

# 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

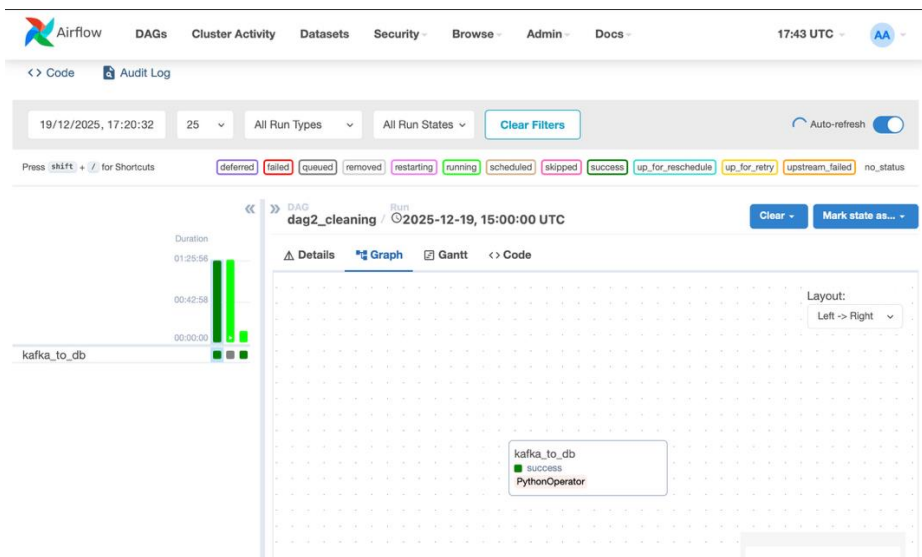
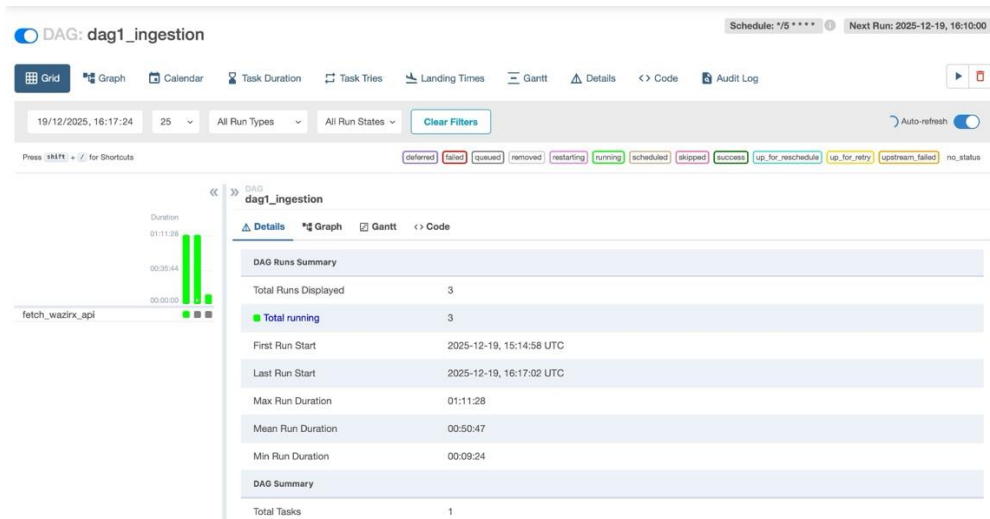
## 11. Airflow DAG Validation

