Final Group Project Report

1. Team Members

- Student 1: Amangeldi Madina
- Student 2: Serikbayeva Aruzhan
- Student 3: Tolegen Nursultan

2. Project Overview

The goal of this project is to design and implement a complete **streaming + batch data pipeline** using **Apache Airflow**, **Apache Kafka**, and **SQLite**. The pipeline demonstrates the full lifecycle of frequently updated real-world data: ingestion, cleaning, storage, and analytical aggregation.

The system consists of **three Airflow DAGs**:

1. Continuous ingestion of live cryptocurrency data from an external API into Kafka
2. Hourly batch processing to clean and store data in SQLite
3. Daily analytics job to compute aggregated metrics and store them in a summary table

3. API Selection and Justification

Chosen API

WazirX Cryptocurrency Tickers API
Endpoint: https://api.wazirx.com/api/v2/tickers

Justification

The WazirX API was selected because it fully satisfies all project requirements:

- **Frequently updated**: Cryptocurrency prices change continuously and update multiple times per hour
- **Stable and documented**: Publicly available and widely used exchange API
- **Structured JSON format**: Returns well-structured JSON objects

Each API call returns the latest ticker information for multiple trading pairs, including price, volume, and timestamps, making it ideal for streaming and analytics use cases.

4. System Architecture Overview

The architecture follows a **Lambda-style pipeline** combining pseudo-streaming and batch processing.

Data Flow

API → Airflow DAG 1 → Kafka (raw_events) → Airflow DAG 2 → SQLite (events) → Airflow DAG 3 → SQLite (daily_summary)

5. DAG 1 – Continuous Ingestion Job

Purpose

Continuously ingest raw cryptocurrency ticker data from the WazirX API and publish it to a Kafka topic.

Key Characteristics

- Implemented as a long-running Airflow task
- Fetches data every 30–60 seconds
- Sends raw JSON responses directly to Kafka without transformation

Kafka Topic

- **Topic name**: raw_events
- **Message format**: Raw JSON per API response

Flow

API → DAG 1 → Kafka (raw_events)

6. Kafka Topic Schema

Each Kafka message contains:

- symbol (string): Trading pair (e.g., btcusdt)
- last (string): Last traded price
- open (string): Opening price
- high (string): Highest price
- low (string): Lowest price
- volume (string): Trading volume
- quoteVolume (string): Quote asset volume
- at (integer): Unix timestamp

Messages are stored as raw JSON to preserve original data integrity.

7. DAG 2 – Hourly Cleaning and Storage Job

Purpose

Consume raw messages from Kafka, clean and normalize the data, and store it in SQLite.

Schedule

- @hourly

Cleaning Rules

- Convert numeric fields from string to float

- Convert Unix timestamp to datetime
- Remove records with missing or invalid prices
- Filter out zero or negative volume values
- Standardize symbol names to lowercase

Output Table

- **Database**: SQLite
- **Table**: events

Flow

Kafka → DAG 2 → Data Cleaning → SQLite (events)

8. SQLite Schema

events Table

Stores cleaned, structured cryptocurrency data.

Column Name	Type	Description
id	INTEGER (PK)	Auto-increment primary key
symbol	TEXT	Trading pair symbol
last_price	REAL	Last traded price
open_price	REAL	Opening price
high_price	REAL	Highest price
low_price	REAL	Lowest price
volume	REAL	Trading volume
quote_volume	REAL	Quote asset volume
event_time	TIMESTAMP	Event timestamp

9. DAG 3 – Daily Analytics Job

Purpose

Compute aggregated daily analytics from cleaned data stored in SQLite.

