Final Group Project Report

Course: Data Collection & Preparation
Project: OpenSky Airflow Kafka Data Pipeline

1. Team Members

- Smailbek Sagdi - 23B031438
- Akhatova Gulnur - 22B030312
- Nuridin Absattar - 22BB22B1536

2. Project Goal

The goal of this project is to design and implement a complete **streaming + batch data pipeline** that collects, processes, stores, and analyzes frequently updating real-world data.

The project demonstrates:

- API data ingestion
- Streaming using Kafka
- Batch processing with Apache Airflow
- Data storage in SQLite
- Daily analytical aggregation

The pipeline fully satisfies the assignment requirements of **three Airflow DAGs**:

1. API → Kafka (continuous ingestion)
2. Kafka → SQLite (hourly batch cleaning & storage)
3. SQLite → analytics → SQLite (daily aggregation)

Final_Project_Requirements

3. Data Source & API Justification

API Used: OpenSky Network REST API
Endpoint: https://opensky-network.org/api/states/all

Why this API was chosen:

- Provides **real-time aircraft state data**
- Updates frequently (multiple times per hour)
- Returns structured **JSON**
- Public, stable, and well-documented
- Contains meaningful real-world data (no synthetic or random values)

Data includes:

- Aircraft ICAO24 identifier
- Callsign
- Country of origin
- Latitude & longitude
- Altitude
- Velocity
- Heading
- Timestamps

This fully satisfies the API requirements defined in the assignment

Final_Project_Requirements

4. Architecture Overview

Pipeline Flow

OpenSky API

↓

Kafka topic (opensky_raw_states)

↓

Airflow DAGs

↓

SQLite (events table)

↓

Daily Aggregation

↓

SQLite (daily_summary table)

Technologies Used

- Python 3
- Apache Airflow 2.9
- Apache Kafka
- Docker & Docker Compose
- SQLite
- Pandas

5. Airflow DAGs Description

DAG 1: Continuous Ingestion (API → Kafka)

Name: opensky_api_to_kafka
Schedule: every 2 minutes

Responsibilities:

- Fetches live aircraft data from OpenSky API
- Wraps the response with metadata (ingested_at)
- Publishes raw JSON messages into Kafka topic opensky_raw_states

Purpose:

- Simulates continuous (pseudo-streaming) ingestion
- Ensures raw data is preserved without modification

DAG 2: Hourly Cleaning & Storage (Kafka → SQLite)

Name: opensky_kafka_to_sqlite_hourly
Schedule: @hourly

Responsibilities:

- Reads messages from Kafka topic
- Normalizes aircraft state vectors
- Cleans data:
    - Type casting to numeric
    - Filtering invalid coordinates
    - Removing impossible velocity and altitude values
    - Handling missing values
- Writes cleaned records into SQLite table events

DAG 3: Daily Analytics (SQLite → SQLite)

Name: opensky_sqlite_daily_summary
Schedule: @daily

Responsibilities:

- Reads data from events table
- Computes daily metrics:
    - Total events

    - Number of unique countries
    - Average and maximum velocity
    - Average and maximum altitude
- Stores results into daily_summary table

6. Kafka Topic Schema

Topic Name: opensky_raw_states

Each message contains:

```
{
  "ingested_at": "ISO-8601 timestamp",
  "payload": {
    "time": "API timestamp",
    "states": [
      [
        "icao24",
        "callsign",
        "origin_country",
        longitude,
        latitude,
        baro_altitude,
        velocity,
        heading,
        vertical_rate,
        geo_altitude,
        position_source
      ]
    ]
  }
}
```

Messages are stored **raw** and processed later by batch jobs.

7. Data Cleaning Rules

Cleaning is performed using **Pandas**, as required

Final_Project_Requirements

.

Applied rules:

- Drop rows without icao24
- Convert numeric fields using safe coercion
- Filter invalid values:

    - Velocity outside 0–400 m/s
    - Altitude outside -500 to 20,000 meters
- Normalize timestamps
- Strip callsign whitespace
- Convert boolean flags to integers

8. SQLite Database Schema

Table: events

Stores cleaned aircraft events.

Column	**Description**
ingested_at	Ingestion timestamp
api_time	API event time
icao24	Aircraft identifier
callsign	Flight callsign
origin_country	Country
latitude	Latitude
longitude	Longitude
baro_altitude	Barometric altitude
geo_altitude	Geometric altitude
velocity	Velocity
true_track	Heading
vertical_rate	Vertical speed
on_ground	Ground flag
position_source	Source type

Table: daily_summary

Stores aggregated daily analytics.

Column	**Description**
summary_date	Date

Column	Description
total_events	Total events
unique_countries	Country count
avg_velocity	Average velocity
max_velocity	Maximum velocity
avg_altitude	Average altitude
max_altitude	Maximum altitude

9. Results & Verification

After running all DAGs successfully:

- Events table contains **thousands of records**
- Daily summary table contains aggregated analytics

Verification example:

```
import sqlite3

with sqlite3.connect("/opt/airflow/data/app.db") as conn:
    cur = conn.cursor()
    cur.execute("SELECT COUNT(*) FROM events;")
    print(cur.fetchone())

    cur.execute("SELECT * FROM daily_summary ORDER BY summary_date DESC LIMIT 1;")
    print(cur.fetchone())
```

This confirms correct ingestion, storage, and aggregation.

10. Conclusion

This project successfully demonstrates a **complete data engineering pipeline** including:

- Real-time API ingestion
- Streaming with Kafka
- Batch processing with Airflow
- Data cleaning with Pandas
- Persistent storage in SQLite
- Daily analytics and summarization

All functional, architectural, and reporting requirements of the assignment have been fully met