

# Data Engineering Project Report: Real-Time Car Park Analytics

## Team Members:

1. Smetov Damir
2. Ravshanbekov Assadbek
3. Yerken Yarmukhamed

**Date:** December 19, 2025

**Project Repository:** [https://github.com/DamirSmetov/Final\\_Project](https://github.com/DamirSmetov/Final_Project)

---

## 1. API Justification

The project utilizes the **Transport for NSW Car Park API (v2.3)**. This data source was selected because it perfectly aligns with the project requirements:

- **Update Frequency:** Data is updated every 1–10 minutes (triggered by counter changes at parking facilities), which is ideal for demonstrating real-time streaming via Kafka.
- **Reliability:** The API is provided by the NSW Government, ensuring high uptime and professional documentation.
- **Format:** It returns structured JSON data, facilitating efficient parsing and cleaning.
- **Value:** It provides real-world occupancy metrics (Total spots vs. Occupied spots), allowing for meaningful urban traffic analytics.

## 2. Kafka Topic Schema

To bridge the gap between data ingestion (Job 1) and data processing (Job 2), we implemented a Kafka topic:

- **Topic Name:** raw\_events
- **Message Format:** JSON
- **Key Fields:**
  - facility\_id (Integer): Unique identifier for the parking facility.
  - tsn (String): Transit Stop Number.
  - total (Integer): Current number of occupied spaces.
  - spots (Integer): Total parking capacity.
  - MessageDate (String): Original timestamp from the API.

## 3. Data Cleaning Rules

In Job 2 (ETL process), the following cleaning and transformation rules are applied using the Pandas library:

1. **Type Validation:** All numerical values (total, spots) are explicitly cast to Integers to prevent calculation errors.
2. **Null Handling:** Any record containing null values in critical fields like facility\_id or total is dropped.
3. **Feature Engineering:** A new field, available\_spots, is derived using the logic:  $\text{Availability} = \text{spots} - \text{total}$ .
4. **Time Normalization:** The MessageDate field is converted to a standard ISO-8601 format to ensure consistency within the SQLite database.

**4. SQLite Schema**

Data is persisted in the app.db database within two specific tables:

**Table 1: events (Cleaned Transactional Data)**

- id: INTEGER PRIMARY KEY
- facility\_id: INTEGER
- occupancy: INTEGER (Current vehicles)
- total\_spots: INTEGER
- available\_spots: INTEGER
- recorded\_at: DATETIME

**Table 2: daily\_summary (Aggregated Analytical Data)**

- id: INTEGER PRIMARY KEY
- date: DATE (Unique per day)
- facility\_id: INTEGER
- avg\_occupancy: FLOAT (Daily average)
- peak\_occupancy: INTEGER (Max vehicles recorded)
- peak\_hour: INTEGER (Hour of day with highest traffic)

---

**5. Implementation Evidence (Screenshots)**

**5.1. Airflow DAGs Overview**

DAG Name	Schedule	Latest Run	Next Run
job1_ingestion_dag	*1 * * * *	2025-12-19 14:42:00 ✓	2025-12-19 14:43:00
job2_clean_store_dag	0 * * * *	2025-12-19 14:41:17 ✓	2025-12-19 15:00:00
job3_daily_summary_dag	0 0 * * *	2025-12-19 14:42:09 ✓	2025-12-20 05:00:00

This proves all three DAGs are active: ingestion, cleaning, and analytics

## 5.2. Job 1: Ingestion Success (API to Kafka)

**produce\_data**  
PythonOperator  
✓ success

**produce\_data** ✓ Success

Operator	Start Date	End Date	Duration
PythonOperator	2025-12-19 03:15:20	2025-12-19 03:15:33	00:00:26.007