Schedule

- @daily

Analytics Metrics

- Average price per symbol
- Minimum and maximum price per symbol
- Total daily trading volume

- Number of records per symbol

Output Table

- **Table**: daily_summary

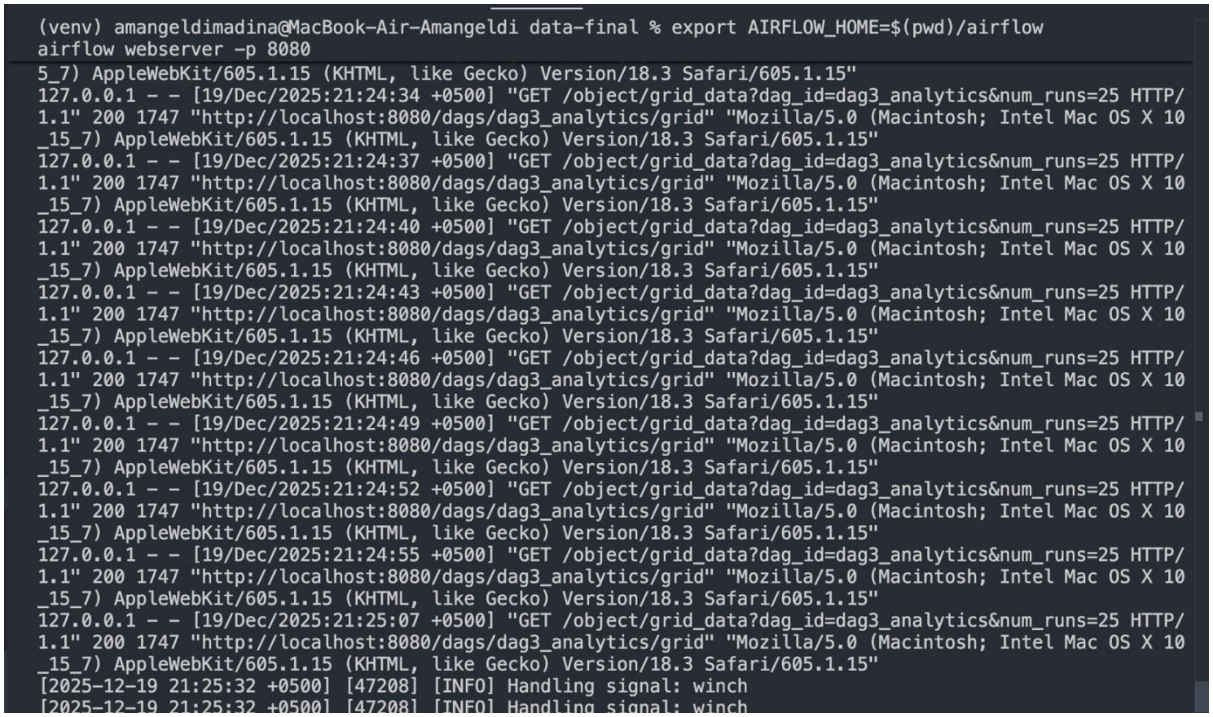
Flow

SQLite (events) → DAG 3 → Analytics → SQLite (daily_summary)

10. daily_summary Table Schema

Column Name	Type	Description
summary_date	DATE	Aggregation date
symbol	TEXT	Trading pair symbol
avg_price	REAL	Average daily price
min_price	REAL	Minimum daily price
max_price	REAL	Maximum daily price
total_volume	REAL	Total daily volume
record_count	INTEGER	Number of records

