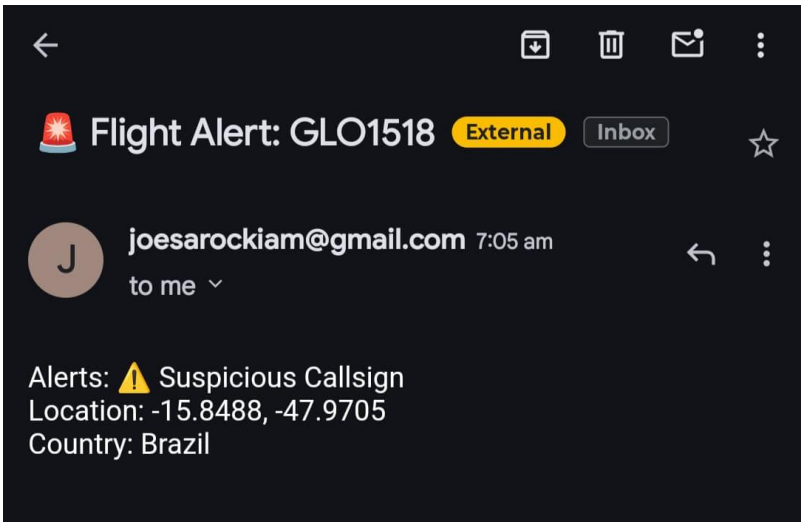
Longitude, Latitude, Altitude, Velocity (m/s) and Callsign



ALERT ENGINE:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Clg\sem-8\Lab\BI\flight-alert-engine> & C:/Users/JOE/AppData/Local/Programs/Python/Python313/python.exe d:/Clg/sem-8/Lab/BI/flight-alert-engine/alert_engine.py
Sent alert for UAL2296 : ['⚠Suspicious Callsign']
Sent alert for ERU957  : ['⚠Suspicious Callsign']
Sent alert for LPE2202 : ['⚠Suspicious Callsign']
```





7.Conclusion

Through detailed flight data analysis, we have gained meaningful insights into in-air flight operations, performance, and efficiency. By examining parameters such as altitude, speed, fuel usage, and flight paths, we can better understand patterns that impact safety, punctuality, and fuel economy. These insights help airlines make data-driven decisions to optimize routes, improve maintenance schedules, and enhance passenger experience. Ultimately, flight data analysis plays a critical role in improving operational efficiency, reducing costs, ensuring safety compliance, and paving the way for innovation in aviation technology.

Additionally, continuous monitoring and analysis of flight data support early detection of potential technical issues, enabling proactive maintenance and reducing the risk of in-flight failures. By leveraging advanced analytics and machine learning models on flight datasets, airlines and aviation authorities can predict trends, ensure regulatory compliance, and drive continuous improvements in flight operations. As the aviation industry evolves, the role of flight data analysis will become even more crucial in building safer, more efficient, and sustainable air travel.