Data Collection and Preparation - Final Project

Members:

1. Amreyeva Alina – 22B031240
2. Kemel Merey – 22B030615
3. Serik Dinmukhammed – 22B030439

1. Project Goal

The objective of this project was to build a robust, scalable data pipeline capable of handling real-world, frequently updating data. We implemented a streaming ingestion with processing to analyze the Carbon Intensity of the UK National Grid. The system automates the collection, cleaning, storage, and analysis of environmental data using Apache Airflow, Kafka, and SQLite.

2. API Selection

Chosen API: Carbon Intensity UK (National Grid ESO) - https://api.carbonintensity.org.uk

We selected this API because it is well suited for a streaming data project. First, the data is updated **every 30 minutes**, which fits the requirement of frequently updated data. Second, the API provides **real-world and meaningful values**, such as forecasted and actual carbon intensity, which allows us to perform proper analysis and comparisons. Third, the API returns **clean and well-structured JSON data**, which makes it easy to process in Python. Finally, the API is **stable, well-documented, and free**, and it does not require complex authentication, which simplifies development.

3. Architecture Overview

The system is containerized using Docker and consists of three main stages controlled by Airflow DAGs:

1. **Ingestion Layer:** A producer script fetches data from the API every 30 seconds and buffers the raw JSON into Apache Kafka.
2. **Processing Layer:** An hourly job consumes messages from Kafka, cleans the data, and stores it in a relational database (SQLite).
3. **Analytics Layer:** A daily job aggregates the data to generate insights, specifically comparing Forecast vs. Actual intensity across different parts of the day.

Technologies Used:

- **Orchestration:** Apache Airflow (LocalExecutor with Postgres backend).
- **Messaging:** Apache Kafka & Zookeeper.
- **Storage:** SQLite.
- **Language:** Python (Pandas, Requests, Kafka-Python).

4. Implementation Details

4.1 Job 1: Continuous Ingestion (DAG 1)

- **Schedule:** Runs every 15 minutes.
- **Logic:** The Python script runs an internal loop for 14.5 minutes. Inside this loop, it queries the API every 30 seconds.
- **Fault Tolerance:** Implemented retry logic (10 attempts) for Kafka connections and error handling for API network timeouts.
- **Output:** Sends a JSON message to the Kafka topic raw_events.

4.2 Job 2: Cleaning & Storage (DAG 2)

- **Schedule:** @hourly
- **Cleaning Rules:**
    1. **Flattening:** Extracts nested fields (data.intensity.forecast, data.intensity.actual, data.intensity.index) from the raw JSON.
    2. **Type Conversion:** Ensures numerical values are stored as Integers.
    3. **Handling Nulls:** Stores actual intensity if available; handles potential missing values gracefully.
- **Output:** Inserts cleaned rows into the SQLite events table.

4.3 Job 3: Daily Analytics (DAG 3)

- **Schedule:** @daily
- **Logic:** Reads historical data from SQLite using Pandas.
- **Feature Engineering:** Adds a part_of_day column based on the timestamp (Morning: 05-12, Afternoon: 12-17, Evening: 17-21, Night: 21-05).
- **Aggregation:** Computes the Mean, Min, and Max intensity for each part of the day, handling cases where actual data might be missing by falling back to forecast values.
- **Output:** Writes results to the daily_summary table.

5. Data Schemas

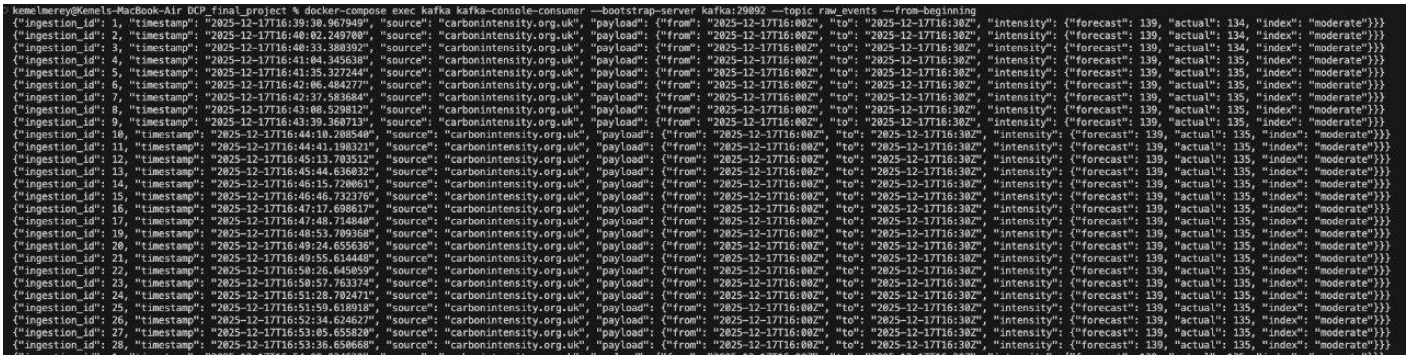
5.1 Kafka Topic Schema (raw_events)