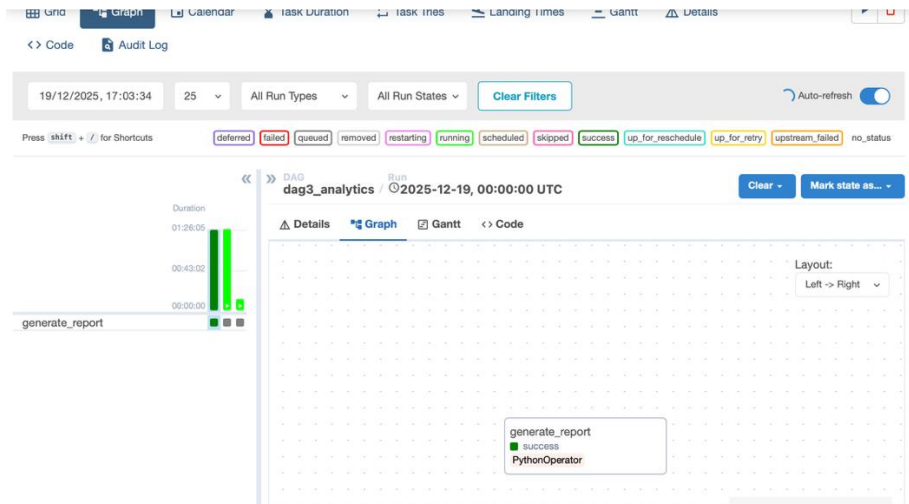
```
(venv) amangeldimadina@MacBook-Air-Amangeldi data-final % export AIRFLOW_HOME=$(pwd)/airflow
airflow webserver -p 8080
5.7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:34 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:37 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:40 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:43 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:46 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:49 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:52 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:24:55 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
127.0.0.1 - - [19/Dec/2025:21:25:07 +0500] "GET /object/grid_data?dag_id=dag3_analytics&num_runs=25 HTTP/
1.1" 200 1747 "http://localhost:8080/dags/dag3_analytics/grid" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10
_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.3 Safari/605.1.15"
[2025-12-19 21:25:32 +0500] [47208] [INFO] Handling signal: winch
[2025-12-19 21:25:32 +0500] [47208] [INFO] Handling signal: winch
```

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





←

→

TABLES

> daily\_su...

> events

50 Rows: 50

Filter 50 rows...

Upgrade to PRO

	date	avg_last...	max_price	total_volume
	Filter	Filter	Filter	Filter...
1	2025-12-19	14.047	14.047	15459.240000000002
2	2025-12-19	33.6799	33.6799	14915.6
3	2025-12-19	170.9	170.9	302.9600000000004
4	2025-12-19	1061.957	1061.957	36.54
5	2025-12-19	20.017	20.017	1286.1599999999999
6	2025-12-19	54993	54993	11.1182
7	2025-12-19	78276.1	78276.1	12.275400000000003
8	2025-12-19	8042725	8042725	1.2654799999999997
9	2025-12-19	164.46	164.46	219.6
10	2025-12-19	3.79	3.79	16228
11	2025-12-19	0.2081	0.2081	55792
12	2025-12-19	1.9580000000...	1.981	127165.20000000003
13	2025-12-19	2.983	2.983	11308.400000000001
14	2025-12-19	3.568	3.568	831.1999999999999
15	2025-12-19	0.0202	0.0202	1515608
16	2025-12-19	0.5	0.5	136606

TABLES		timesta...	ticker	last_price	volume	buy
		Filter...	Filter...	Filter...	Filter...	Filter...
> daily_su...						
> events						
	1	176616094...	btclnr	8042725	0.6327399999999999	80...
	2	176616094...	xrpinr	174.5	10284.900000000003	172...
	3	176616094...	ethlnr	270156.3	14.039099999999992	27...
	4	176616094...	trxinr	25.7497	17555.100000000002	25...
	5	176616094...	zilinr	0.4	94786	
	6	176616094...	batlnr	20.017	643.0799999999999	20...
	7	176616094...	zrxlnr	10.8	1004.22	
	8	176616094...	reqlnr	9.3066	395.62	9...
	9	176616094...	dentlnr	0.0202	757804	
	10	176616094...	hotlnr	0.045	165361	
	11	176616094...	funlnr	0.15	424145	
	12	176616094...	usdtlnr	91.78	75445.85999999993	
	13	176616094...	wrxlnr	5.14	151973.5	
	14	176616094...	bchlnr	54993	5.5591	
	15	176616094...	bnblnr	78276.1	6.1377000000000015	78...
	16	176616094...	vtlnr	290972	0.194458	29...

## 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.