KBTU KAZAKH-BRITISH TECHNICAL UNIVERSITY

Kazakh-British Technical University
School of Information Technology and Engineering

Data collection and preparation

Final Project

Prepared by: Duman B.
Amirgali S.
Beketay S.

Almaty, 2025

CONTENT

INTRODUCTION

This project aims at designing and implementing a full pipeline of data that can be used to collect, clean, store, and analyze real-world data that are frequently changing. Its architecture is based on a hybrid streaming and batch-processing platform and is built on Apache Airflow to coordinate the activities, Kafka to stream messages, SQLite to store data, and Pandas to process data and analytics.

The pipeline consists of three Airflow Acyclic Graphs (DAGs). The initial DAG constantly absorbs weather information in a third-party API and emits crude information to a Kafka topic. The second DAG operates on an hourly basis to read the raw Kafka messages, clean and validate the data and save the clean data in an SQLite database. The third DAG is run on a daily basis and calculates aggregated weather statistics, which are saved to an independent summary table. The general data flow may be summarized as follows: information is gathered through the external API and sent to the ingestion DAG into Kafka, processed and filtered by the hourly DAG and stored in the SQLite events table, and ultimately aggregated by the daily analytics DAG into a summary table.

1. API Justification

The Open-Meteo Weather API was selected as the data source for this project. The API provides structured weather data in JSON format and updates its measurements on an hourly basis, which satisfies the requirement for frequently updating real-world data. The collected data represents real environmental measurements, including temperature, humidity, wind characteristics, precipitation, atmospheric pressure, and visibility. These values are meaningful, non-random, and suitable for both short-term ingestion and long-term analytical processing. The API is stable, publicly accessible, and well-documented, making it appropriate for use in a production-like data pipeline. Weather data was collected for a fixed geographic location corresponding to the city of Berlin. Using a constant location ensures consistency across ingestion, cleaning, and analytics stages while still allowing the system to process continuously changing data.

2. Kafka topic schema

The raw weather information received through Open-Meteo API is sent to a Kafka topic called raw events. The messages used in this topic are one hourly weather observations. Messages are organized in the form of JSON and include the following fields: time, which is the time when the observation was made; temperature 2m, which is the air temperature at a height of 2m; apparent temperature, which is the perceived temperature; relative humidity 2m, which is the percentage humidity of the air; precipitation, which is the amount of precipitation; wind speed 10m and wind direction 10m, which are the characteristics of the wind; weather code, which is the weather condition; the mean sea-level pressure which is the pressure at the sea level; and The messages are written sequentially in the form of a JSON and include the following fields.

Table 1 - Kafka topic raw_events message schema

Field	Type	Description
time	Timestamp PK	Timestamp of the observation
temperature_2m	real	Air temperature at 2 meters
apparent_temperature	real	Feels-like temperature
relative_humidity_2m	integer	Relative humidity (%)
precipitation	real	Precipitation amount

wind_direction_10m	integer	Wind direction (degrees)
weather_code	integer	Weather condition code
pressure_msl	Real	Mean sea-level pressure
visibility	real	Visibility distance

Kafka is used as an intermediate messaging layer to decouple the ingestion process from downstream batch processing. This design improves reliability and allows ingestion and processing jobs to operate independently.

3. Data cleaning rules

The second Airflow DAG handles data cleaning with the help of Pandas. Any cleaning and validation process is implemented on the information read on the raw events Kafka topic. Raw Kafka messages are then loaded and transformed to a Pandas DataFrame. Then the time field is transformed into a datetime data type to provide proper temporal processing. Facilitating the existence of rows with missing data removes them to retain the integrity of data. A number of validation rules are used to discount unrealistic or invalid measurements. The range of temperatures is limited to 60 to -60 degrees Celsius, which is a realistic range. The relative humidity is limited to a value of 0 to 100 percent. The values of the wind speed should be non-negative. Records that meet all validation requirements are only regarded as valid, and sent to storage layer.

