# Data Pipeline Report

## Team Members:

Serik Bakhramov 20B030585
Sadykova Yasmin 22B22B1540
Birlikzhanova Aruzhan 22B030329

**- Data Source (API) Source:**
**API - https://api.eia.gov/v2/electricity/rto/fuel-type-data/data/**
Public API with power generation data API returns data in JSON format.
Sample data:

- period — date and hour
- respondent — electricity market operator
- fueltype — fuel type
- value — electricity volume

The API is read-only and requires an API key for access.

**- Virtual environment and libraries**
A Python virtual environment was created to isolate project dependencies:
python -m venv venv
venv/bin/activate (Windows: venv\Scripts\activate)
All required libraries were installed from the requirements.txt file:
pip install -r requirements.txt
The requirements.txt file contains all dependencies necessary to run
Kafka producers/consumers, data processing with Pandas, and Airflow DAGs.

**- Kafka (data streaming)** Kafka launch Kafka was installed via Homebrew and launched: brew services start kafka Creating a Kafka topic kafka-topics --create \ --topic raw_electricity \ --bootstrap-server localhost:9092 \ --partitions 1 \ --replication-factor 1 Kafka Verification kafka-console-consumer \ --topic raw_electricity \ --from-beginning \ --bootstrap-server localhost:9092

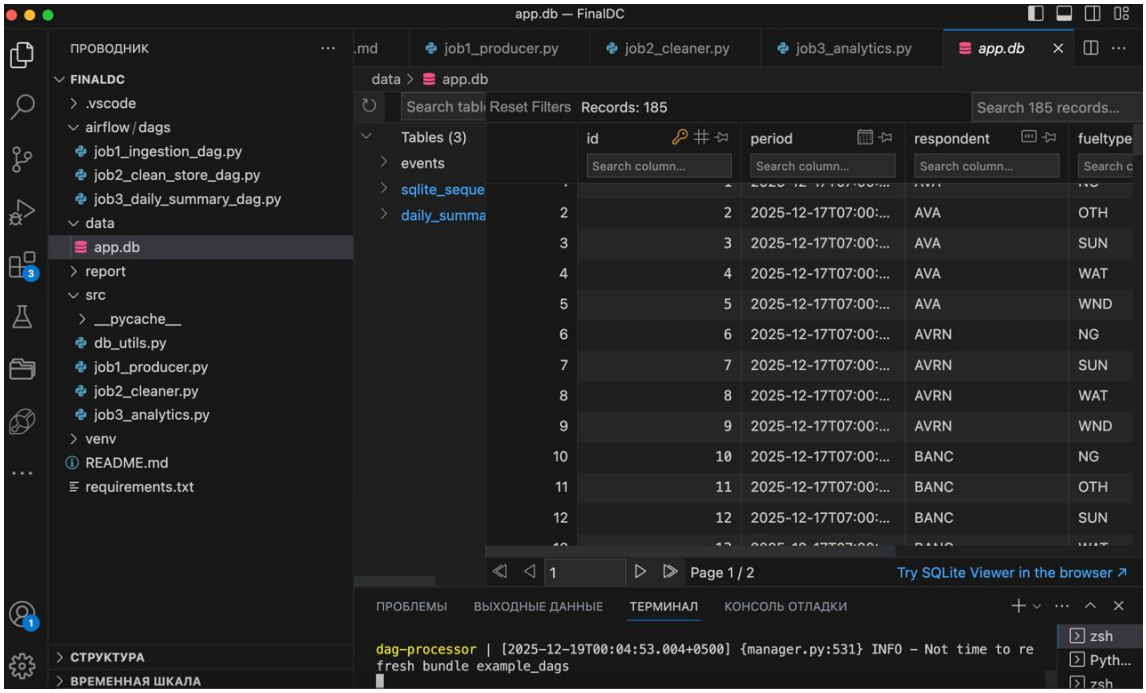
**- Job 1 — Ingestion** (API → Kafka) File src/job1_producer.py What does * Makes a request to the API * Receives JSON * Sends each record to Kafka topic raw_electricity Manual verification python src/job1_producer.py If consumer shows JSON— ingestion works.

