Final Project Report

1. Team members

Temirbayev Zhassulan
Kumbayeva Ingkar
Duisenbek Gulzat

2. API justification

We have selected API for binance, which is a trading platform, that provides real time data about cryptocurrency prices. Since the market is very volatile, prices are changing frequently, about once or twice a minute. It is a trustworthy source, so the data can be considered real. Data is sent in a JSON format which makes it easier to work with.

https://api.binance.com/api/v3/ticker/24hr?symbol=BTCUSDT

3. Kafka topic schema

raw_events - stores unprocessed API responses

```
{
  "event_id": "uuid",
  "source": "binance",
  "ingested_at": "2025-12-19T14:26:46.552956+00:00",
  "payload": {
    "symbol": "BTCUSDT",
    "lastPrice": "88006.09",
    "volume": "27033.33",
    "openTime": 1766068006013,
    "closeTime": 1766154406013,
  }
}
```

4. Cleaning rules

- Drop rows missing symbol, price, or event_time
- Drop rows where price ≤ 0
- Remove duplicate event_id
- Remove duplicate (symbol, event_time) combinations
- Convert numeric columns to proper types (float/int)
- Standardize symbol format: uppercase, trimmed

5. SQLite Schema

```
        events (
    event_id TEXT PRIMARY KEY,
    symbol TEXT NOT NULL,
    price REAL NOT NULL,
    event_time TEXT NOT NULL,
    source TEXT NOT NULL,
    ingested_at TEXT NOT NULL,

    priceChange REAL,
    priceChangePercent REAL,
    weightedAvgPrice REAL,
    prevClosePrice REAL,
    lastPrice REAL,
    lastQty REAL,
    bidPrice REAL,
    bidQty REAL,
    askPrice REAL,
    askQty REAL,
    openPrice REAL,
    highPrice REAL,
    lowPrice REAL,
    volume REAL,
    quoteVolume REAL,
    openTime INTEGER,
    closeTime INTEGER,
    firstId INTEGER,
    lastId INTEGER,
    count INTEGER
);
```

```
daily_summary (
    day TEXT NOT NULL,
    symbol TEXT NOT NULL,
    n_records INTEGER,
    first_event_time TEXT,
    last_event_time TEXT,
    open_price REAL,
    high_price REAL,
```

```
    low_price REAL,
    close_price REAL,
    avg_price REAL,
    min_price REAL,
    max_price REAL,
    avg_volume REAL,
    sum_volume REAL,
    avg_quote_volume REAL,
    sum_quote_volume REAL,
    avg_count REAL,
    sum_count REAL,
    avg_price_change REAL,
    avg_price_change_pct REAL,
    avg_spread REAL,
    avg_spread_pct REAL,
    daily_return REAL,
    log_return REAL,
    volatility REAL,
    computed_at TEXT NOT NULL,
    PRIMARY KEY (day, symbol)
);
```