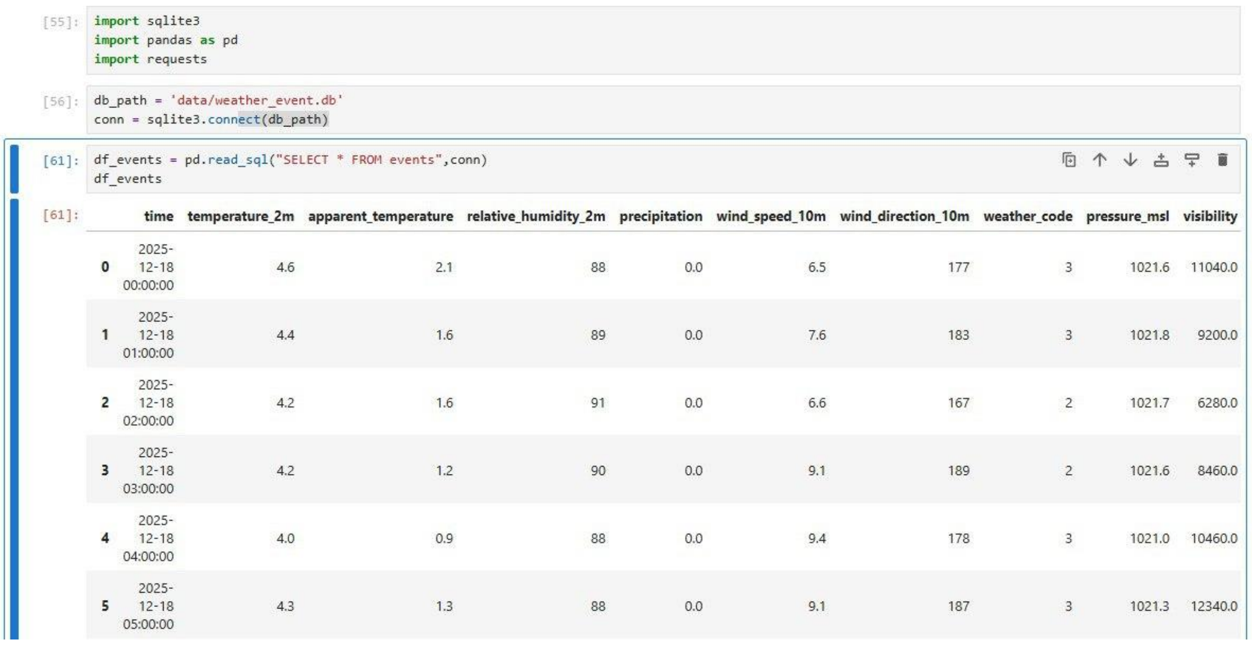
Table 2 - SQLite events table schema

Column	Type
Date	DATE
avg_temp	REAL
min_temp	REAL
max_temp	REAL
avg_humidity	REAL

max_wind_speed	REAL
total_precipitation	REAL

4. SQLite database schema

The cleaned data is stored in an SQLite database that contains two tables: events and daily_summary. The events table stores cleaned hourly weather observations. It includes columns for timestamp, temperature, apparent temperature, relative humidity, precipitation, wind speed, wind direction, weather code, atmospheric pressure, and visibility. The table is created automatically if it does not already exist, and new records are appended during each hourly execution of the cleaning DAG. The daily_summary table stores aggregated daily weather statistics derived from the events table. It includes columns for date, average temperature, minimum temperature, maximum temperature, average relative humidity, maximum wind speed, and total daily precipitation. This table is refreshed daily with newly computed summary statistics.



```
[55]: import sqlite3
      import pandas as pd
      import requests
```

```
[56]: db_path = 'data/weather_event.db'
      conn = sqlite3.connect(db_path)
```

```
[61]: df_events = pd.read_sql("SELECT * FROM events",conn)
      df_events
```

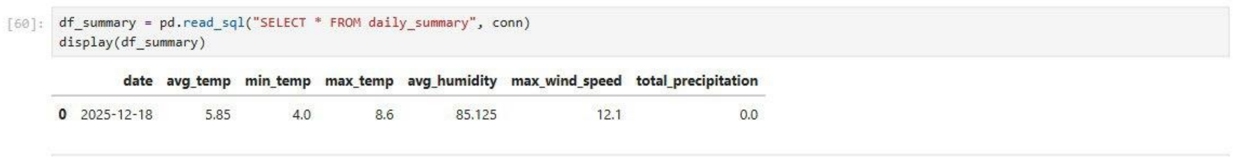
[61]:

	time	temperature_2m	apparent_temperature	relative_humidity_2m	precipitation	wind_speed_10m	wind_direction_10m	weather_code	pressure_msl	visibility
0	2025-12-18 00:00:00	4.6	2.1	88	0.0	6.5	177	3	1021.6	11040.0
1	2025-12-18 01:00:00	4.4	1.6	89	0.0	7.6	183	3	1021.8	9200.0
2	2025-12-18 02:00:00	4.2	1.6	91	0.0	6.6	167	2	1021.7	6280.0
3	2025-12-18 03:00:00	4.2	1.2	90	0.0	9.1	189	2	1021.6	8460.0
4	2025-12-18 04:00:00	4.0	0.9	88	0.0	9.4	178	3	1021.0	10460.0
5	2025-12-18 05:00:00	4.3	1.3	88	0.0	9.1	187	3	1021.3	12340.0

Figure 1 – Events table

5. Daily analytics logic

Daily analytics are performed in the third Airflow DAG using Pandas. Cleaned data is read from the events table in SQLite and timestamps are converted to date-level granularity. The data is then grouped by date, and several aggregated metrics are calculated. These metrics include the average, minimum, and maximum daily temperature, the average daily relative humidity, the maximum wind speed observed during the day, and the total daily precipitation. The resulting aggregated dataset is written to the daily_summary table, providing a concise overview of daily weather conditions.