The raw message sent to Kafka follows this JSON structure:

code JSON

```
{
  "ingestion_id": 123,
  "timestamp": "2025-12-17T10:00:00.123456",
  "source": "carbonintensity.org.uk",
  "payload": {
    "from": "2025-12-17T10:00Z",
    "to": "2025-12-17T10:30Z",
```

```
    "intensity": {
      "forecast": 260,
      "actual": 255,
      "index": "moderate"
    }
  }
}
```

5.2 SQLite Database Schema

Table 1: events (Cleaned Data)

Column	Type	Description
id	INTEGER PK	Auto-incrementing ID
ingestion_timestamp	TEXT	Timestamp of ingestion
forecast_intensity	INTEGER	Predicted carbon intensity
actual_intensity	INTEGER	Realized carbon intensity
index_intensity	TEXT	Category (e.g., 'moderate')
source	TEXT	API Source
created_at	TIMESTAMP	Record creation time

	cid	name	type	notnull	dflt_value	pk
1	0	id	INTEGER	0	<null>	1
2	1	ingestion_timestamp	TEXT	0	<null>	0
3	2	forecast_intensity	INTEGER	0	<null>	0
4	3	actual_intensity	INTEGER	0	<null>	0
5	4	index_intensity	TEXT	0	<null>	0
6	5	source	TEXT	0	<null>	0
7	6	created_at	TIMESTAMP	0	CURRENT_TIMESTAMP	0

Table 2: daily_summary (Aggregated Analytics)

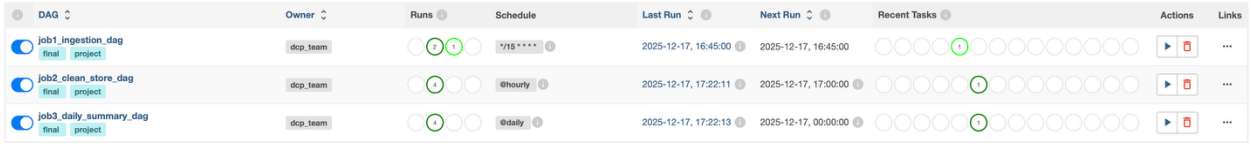
Column	Type	Description

summary_date	DATE	Date of the analysis
part_of_day	TEXT	Morning, Afternoon, Evening, Night
avg_forecast	REAL	Average predicted intensity
avg_actual	REAL	Average actual intensity
max_intensity	INTEGER	Max intensity recorded
min_intensity	INTEGER	Min intensity recorded