6. Screenshots

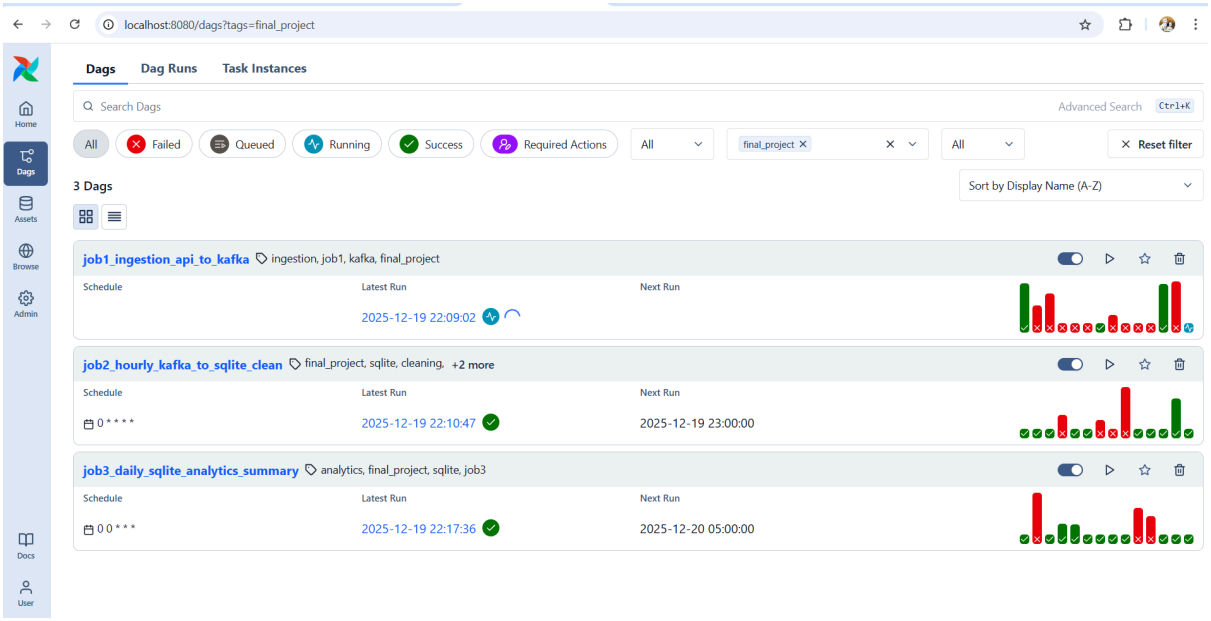

Figure 1. Overview of Airflow DAGs Execution

Figure 1 shows the Apache Airflow web interface displaying all DAGs related to the final_project pipeline.

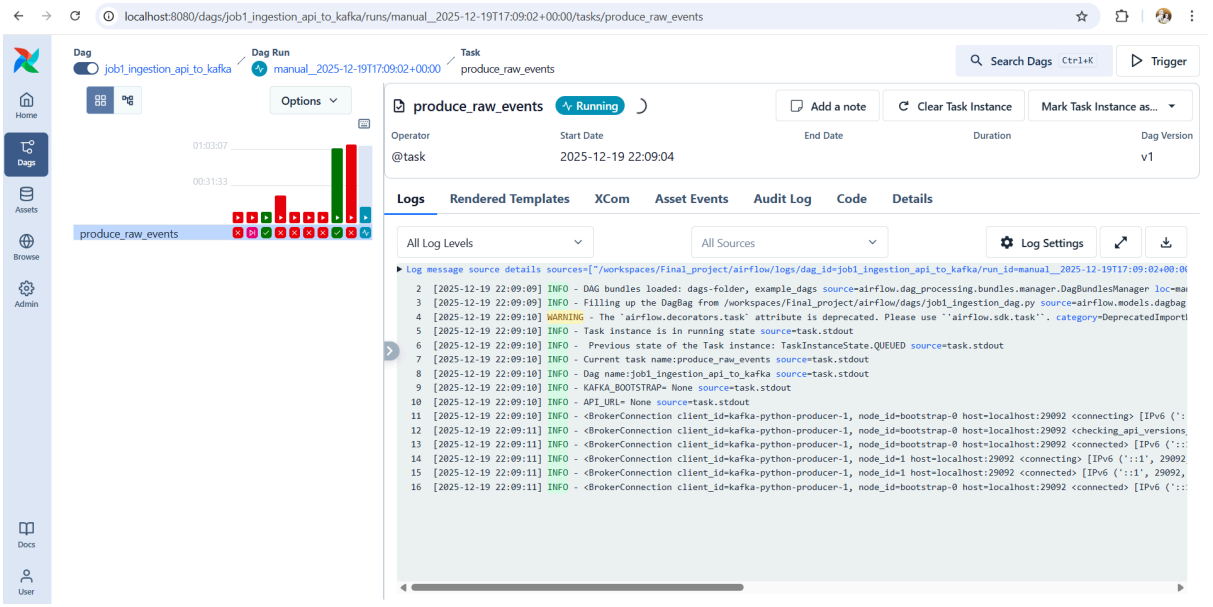

Figure 2. DAG 1 – API Ingestion to Kafka (Job 1)

This figure shows the execution of DAG 1, which is responsible for ingesting data from the external API and publishing it to a Kafka topic.