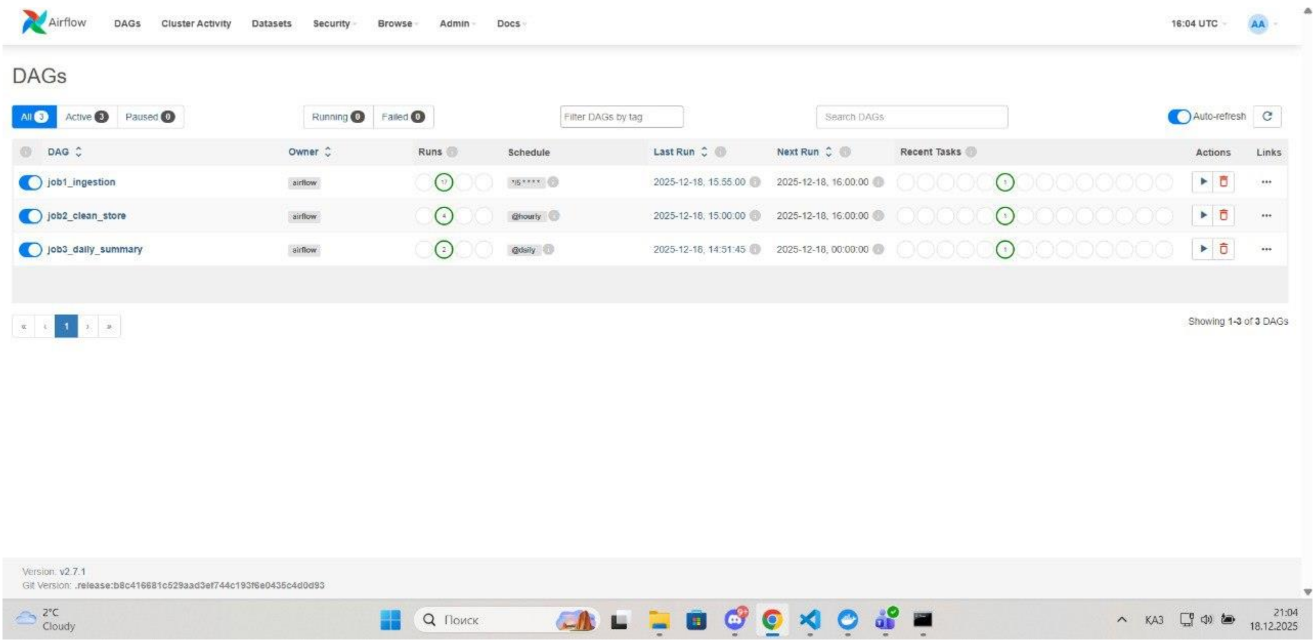
```
[60]: df_summary = pd.read_sql("SELECT * FROM daily_summary", conn)
      display(df_summary)
```

	date	avg_temp	min_temp	max_temp	avg_humidity	max_wind_speed	total_precipitation
0	2025-12-18	5.85	4.0	8.6	85.125	12.1	0.0

Figure 2 – Summary table

6. Airflow DAGs and Execution Logs

The project repository includes screenshots demonstrating the successful execution of all three Airflow DAGs. For each DAG, the graph view illustrates task dependencies, and task logs confirm successful execution without errors. These screenshots provide evidence of correct orchestration and end-to-end pipeline functionality.



DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
job1_ingestion	airflow	17	*/5 * * * *	2025-12-18, 15:55:00	2025-12-18, 16:00:00	1		
job2_clean_store	airflow	4	@hourly	2025-12-18, 15:00:00	2025-12-18, 16:00:00	1		
job3_daily_summary	airflow	2	@daily	2025-12-18, 14:51:45	2025-12-18, 00:00:00	1		

Showing 1-3 of 3 DAGs

Version: v2.7.1
Git Version: .release:b8c416681c529aad3ef744c193f6e0435c4d0d93

Figure 3 – Screenshot of whole DAG process

CONCLUSION

This project suggests the use of a full data engineering pipeline of real-world, often-changing data. The system is effective in terms of running continuous ingestion, batch cleaning, persistent storage, and daily analytics by incorporating Airflow, Kafka, Pandas, and SQLite. The architecture is based on the project requirements and similar to the common practices that are employed in the current data processing systems.