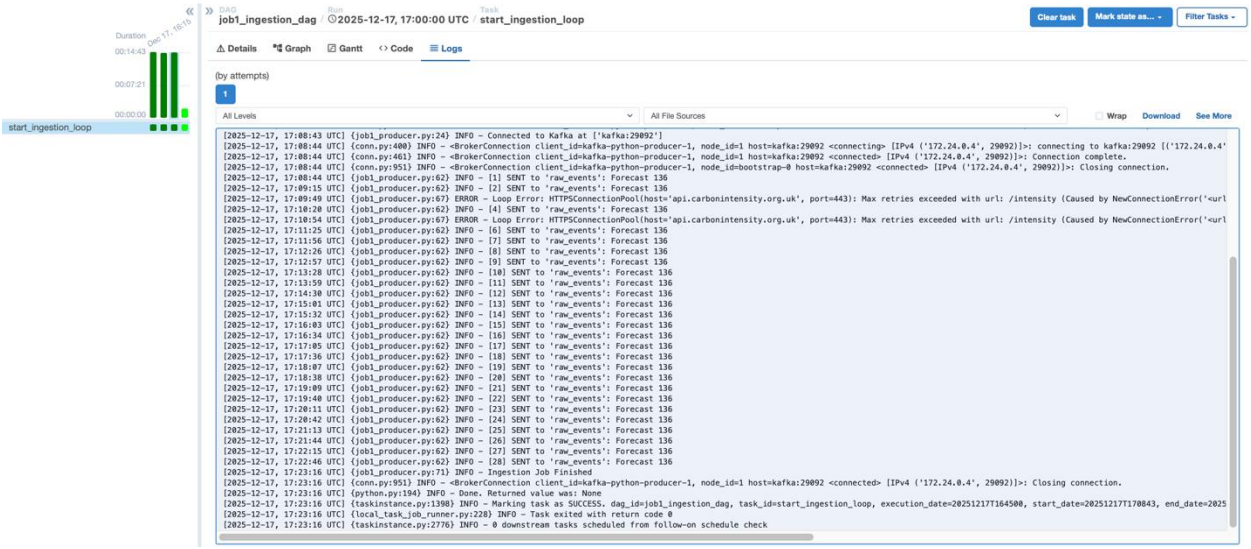
	cid	name	type	notnull	dflt_value	pk
1	0	summary_date	DATE	0	<null>	0
2	1	part_of_day	TEXT	0	<null>	0
3	2	avg_forecast	REAL	0	<null>	0
4	3	avg_actual	REAL	0	<null>	0
5	4	max_intensity	INTEGER	0	<null>	0
6	5	min_intensity	INTEGER	0	<null>	0