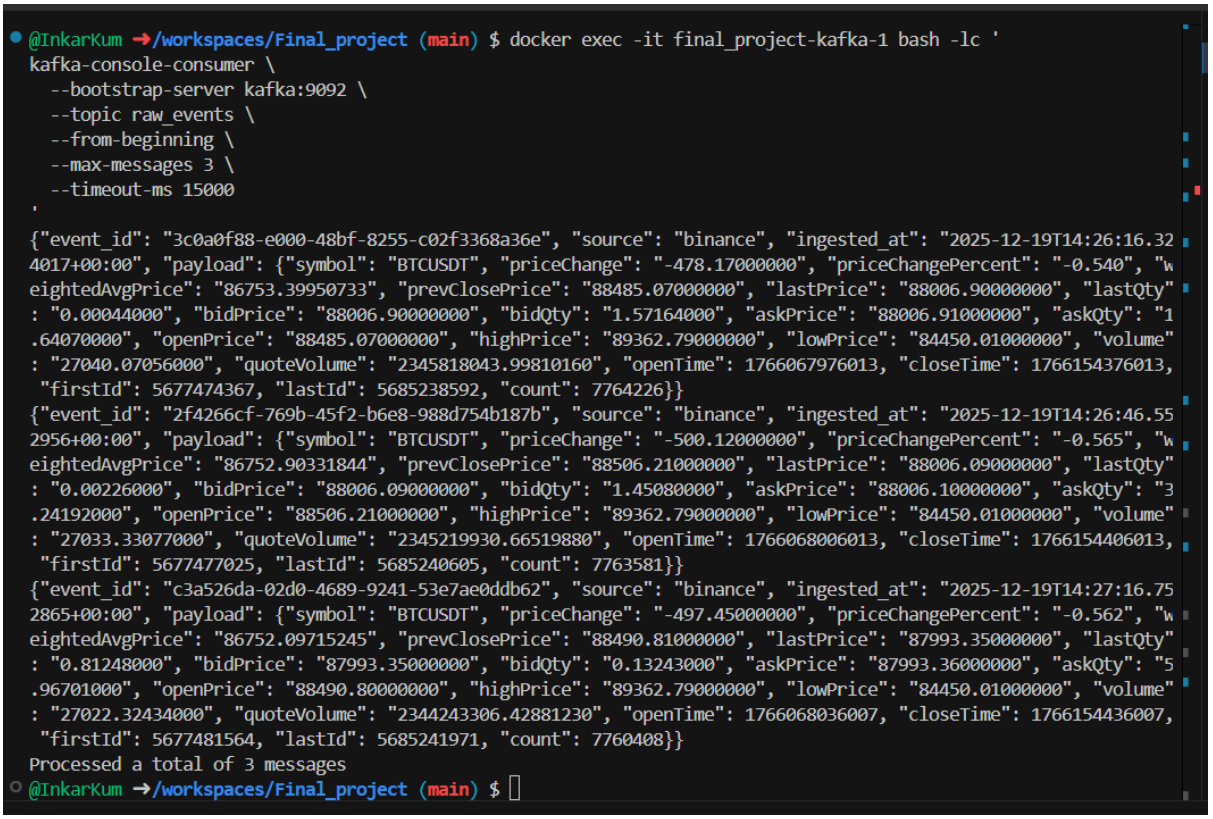

Figure 3. Kafka topic inspection (raw_events)

This figure demonstrates direct inspection of the Kafka topic using the Kafka console consumer. The output shows multiple raw JSON messages produced by Job 1, each containing market data fields such as symbol, prices, volumes, timestamps, and trade statistics.

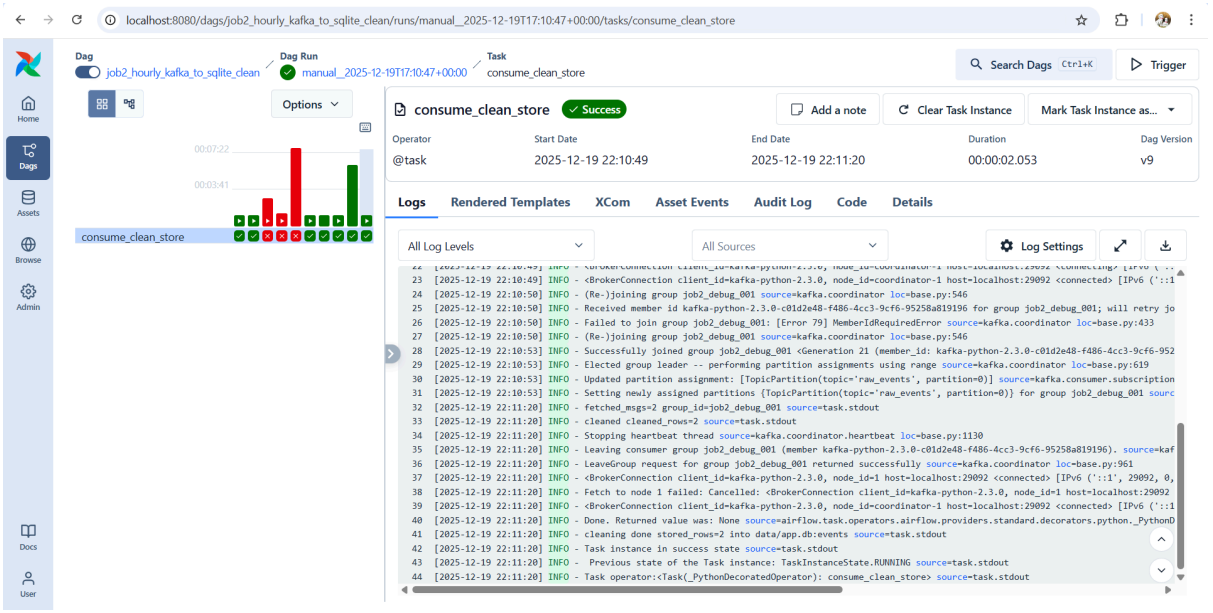

Figure 4. DAG 2 – Kafka to SQLite Cleaning and Storage (Job 2)

This figure presents DAG 2, which consumes raw Kafka messages, cleans and normalizes the data, and stores it in a SQLite database.

