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 Justification

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

We selected the Carbon Intensity API (UK National Grid).

Validity: The API updates every 30 minutes, satisfying the requirement for "frequently updating" data (at least every hour).

Data Quality: It provides meaningful, real-world environmental data (forecast vs. actual carbon intensity), not random numbers.

Technical Fit: The API is stable, well-documented, and returns structured JSON without requiring complex authentication.

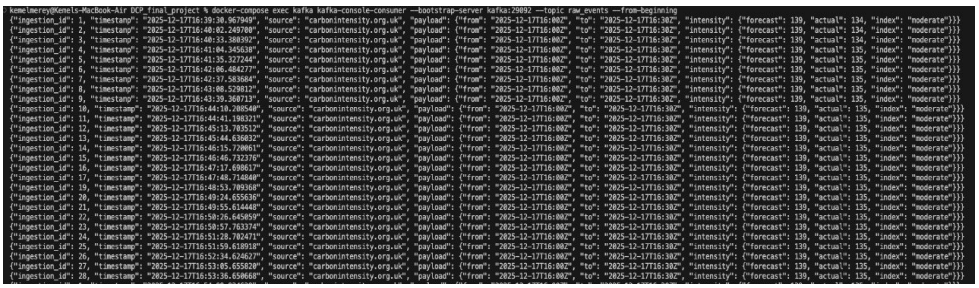
3. Data Schemas

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"
  }}}
```



4. Cleaning rules

All data cleaning is performed in DAG 2 using the Pandas library, as required.

Handling Missing Values: We use df.dropna(subset=['forecast_intensity']) to remove records where the primary data point is missing.

Removing Duplicates: We use df.drop_duplicates() to handle any message replays from Kafka (ensuring exactly-once processing in the DB).

Text Normalization: We apply .str.lower().str.strip() to the source and index_intensity columns to ensure consistency (e.g., "Moderate" becomes "moderate").
Type Conversion: We explicitly convert forecast_intensity and actual_intensity to numeric types using pd.to_numeric(..., errors='coerce') to prevent database type errors.