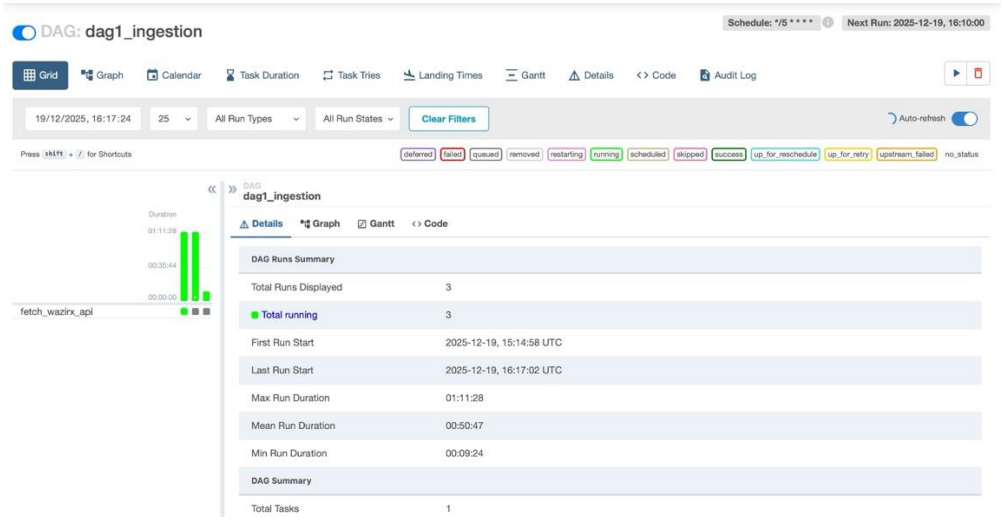
11. Airflow DAG Validation

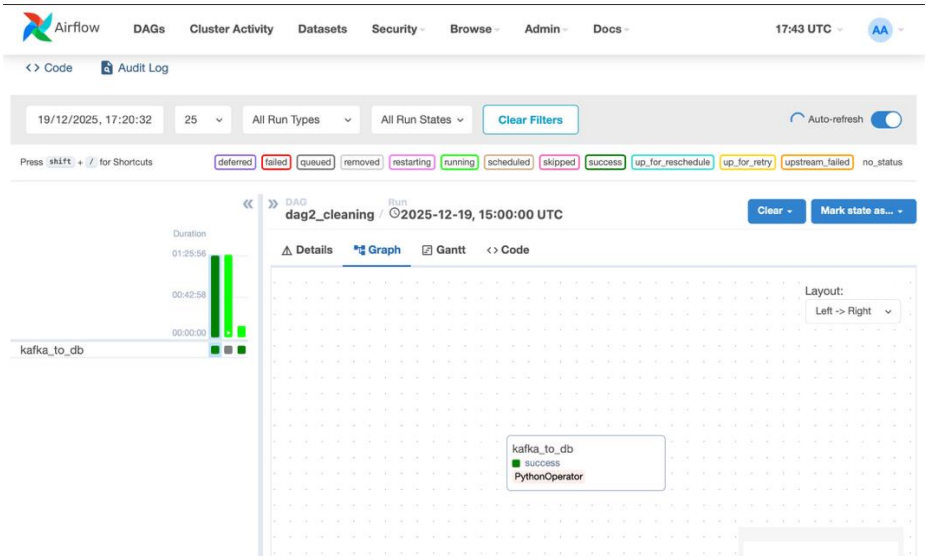


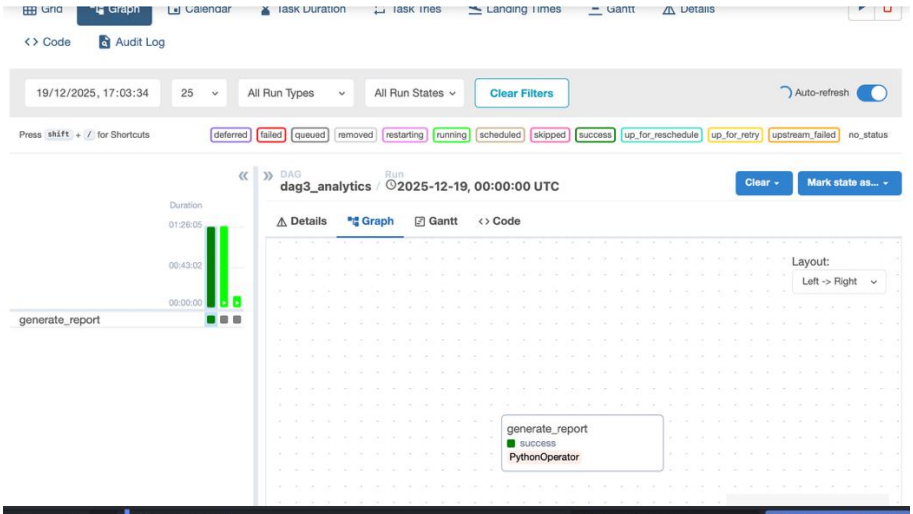
All three DAGs were successfully executed in Airflow:

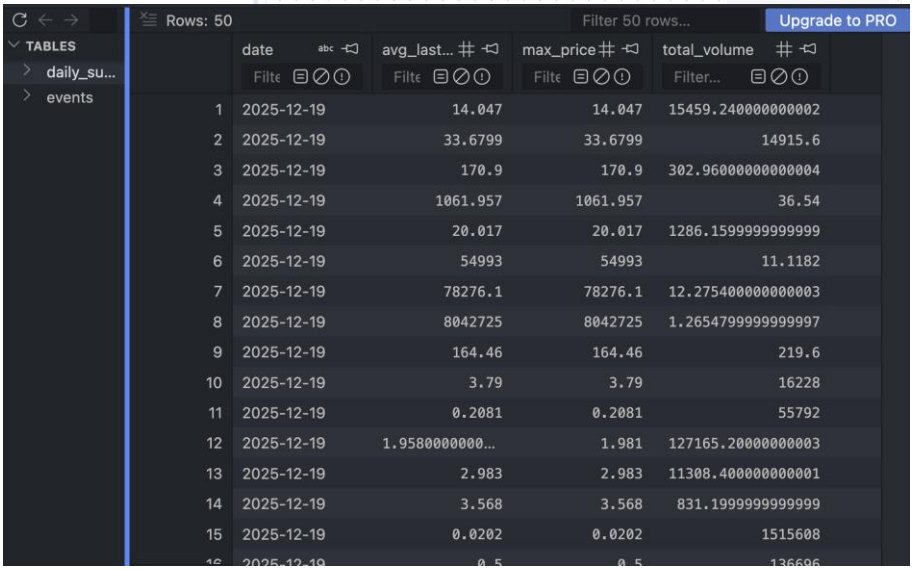
- DAG 1 runs continuously and publishes messages to Kafka

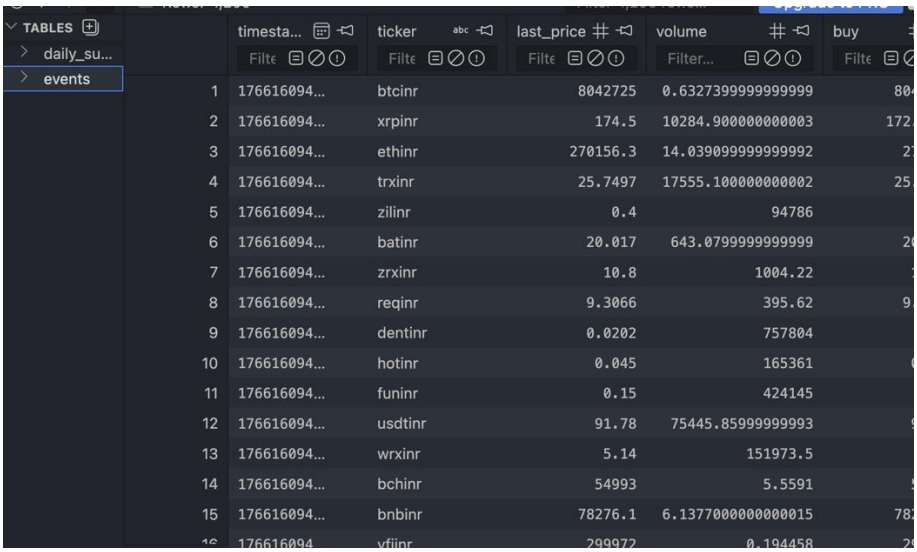
- DAG 2 executes hourly and writes cleaned data to SQLite
- DAG 3 executes daily and generates aggregated summaries











12. Conclusion

This project demonstrates a complete end-to-end data engineering pipeline using real-time cryptocurrency data. The system successfully integrates API ingestion, Kafka-based

streaming, batch data processing, persistent storage, and analytical aggregation, fulfilling all course requirements and showcasing practical data engineering skills.