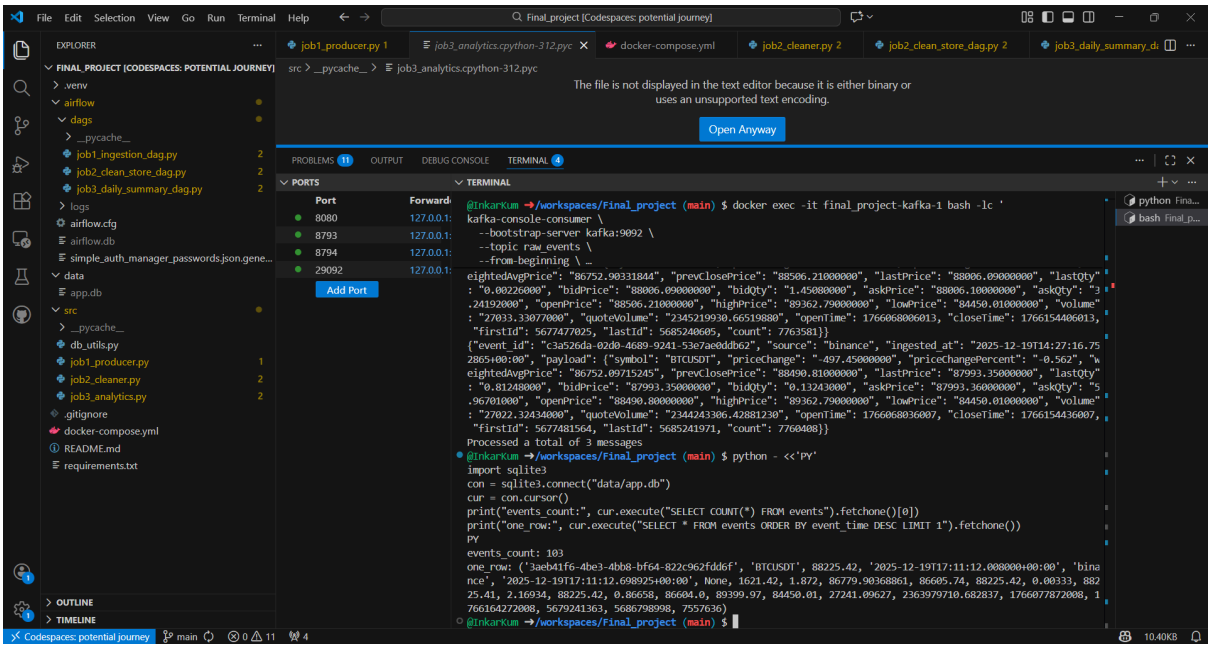

Figure 5. SQLite verification – events table

This figure shows a direct query of the SQLite database after Job 2 execution. The query confirms that cleaned records are stored in the events table, including price, symbol, timestamps, and additional numeric fields extracted from the API response.