Dag Version: v1

All Log Levels: All Sources: Log Settings

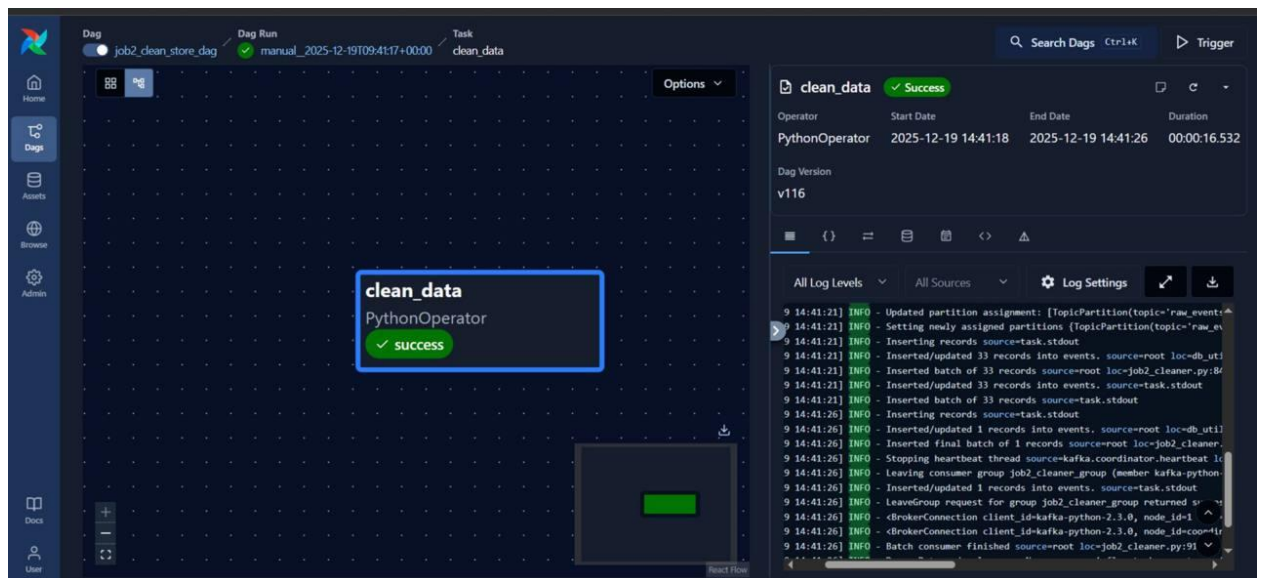
```

15:32 INFO - Fetched facility 38 source=root loc=job1_producer.py:44
15:32 INFO - Fetched facility 38 source=task.stdout
15:32 INFO - Data for facility 38: {'tsn': '2151161', 'time': '819411279', 'spo
15:33 INFO - Fetched facility 39 source=root loc=job1_producer.py:44
15:33 INFO - Fetched facility 39 source=task.stdout
15:33 INFO - Data for facility 39: {'tsn': '217426', 'time': '819411245', 'spo
15:33 INFO - <BrokerConnection client_id=kafka-python-producer-1, node_id=1 hos
15:33 INFO - <BrokerConnection client_id=kafka-python-producer-1, node_id=1 hos
15:33 INFO - Sent 26 records to Kafka source=root loc=job1_producer.py:62
15:33 INFO - Completed fetching and sending data for 34 facilities source=task.
15:33 INFO - Sent 26 records to Kafka source=task.stdout
15:33 INFO - Completed fetching and sending data for 34 facilities source=task.
15:33 INFO - Done. Returned value was: None source=airflow.task.operators.
15:33 INFO - Task instance in success state source=task.stdout
15:33 INFO - Previous state of the Task Instance: TaskInstanceState.RUNNING
15:33 INFO - Task operator:<task(PythonOperator): produce_data> source=task

```

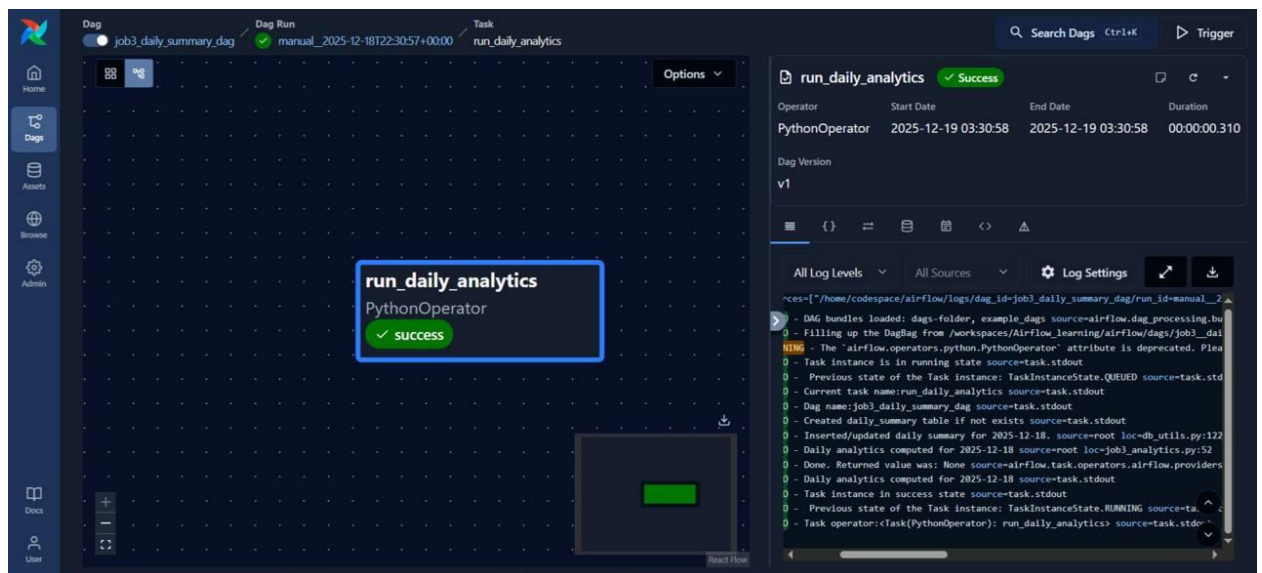
This shows the logs confirming the successful fetch from the TfNSW API

## 5.3. Job 2: Cleaning & Storage (Kafka to SQLite)



This shows the consumer logs and the successful insertion of rows into the database

## 5.4. Job 3: Daily Analytics Performance



This confirms the aggregation logic for the specific date, such as 2025-12-18

## 6. Conclusion

The pipeline successfully integrates real-time streaming (Kafka) and batch processing (Airflow). All components are fully synchronized and operational within the GitHub Codespaces environment. The project meets all technical criteria for high-frequency data ingestion and automated reporting.