6. Verification

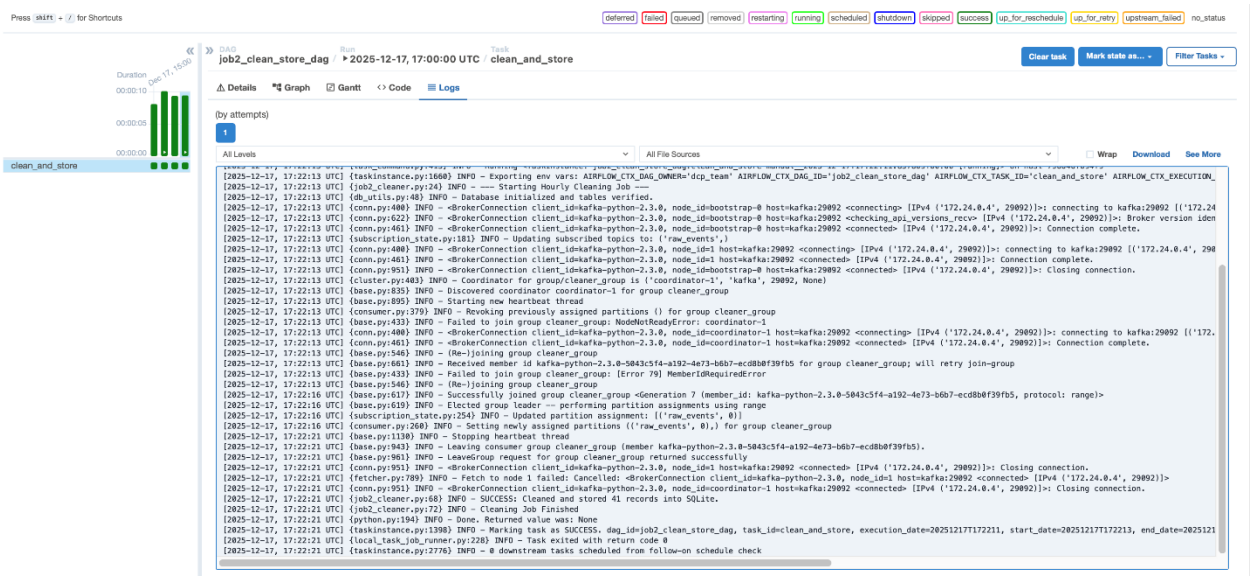
6.1 Airflow DAGs



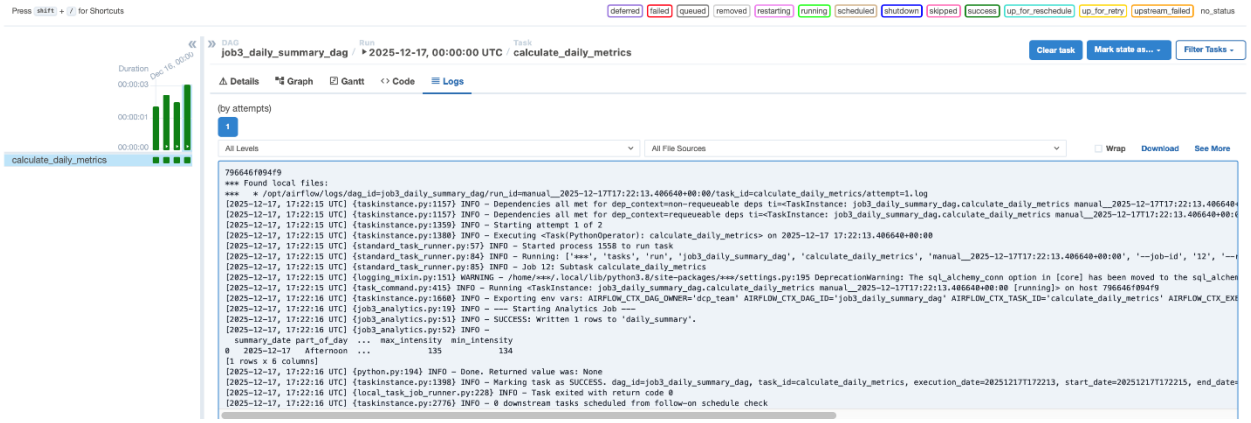
6.2 Job 1 Logs (Ingestion)



6.3 Job 2 Logs (Cleaning)



6.4 Job 3 Logs (Analytics)



6.5 Database Content & Visualization

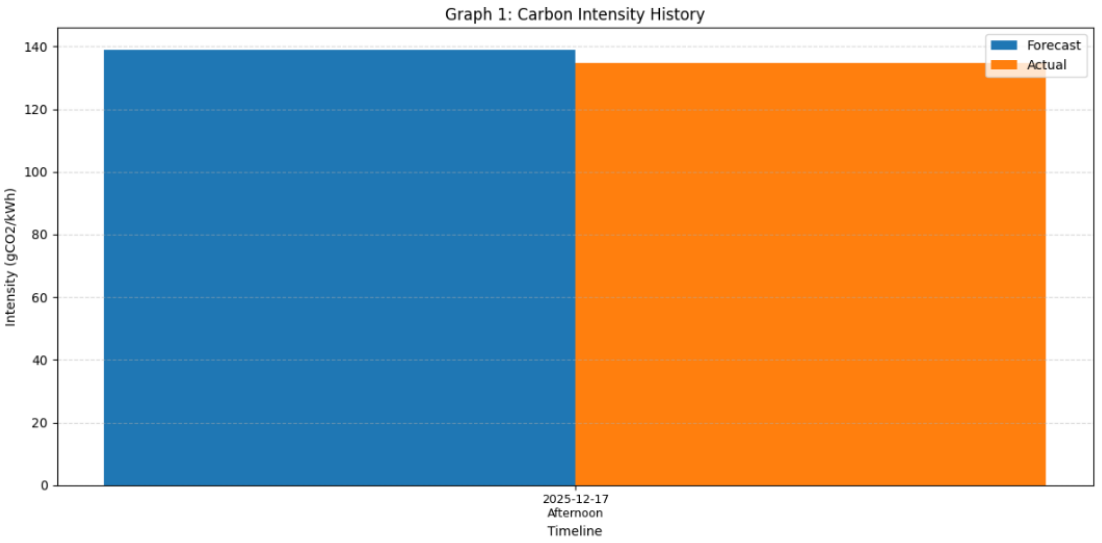
In [47]:

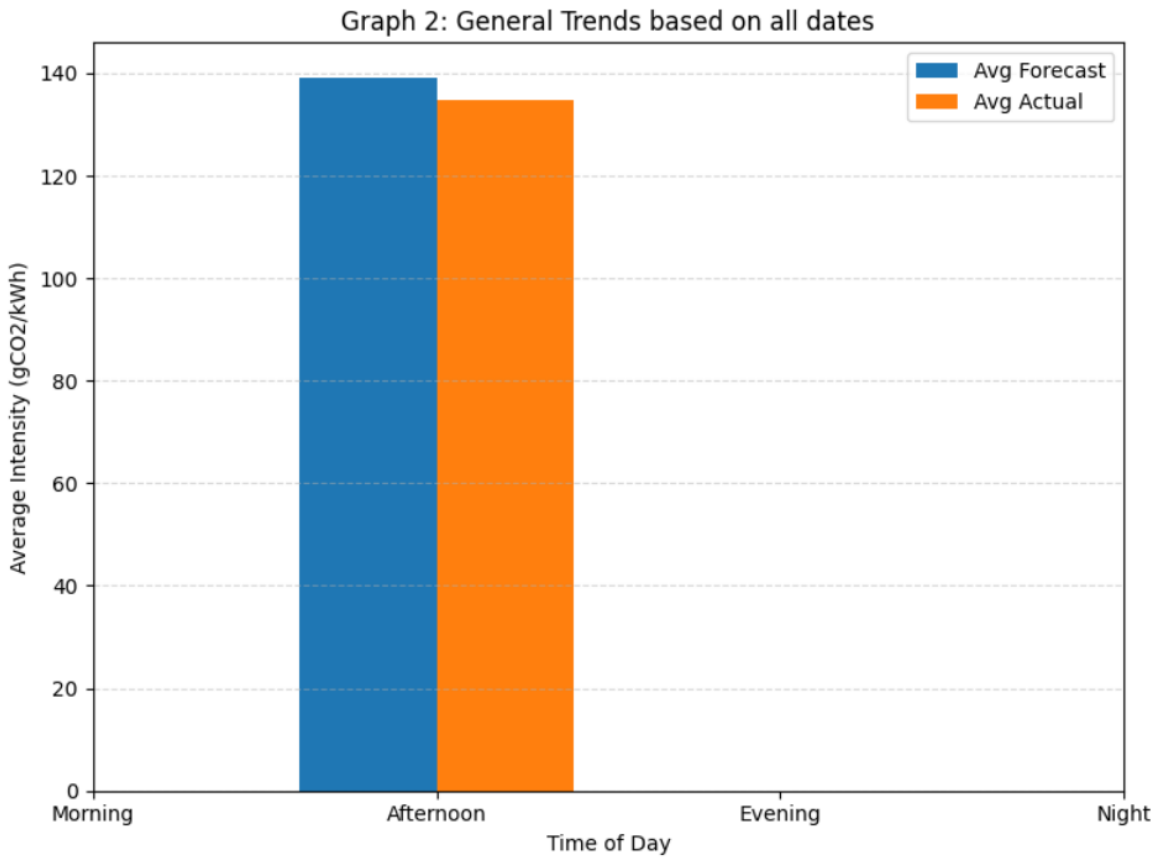
```
query_events = "SELECT * FROM events ORDER BY id DESC LIMIT 10"
df_events = pd.read_sql_query(query_events, conn)

print("Last 10 Recorded Events (Cleaned Data)")
display(df_events)
```

```
Last 10 Recorded Events (Cleaned Data)
```

	id	ingestion_timestamp	forecast_intensity	actual_intensity	index_intensity	source	created_at
0	80	2025-12-17T17:22:15.201889	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
1	79	2025-12-17T17:21:44.200970	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
2	78	2025-12-17T17:21:13.305457	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
3	77	2025-12-17T17:20:42.363494	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
4	76	2025-12-17T17:20:11.319593	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
5	75	2025-12-17T17:19:40.363466	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
6	74	2025-12-17T17:19:09.158542	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
7	73	2025-12-17T17:18:38.321542	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
8	72	2025-12-17T17:18:07.356356	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21
9	71	2025-12-17T17:17:36.220188	136	131	moderate	carbonintensity.org.uk	2025-12-17 17:22:21





7. Conclusion

In this project, we built an end-to-end data pipeline that fully meets the project requirements. The system reliably collects, processes, and stores frequently updated data using Apache Airflow, Kafka, and SQLite. The pipeline is resilient to temporary failures and ensures stable data flow from ingestion to analytics. The final analytical results provide clear insights into carbon intensity patterns during different parts of the day, demonstrating a practical real-world data engineering solution