**- Job 2 — Cleaning & Storage** (Kafka → SQLite) Files src/job2_cleaner.py src/db_utils.py What does * Reads data from Kafka * Clears data: * converts dates * converts value to a number * deletes incorrect values * Creates an SQLite database * Saves data to the events table Manual launch python src/job2_cleaner.py Checking the database sqlite3 data/app.db SELECT COUNT(*) FROM events;

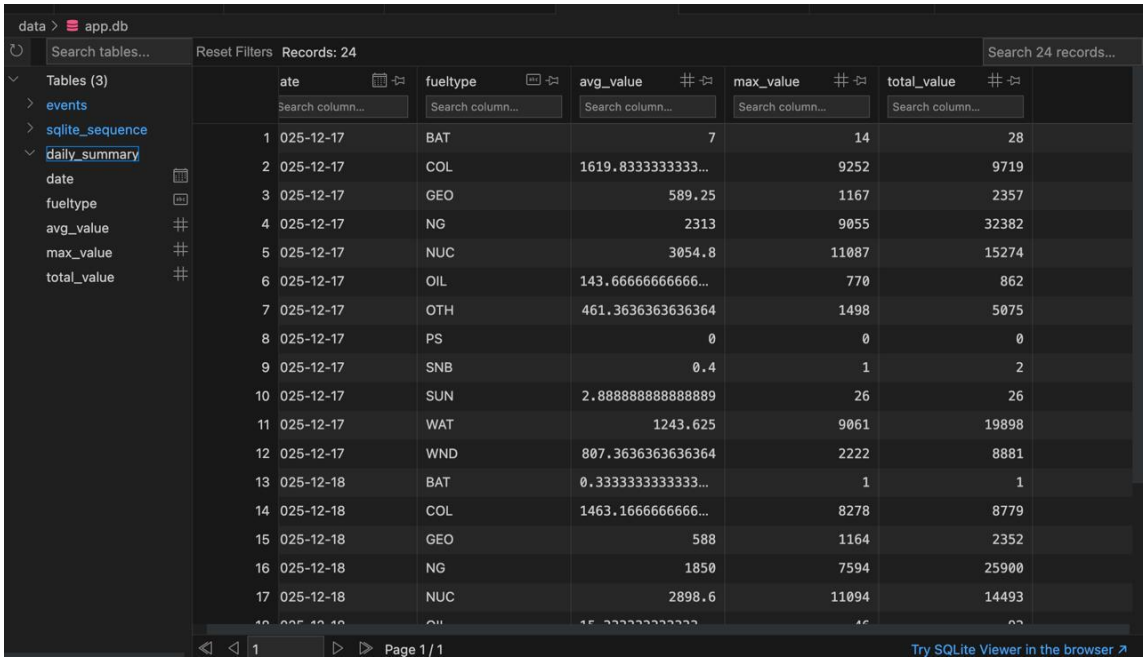
**- Job 3 — Analytics** (SQLite → Aggregation) File src/job3_analytics.py What does * Reads data from the events table * Groups by: * date * type of fuel * Counts: * average value * maximum * the amount * Saves the result to the daily_summary table Manual launch python src/job3_analytics.py

**- Airflow (orchestration)** DAG files airflow/dags/job1_ingestion_dag.py airflow/dags/job2_clean_store_dag.py airflow/dags/job3_daily_summary_dag.py DAG 1 — job1_ingestion Purpose: continuous data loading Schedule: None (starts manually) Reason: ingestion is a continuous pipeline (by assignment condition). DAG 2 — job2_clean_store Purpose: cleaning and preservation Schedule: 0 * * * * (every hour) DAG 3 — job3_daily_summary Purpose: Schedule Analytics: 0 0 * * * (once a day)

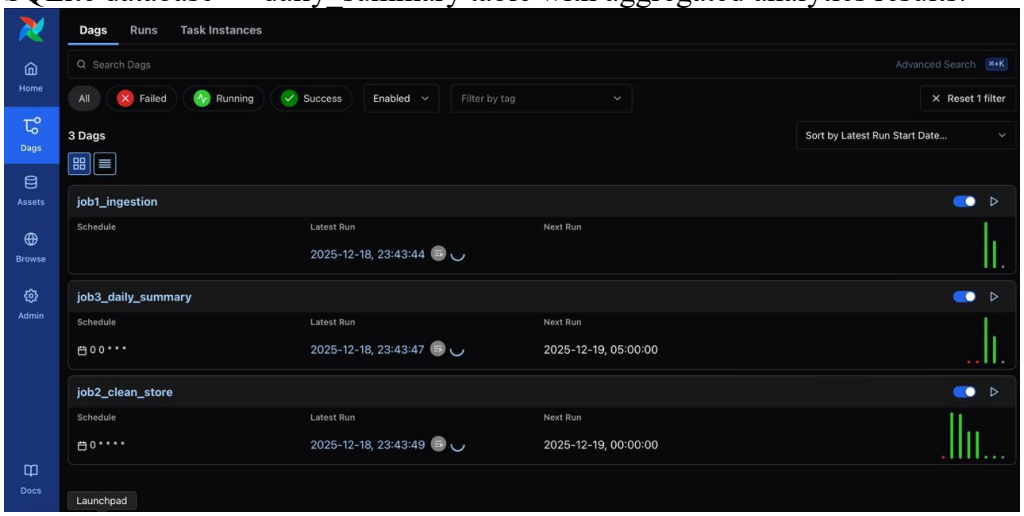
**- The launch of Airflow** Initialization airflow db init Launch airflow standalone Web UI http://localhost:8080
**- Checking in the Airflow UI** Checked in the UI: * DAGi are visible * job2 and job3 are running on schedule * job1 starts manually * the data is successfully traversing the entire pipeline