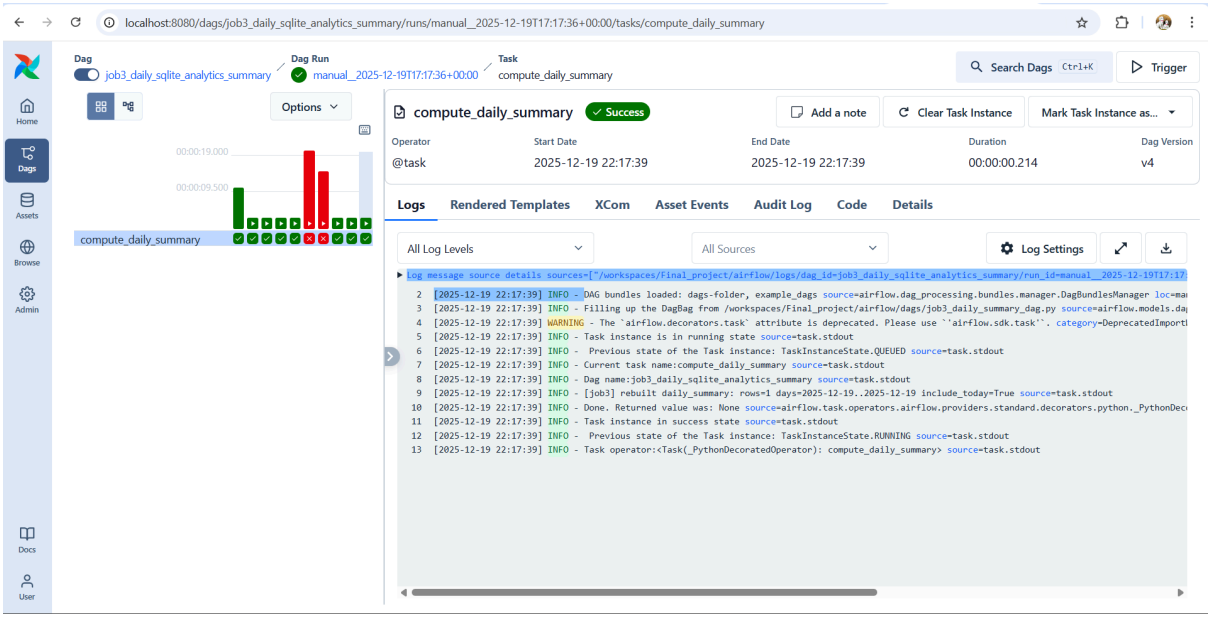

Figure 5. DAG 3 – Daily Analytics and Aggregation (Job 3)

This figure shows DAG 3, which performs analytical processing on the cleaned data stored in SQLite.

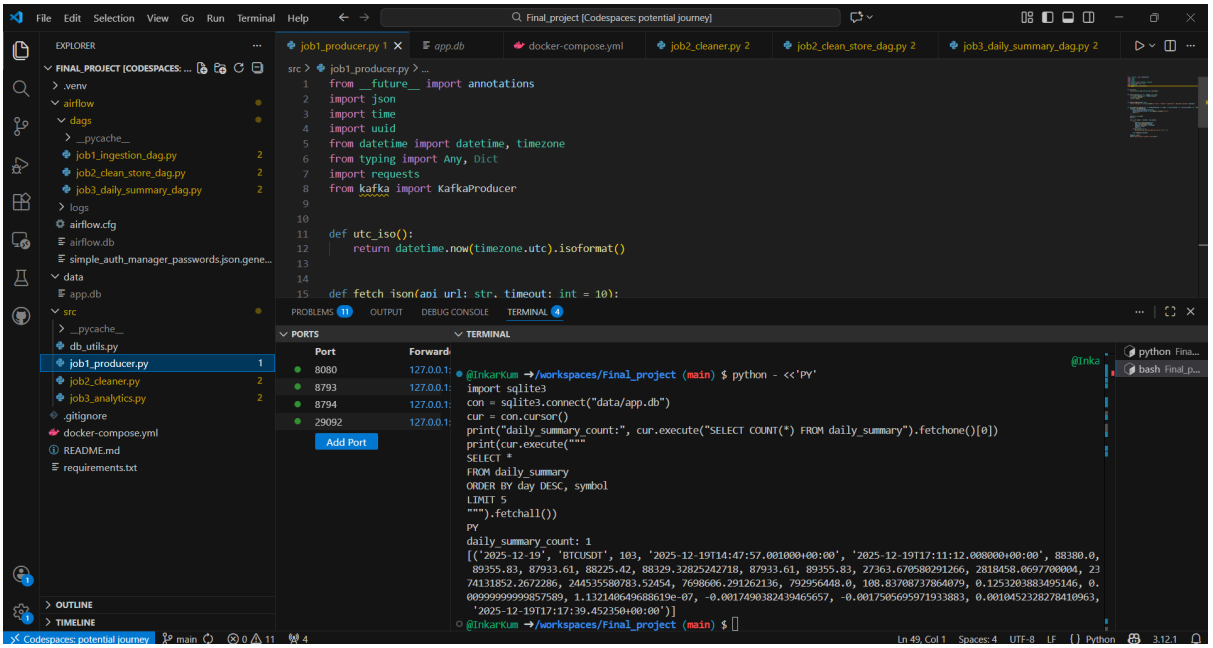

Figure 6. Job 3 – Daily analytics and aggregation

This figure shows Job 3 execution, which reads cleaned event data from SQLite and computes daily analytical metrics such as minimum, maximum, average price, and record counts per symbol. The task successfully rebuilds the daily_summary table on each run.