5. SQLite Database Schema

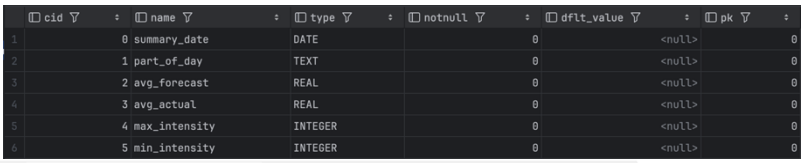
Table 1: events (Cleaned Data)

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)

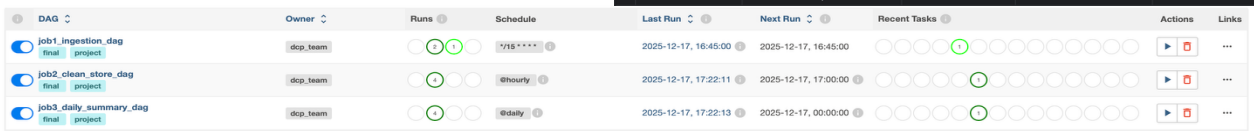
Column	Type	Description
summary_date	DATE	Date of the analysis
part_of_day	TEXT	Morning, Afternoon, Evening, Night
avg_forecast	REAL	Average predicted intensity
avg_actual	REAL	Average actual intensity
max_intensity	INTEGER	Max intensity recorded
min_intensity	INTEGER	Min intensity recorded



	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

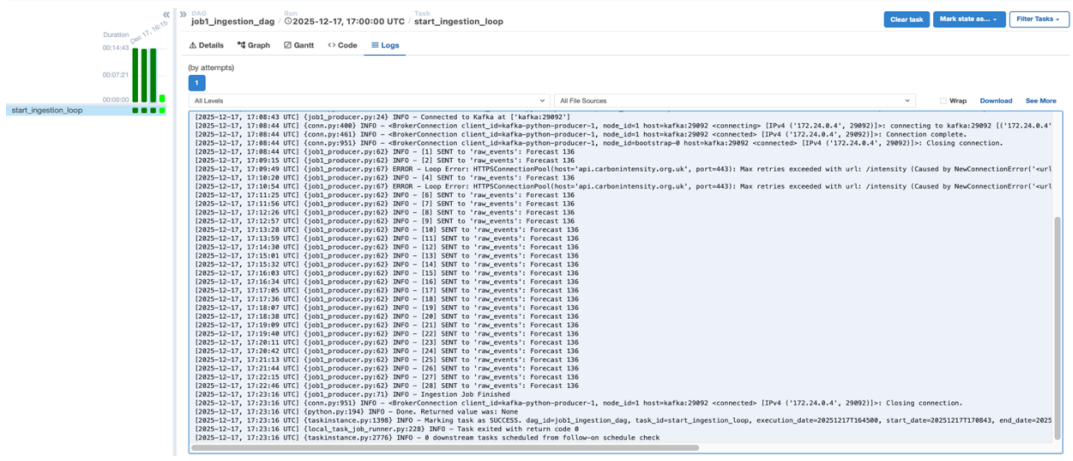
6. Verification

6.1 Airflow DAGs

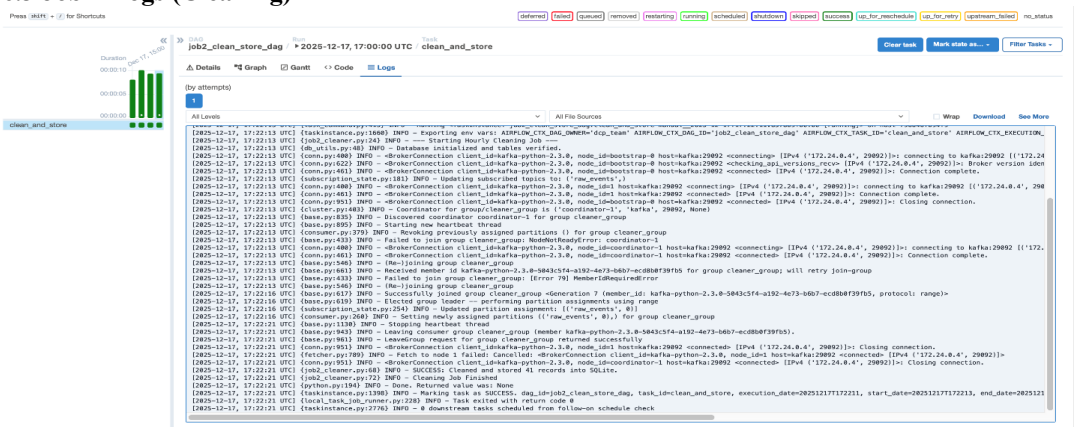


DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
job1_ingestion_dag final project	dcp_team	2 1	*/15 * * * *	2025-12-17, 16:45:00	2025-12-17, 16:45:00	1		
job2_clean_store_dag final project	dcp_team	4	@hourly	2025-12-17, 17:22:11	2025-12-17, 17:00:00	1		
job3_daily_summary_dag final project	dcp_team	4	@daily	2025-12-17, 17:22:13	2025-12-17, 00:00:00	1		

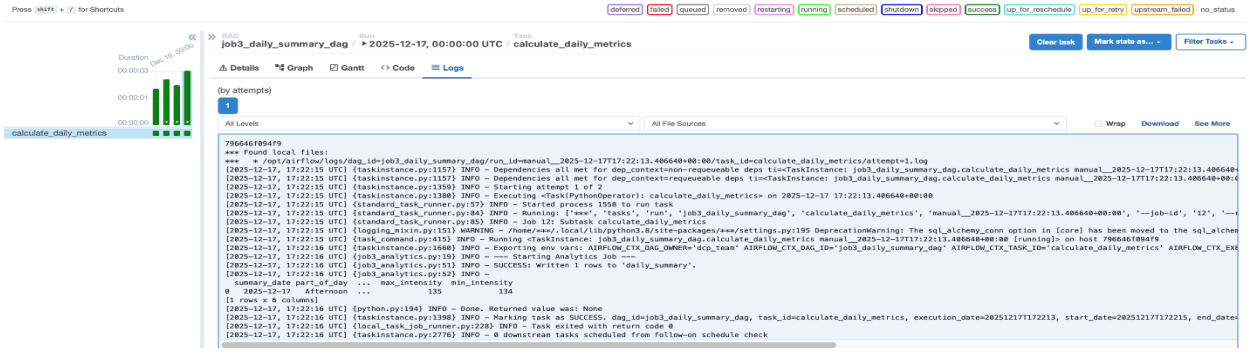
6.2 Job 1 Logs (Ingestion)



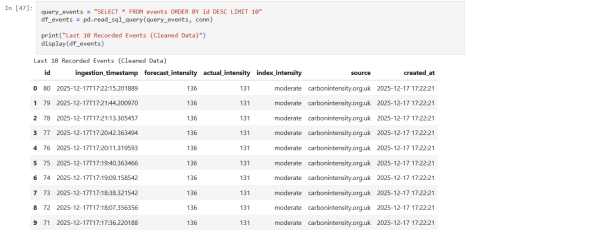
6.3 Job 2 Logs (Cleaning)



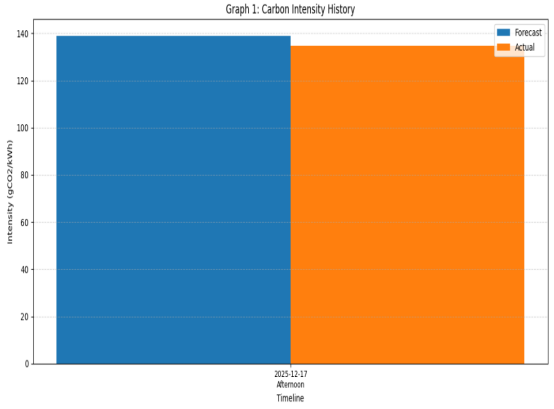
6.4 Job 3 Logs (Analytics)

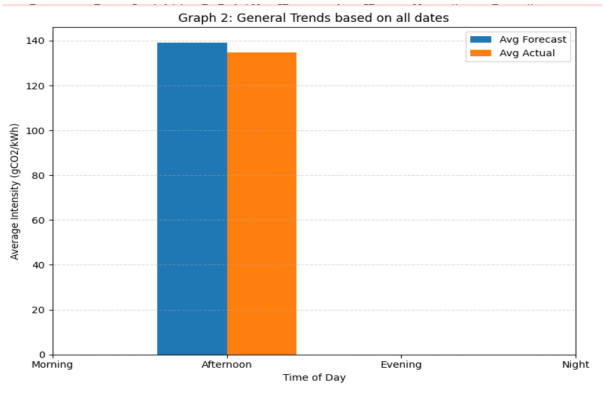


6.5 Database Content & Visualization



	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.