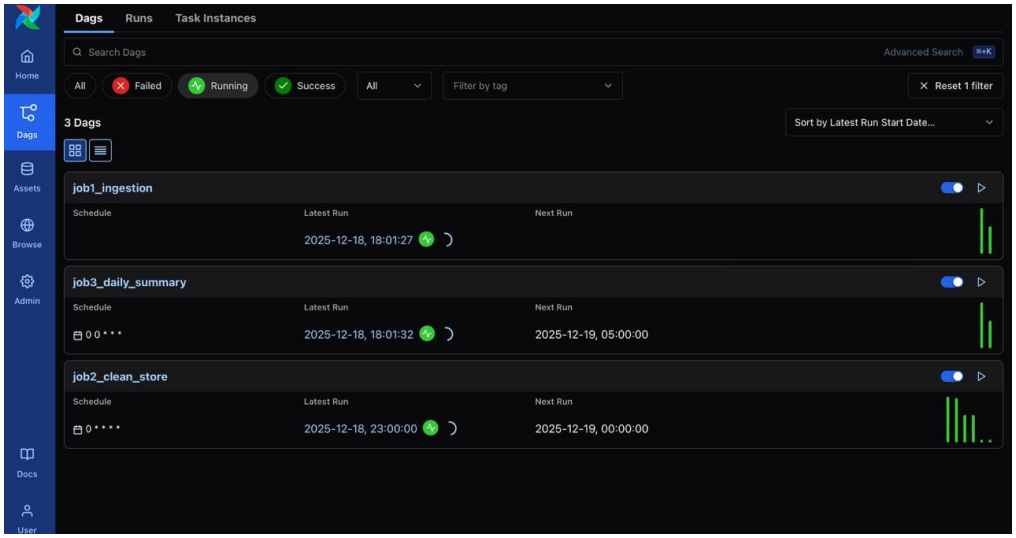


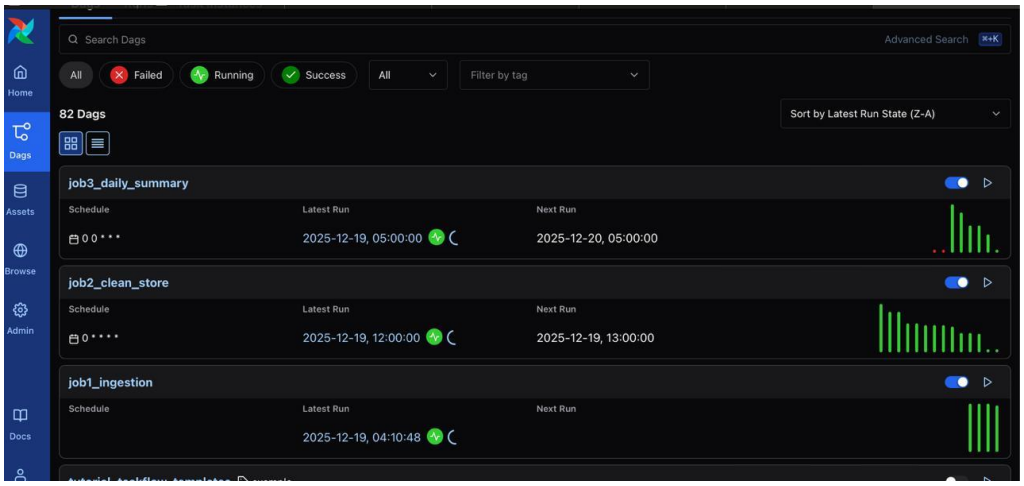

SQLite database — events table with cleaned data.




SQLite database — daily_summary table with aggregated analytics results.






Airflow UI showing all DAGs and their successful runs
Successful DAG execution in Airflow
Successful task log — data cleaning and storage
Successful task log — daily analytics aggregation